ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

UNIVERSITÀ
DEGLI STUDI
DI TORINO

Doctoral Dissertation
Doctoral Program in Pure and Applied Mathematics (XXXIV cycle)

A Theory-Guided Approach to Explaining and Modeling Deep Neural Networks

Antonio Mastropietro
* * * * * *

Supervisor
Prof. Francesco Vaccarino

Doctoral Examination Committee:

Prof. Roberto Fontana - Politecnico di Torino

Prof. Patrizio Frosini - Università di Bologna

Dr. Daniela Giorgi, Referee - CNR, Pisa

Dr. Giovanni Petri, Referee - Centai Institute, Torino

Politecnico di Torino
19-10-2022

I hereby declare that the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

. . . . . .

Antonio Mastropietro

Turin, 19-10-2022

Abstract

Drawing benefits from data has become a widespread priority for industry and science. The discoveries of novel methods, technologies, and algorithms offer possibilities to derive value from data, so much to portray a third industrial revolution. From a scientific perspective, data-driven technologies suggest opportunities to move on in the knowledge of the natural world. Therefore, the brand new Data Science subject has emerged as a prominent player in the modern era. Among the other data-driven innovations, Machine Learning (ML) is one of the most disruptive and impactful methodologies to extract value from data. This work focuses on Supervised ML, aimed at computing predictions by approximating a function of interest. Inside the Supervised class, Deep Learning (DL) has significantly enriched the application potential of data-driven methods. Nonetheless, the usage of ML in critical domains, like healthcare, social media, industrial production, or automotive, has raised concerns about new emerging risks. Hence, researchers have oriented on improving the understanding of ML methods, especially the DL class. Explainable Artificial Intelligence (XAI) is a framework suggesting methods to overcome the faced obstacles. In practice, it aims at designing techniques able to give insights into ML predictions. XAI has reached considerable capabilities to infer the important variables for an ML prediction. A recent breakthrough in XAI is an algorithmic class based on the solution concept of cooperative game theory called Shapley value.

From a scientific perspective, ML methods are ideal companions for theory-guided science. Indeed, scientific disciplines seek relationships among quantities of a studied phenomenon. Instead, ML models learn statistical linkages between variables using adaptive algorithms that process observational data. Theory-Guided Data Science (TGDS) is a paradigm oriented at improving Data Science generalization capabilities and deepening the scientific knowledge through data-driven models. Two interactions arise, with opposite orientations, to integrate the better from theory-guided and data-driven methods. The first consists of learning a data science model and using its explanations to interpret the learned patterns. Thus, it helps the researcher to build hypotheses, to simplify assumptions, or to obtain an insightful perspective on the studied phenomenon. The second interaction consists of informing data science models with theoretical understanding. Indeed, a data-driven model is commonly blind to known connections. The insertion of knowledge enhances the learning capabilities, it supports dropping biased models, and it cleans the algorithmic design.

After an extended introduction and a review of XAI motivations and methods, this work provides examples of the two mentioned interactions. The second chapter employs XAI to improve a numerical model simulating a physical phenomenon. In

particular, it shows how to augment theory-guided modeling by leveraging explanations of ML model predictions. The third chapter illustrates a new graph neural network architecture that can leverage the known structure of the input variables. The presented experiments show the advantages of the architecture compared with baselines on examples of graph regression task. The last chapter proposes a new concept for game theory to fairly allocate worth to coalitions, inspired by the notable Shapley value solution concept for the fair allocation to players. The proposed solution, named $\mathcal{X}$-Shapley, is derived by extending the view provided by a recent paper in cooperative game theory, which has shown an appealing parallel with graph theory via the Hodge decomposition. The chapter thoroughly investigates the properties of the $\mathcal{X}$-Shapley, and it proves characterizations similar to the original Shapley value. $\mathcal{X}$-Shapley could provide significant advancement for XAI in future research for a better understanding of ML models.

Per Francesca

Contents

Chapter 1

Introduction

Digital innovations have provided the modern era with plenty of data sources. Data pervade increasingly every aspect of work and social life [160]. Managing, sorting, and extracting benefits from data has become a widespread priority for industry and scientific research. The discoveries of novel methods, technologies, and algorithms offer many possibilities to derive value from data, so much to portray a third industrial revolution. From a strictly economic point of view, the impact of data value has become predominant, to such an extent that economists have defined data as the new oil [187].

Among the other data-driven innovations, Machine Learning (ML) constitutes one of the disruptive and impactful methodologies to extract value from data [99]. In particular, this work focuses on Supervised ML, a class that provides predictions by approximating a function of interest. The approximation is built by progressively learning the relationship between the independent and the dependent variable from statistical samples collected about a phenomenon of interest. The generalization capability of a trained algorithm allows inferring predictions and building information from data. Inside the Supervised class, the development of Deep Learning (DL) has taken the application potential of data-driven methods to a higher level [122]. On the other hand, applying such technologies to critical domains, like healthcare, social media, industrial production, or automotive, has raised concerns about the new risks faced by a human subject in the loop [62]. Hence, the research community has oriented its effort toward improving the knowledge and understanding of ML methods, especially in the DL class. The brand new Data Science (DS) subject has emerged as a prominent player in the modern era.

Furthermore, from a scientific point of view, the development of ML technologies offers many opportunities to move on in the natural world understanding [44]. Indeed, the data-driven class represents an ideal companion for theory-guided and experience-based science. The opportunities come together with challenges to the practical involvement of those methods in scientific disciplines. Indeed, data are a raw representation of an observation. They require a conversion to step up to

the level of information or knowledge. That is, to increase the efficiency of comprehension and decision processes. The above argument recalls the types of the hierarchical description of the content of human mind: Data, Information, Knowledge, and Wisdom (DIKW): [3, 162]. But the more complex the data, the more the algorithms used to process them become significantly opaque [91]. Thus, ML algorithms usually allow climbing up in the mentioned hierarchy only from data to information. They face restraints in escalating to knowledge, that is, the level where scientific understanding starts to come into play. Explainable Artificial Intelligence (XAI) is a class of algorithms providing a roadmap to overcome those obstacles. In practice, it consists in designing a framework able to give insights into ML predictions, in such a way to derive knowledge from information [62]. XAI has been the subject of an extensive endeavor of investigations [92]. A recent breakthrough in XAI is an algorithmic class based on the cooperative game theory solution concept named Shapley value [170]. In particular, the parallel between Shapley value and feature importance provides an attractive trial to reach theory-grounded guidance for XAI [126].

In this context, there are potential intertwines between theory-guided and data-driven methodologies for researchers. Theory-Guided Data Science (TGDS) is a paradigm oriented to improving Data Science generalization capabilities. In addition, TGDS suggests a way to enhance the resources available to deepen the scientific knowledge by means of data-driven predictive models [112]. Indeed, scientific disciplines seek relationships among measured quantities of a specific phenomenon. Theory-based models commonly represent such connections, incorporating cause-effect links between variables validated through empirical experiments or theoretical deductions from first principles. These models are feasible for describing well-understood phenomena according to scientific disciplines. To predict the evolution of a studied process, the recipe of a theory-guided method consists of building a numerical model, commonly named a simulator. The simulator assimilates the equations of the phenomenon to obtain the desired outcome. Numerical models can be computationally intensive, and it is common to simplify the principles to make their calculations feasible. On the other hand, data science models suit for analyzing events when a set of abundant representative observations is available and is the leading source. They are usually employed to learn statistical relationships between variables using an adaptive algorithm that processes all the observational data.

Two intriguing interactions between the mentioned approaches arise, with opposite orientations, to integrate the better from theory-guided and data-driven models. The first interaction consists of learning a data science model and using its explanations to interpret the learned patterns. The XAI algorithmic class has reached the considerable capability to infer the most significant variables for a model prediction. The derived interpretation helps the researcher to build further hypotheses

about stimulating relationships between variables. In addition, it can aid in simplifying the numerical model, ignoring insignificant variables for the phenomenon at hand. In particular, a suggested pipeline is the following: a data-driven ML model takes as input the output of a theory-based model; then, the statistical analysis of the ML model, with the help of the XAI method, allow down scaling the theory-based model, fastening its computations and giving insights about the modeled phenomenon. Finally, the interpretation contributes to obtaining a different perspective on the studied phenomenon, thus enhancing the curiosity, imagination, and creativity that drives the scientific investigator. The second opposite interaction is to inform data science models with the knowledge already discovered by theoretical arguments. Indeed, a data-driven model is commonly blind to elementary scientific connections. It struggles to provide grounded physical plausible relationships because only available data determine its view. Therefore, inserting knowledge from the theory-guided approach boosts the capabilities of a data science model to learn mechanistic linkages between variables. Moreover, it helps to drop the unfeasible learning models at the source, excluding biased models and cleaning the design of the algorithmic architecture.

This work provides examples of the two discussed orientations where theory-guided and data science models interact. The driving motivation is to provide perspectives on the composition of the two approaches. The following presents the work subdivision in chapters. After a review of XAI, Chapter 2 and Chapter 3 deal with the two recalled approaches; instead, Chapter 4 suggests an extension of the Shapley value in cooperative game theory to allocate worth to participating coalitions, motivated by the possible impacts on XAI. A concluding section accompanies each chapter.

The prologue of the thesis, in Section 1.1, presents a thorough review of the literature about XAI, inspired by survey works about the topic [62, 91]. It retraces the motivations for the development of this new algorithm class: from the need for trust for critical applications of artificial intelligence to the novel opportunities offered to developers and scientists. Furthermore, it proceeds debating the differences between the various notions of explainability and interpretability for ML models. Then, it defines the class of attribution methods, oriented to assigning a relevance score to the input features processed by an ML model, comparing them with feature selection algorithms. The discussion presents validation methods for explanations, to ensure trust for XAI by itself. Indeed, the investigator should not deem the explanations furnished by XAI methods as oracles.

Chapter 2 presents an example of the application of XAI for gaining insights into a numerical model derived from theoretical knowledge about a studied phenomenon. It is mainly based on an already published work of the author [34], together with Stefano Berrone, Francesco Della Santa, Sandra Pieraccini, and Francesco Vaccarino. An earlier version of the same paper have appeared before [32]. It moves on by reviewing a particular class of XAI algorithms, named Layer-wise Relevance

Propagation (LRP), oriented to single instance explanations. Starting from the related literature [18, 133], it proposes a new formal point of view of the algorithm class and derives a novel global LRP explanation algorithm. Finally, it shows the experimental results over a case study numerical model. In particular, the theory-based model of interest for the experiments is the Discrete Fracture Network (DFN). DFNs are popular network models for performing flow simulations in underground fractured media. An interesting problem in studying DFNs is the backbone identification. A DFN backbone is a suitable subnetwork whose characteristics approximate the ones of the original network [178]. The proposed backbone identification methods use an Artificial Neural Network (NN) model and an LRP algorithm. Specifically, the experiments show the properties of the global LRP algorithm for the NN in this context and apply a possible generalizable validation method. Inspired by the promising results, the chapter discusses a general framework for applying ML and XAI to numerical simulator models. Besides, it suggests a refinement for the DFN test cases at hand.

The goal of Chapter 3 is to present a specific possibility of inserting knowledge into an ML model, concretely into NN architectures. This chapter rests on a paper of the author [33] written together with the same co-authors listed above. The main subject of this chapter is a novel NN architecture named Graph Informed Neural Network (GINN). The GINN architecture belongs to the class of Graph Neural Networks, and it is conceived for regression tasks over datasets endowed with a graph structure. GNNs apply to node classification, graph classification, or link prediction problems. The graph regression tasks illustrated in the chapter represent a new challenge for a GNN. The Graph Informed (GI) layer, where the data graph adjacency matrix informs the neural connections, marks the novel architecture. GINNs are inspired by Convolutional Neural Networks: in particular, similarly to spatial GNN, they draw from the intuition of switching off specific connections between nodes of subsequent NN layers, to improve the adaptability of the NN to the task of interest. The chapter builds the definition of the GI layer from a basic form, having one input feature and one output channel, to a generalized tensor formulation. Furthermore, it discusses the GINN properties concerning the graph data at hand. The experiments analyze GINN models and Multi-Layer Perceptron (MLP) architectures, the latters assumed as a baselines for graph regression tasks. In particular, it compares two different case studies of graph regression tasks. The first case study is the stochastic maximum-flow problem. Deriving from the classical maximum-flow problem [116], the chapter describes a variation where the edge capacities are random variables. The graph regression task devises from the stochastic version of the maximum-flow problem. The second graph regression task is the DFN model approximation already described in Chapter 2. The experimental comparison between the GINN and the baseline architectures shows the advantages that the novel proposed architecture delivers for the tasks. Therefore, the proposed GINN architecture is proven to be a significant, useful contribution to the class of

spatial-based graph neural networks.

Chapter 4 focuses on cooperative game theory. It is motivated by the known relationships between the Shapley value solution concept and XAI. It illustrates a novel paradigm for allocating a transferable utility in a cooperative game to coalitions, and it is the result of a joint work with Francesco Vaccarino. The paradigm builds upon and compares with the famous Shapley value, named after the Economics Nobel Prize-winning Lloyd S. Shapley. The Shapley's solution concept deals with the problem of allocating the payoff gained by a coalition of rational players to the single participants. The chapter reviews three known characterizations of the Shapley value. The first characterization originates from the Shapley seminal work [170], which relies on the axiomatization of the desirable fairness property that a value allocation should satisfy. The chapter discusses the little variations of the axiomatization from the classic literature of cooperative game theory [69, 89, 96, 148]. The second classic characterization reviewed in the chapter is a property of the stability of the Shapley value, where the equilibrium is attained by negotiation among players through objections and counter-objections to the candidate allocation. This characterization relies on the balanced contribution property, presented firstly by the works of Myerson et al. [138, 136]. The presentation of this part follows the cooperative game theory textbook of Osborne et al. [148]. The third characterization recalled is a recent link proposed by [180] between the problem of the allocation among players and graph theory. First, the authors look at the marginal contribution of a player to a coalition as a discrete differential over an oriented hypercube graph describing a cooperative game, where the graph represents the Hasse diagram of the inclusion relation between coalitions. Second, the authors show that isolating the grand coalition entry of the vector games derived from the combinatorial Hodge decomposition of the mentioned hypercube graph allows attaining the Shapley value. After reviewing the classical Shapley value, the chapter proposes a new allocation paradigm. In particular, it starts deriving a new differential, parallel with the one of [180], on the transitive closure of the Hasse diagram, which is a new oriented graph describing a generic cooperative game. Then, it describes the properties of the vector games derived from the combinatorial Hodge decomposition of the new graph representation of the cooperative game. Furthermore, the chapter illustrates how to compute the candidate solution concept for coalitions analogous to the Shapley value. The simple analytic formula defines explicitly the newly denominated $\mathcal{X}$-Shapley allocation (Chi-Shapley or Coalitional Shapley). Proceeding further, it presents two new characterizations resembling the recalled ones for Shapley value. The first rests on the concept of the game map: $\mathcal{X}$-Shapley happens to be the unique game map satisfying a set of fairness axioms inspired by the ones of values for players. The second extends the cited balanced contribution property to coalitions: again, $\mathcal{X}$-Shapley is the only efficient game map with this property. Then, the chapter proves other properties of $\mathcal{X}$-Shapley,

useful for its understanding. Finally, it discusses proposals for supplementary investigations in cooperative game theory and XAI.

This following introductory Section 1.1 describes the class of XAI, to provide a solid foundation useful for the discussion of Chapter 2 and Chapter 4.

1.1 Opening the black-box: Explainable Artificial Intelligence (XAI)

This section summarizes the main peculiarities of XAI to review the state of the art and to summarize the main open questions. In particular, Section 1.1.1 discusses the motivation behind XAI methodologies.Second, Section 1.1.2 reviews a subclass of XAI algorithms from the literature and compare with the ones of feature selection. The reviewed subclass defines attribution methods: they aim at providing explanations for predictions of Supervised ML models by assigning a relevance score to each input feature. Finally, Section 1.1.4 discusses the need for explanation validation, starting from recent studies from the literature. Then, the same section proposes a new validation method whenever the data generating distribution comes from a numerical model simulating a physical phenomenon.

1.1.1 Why Explainable Artificial Intelligence?

Machine Learning (ML) models have gained increasing attention in the last few years due to the outstanding results obtained in many fields. Among the others, the most significant and impactful outcomes have emerged in domains represented by high-dimensional data, like computer vision and natural language processing. Such results have been made possible by the increasing availability of data and the rise of hardware performance, together with optimization algorithms that allow the efficient training of ML models [62]. The benefits of the application of ML models on such domains are nowadays evident in everyday life in sectors like automotive, healthcare, finance, and social media, to name a few.

Since 2012, Deep Neural Networks (DNNs) have emerged as the most mighty class of ML models of supervised learning on the high-dimensional data domains listed above [120, 122]. Nonetheless, DNNs merge their prediction power with increased model complexity. The complexity of the DNNs model class derives from the inherent opaqueness of their abstract prediction mechanisms. Thus, DNNs lack transparency about their inner workings and are black boxes for the user. Similarly, Support Vector Machine models (SVMs) [144] are deemed as a black-box family because of the abstract nature of the extracted features used for classification or regression tasks. The highly non-linear behavior of the approximated decision boundary or regression function is a crucial property of both DNNs and SVMs to deem them as black boxes. Other ML model classes, like linear models or decision

trees, have lower complexity and higher clearness of the working logic from the user's point of view. Notably, complex models are often not suitable and accurate for all data generating processes, as reported and clearly explained in [163]. In particular, this happens when the data is structured, and the observed features are valuable for the target variable [14]. In some sense, this is common in industrial scenarios, where the disposable features are constrained and come from very manageable physical problems [56]. In [126], the authors observe that the best explanation of a simple model is the model itself: it is a perfect and complete representation, it is transparent and easy to understand. However, transparent ML model classes are less flexible, especially in the high dimensional domains mentioned before. Indeed, they bear an intrinsic bias, and the increased data availability becomes worthless. Therefore, a dualism between predictive performance and transparency arises; a careful choice of the ML model class is decisive for the practical outcome [62].

When applying DNNs to critical domains, like medicine, finance, or industry, the complexity of DNNs raises concerns about the user's trust towards this class of ML models [62]. Model predictions could reflect possible systematic bias in the data. Training an ML model on historical datasets labeled via human decisions could lead to the discovery of endemic preconceptions [150]. The model complexity can shade the training data distortion and thereupon impede the identification of the bias causes. Dangerous consequences can occur, especially when dealing with sensitive data. Some applications have already experienced controversial situations [91]. In a social context, like the hiring process of human resources in a company, the loan approval for banking, or the medical illness prediction, the reasons for the decision are fundamental [125]. The user in charge of adopting a decision supported by an automatic system should be able to give explanations about its predictions. As an attempt to rule on the matter, European Union has defined the right to explanation in the General Data Protection Regulation (GDPR) [47]. The regulation is a step to limit and counteract possible problems appearing when applying nontransparent algorithms to sensitive, crucial, and potentially life-threatening situations. Therefore, interpretability is needed to afford trust to decision-makers dealing with predictions in critical areas. While it is easy to express the actual training objective function for the computation, the concept of trust is hard to formalize and quantify for the computational task [125]. Interpretability and explainability are intermediate targets easier to tackle [62]. The horizon is to cross-check the overall system trust according to qualitative criteria whenever a second source of prediction and explanation is available.

The interest in interpretability and explainability surpasses the sole user's trust. Indeed, the complexity of DNNs makes it hard to debug and improve the prediction model [125]. The opportunity to explain the model predictions is desirable because it allows designers and developers to gain knowledge about prediction errors and clues about their mistakes before the eventual deployment [125]. Therefore, they can manage to analyze inaccurate predictions by model explanations, and finally

provide founded hypotheses about improvement directions. In addition, they can avoid putting in production a model having limited generalization capability.

A last usage of explanation is the possibility of learning from the ML predictive model about the data generating process. In fact, this is particularly meaningful when the target of the ML model development is not limited to furnishing predictions but also analyzing the natural world; in other words, to infer properties or insights about a natural phenomenon. In general, abstracted explanations are suitable to generate hypotheses and potentially find significant properties of the studied data-generating processes. Indeed, the training of a Supervised ML model could be oriented to explore the underlying data structure as the final aim. This task is indirectly related to Unsupervised Learning [125], in the sense that the label objective represents weak supervision.

A recent effort of the research community points out that the advancement of scientific understanding could benefit from data-driven models. The paradigm of Theory Guided Data Science (TGDS) [112] leverages scientific knowledge to improve the effectiveness of ML models. In addition, TGDS suggests that interpretability and explainability can provide novel domain insights. Translating learned relationships from data to scientific theories and hypotheses requires interpretability. Indeed, the epistemological advancement needs proof of the physical cause-effect mechanisms between variables. Therefore, the usage of model explanation is fundamental for the purpose. The numerical models deriving from theoretical knowledge could be the best companion for the data-driven models. Focusing on physical sciences, a numerical model derives from a set of differential equations describing the studied phenomenon, commonly called a simulator. The link between physical sciences and ML has significantly tightened in recent years [44] TGDS proposes to employ a hybrid approach: merging a simulator and data-driven modeling. This approach is interesting for the new industry 4.0 revolution because the scientific knowledge discovered in this wise carries to the digital twins used in industrial applications.

Summarizing, interpretability, and explainability are effective to (i) ensure the user trust (ii) allow model debugging (iii) provide insights about the data generating process [126].

1.1.2 What is Explainable Artificial Intelligence?

This section presents a tentative clarification of the concepts related to interpretability and explainability. In particular, it set up on recent survey papers [61, 62, 125]. The recent literature uses different notions for explainability and interpretability for ML models. Nonetheless, there is no agreement about their formal definitions. In [125], interpretability is declared not a monolithic concept. Instead, it figures as composed of several ideas that need disentanglement. In reality, many papers proclaim the interpretability of analyzed ML models axiomatically without further arguments. In [61], the authors define the verb *to interpret* as to explain

or to present in understandable terms. Then, for ML models, interpretability is the ability to explain or present in understandable terms to humans. Literature papers use interpretability and explainability as synonyms. Yet, some papers make distinctions [62]. In [133], the authors define the interpretation as a map of an abstract concept into a domain humans can make sense of. For the same authors, an explanation is the collection of features of the interpretable domain that have contributed to produce a decision for a specified example. EU GDPR defines the legal conception of explanations as "meaningful information about the logic of processing" [47]. The regulation has practical implications that define requirements in engineering and software architectures [102]. At the same time, the kind of ML explanation methods scientists and engineers have developed might not provide the legal conception of explanations [66]. Notably, the same methodologies come partially in response to the constraints of the law. Therefore, the authors of [66] suggest splitting explanations into model-centric and subject-centric. Those notions correspond to definitions of interpretability and explainability already mentioned by [133]. In accordance, [61] mentions the definition of interpretability as representing a global understanding of the model over all the data domain. Conversely, explainability takes the role of local interpretability. In this sense, interpretability comes to play when the goal is scientific understanding or bias detection. Instead, explainability intervenes when one needs to justify a specific prediction. In this view, it appears that the EU GDPR covers only explainability. Other terms like comprehensibility [74] or transparency [125] are used in the literature, meaning some sense of the understanding of the inner working logic of the model.

An extensive presentation of interpretability and explainability goes beyond the scope of this work. However, the above discussion shows that none of the mentioned arguments enables a formal definition. They implicitly depend on the user's knowledge, preferences, and other variables in the application context. This chapter assumes the definition of [61]: interpretability means a global understanding of the model prediction on all the data domain; explainability translates into understanding a prediction coming from a single sample.

XAI is not exhausted by defining interpretability and explainability. There is a need to solve the dualism between predictive performance and transparency, as stated in Section 1.1.1. A first solution to tackle the problem is integrating transparency into the model design [62]. The name for this approach is transparent box design [91]. Two alternative schemes come at hand for this approach. The first scheme consists of providing methods using only interpretable models by design. This choice commonly drives to give up on predictive performance and restrict to uncomplicated model classes. The work of [43] shows an example of this scheme: evolutionary programming allows finding sets of simple decision rules. A second scheme consists of using a hybrid method that combines a model belonging to a transparent family and a model of a black-box one. The work of [189] presents an example of this scheme: a compound of an SVM and a logistic regression model

to accomplish a credit scoring task. The goal of transparent DNNs has driven researchers to ask when it is possible to design self-explaining DNNs [11]. At the extreme, the authors of [163] argue that integrating transparency into the model or designing inherently interpretable models is the only way to solve the risk of practical applications of ML models. In other words, the authors suggest that the intrinsic comprehensibility of ML models is the preferred way to achieve interpretability and ultimately trust.

A second solution to face the dualism of performance and transparency is to start with a trained black-box model and, in some cases, with the training data; then to propose methods to explain the prediction of the predictor ex-post. To this end, this class of methods is named post-hoc explanation methods [62]. Other works name this class as reverse engineering [91]. The work of [62] advocates that providing post-hoc explanations is a similar process to how people justify their own decisions, that is, without fully knowing the actual functioning of their decision-making mechanisms. The advantage of post-hoc explanation methods consists in keeping the original prediction power of complex models, like DNNs or SVMs, especially in high-dimensional data domains. Because of the recent hardware improvements and the increased data availability, the predictive performance benefits of such complex models are significant to justify a research effort in this direction. In practice, they endow black-box models with additional algorithms to extract information from already learned ML models.

The survey of [91] subdivides further the category of post-hoc explanations according to three sub-problems they aim to solve. According to the definitions of interpretability and explainability assumed before, this work refers to the three subproblems as *model interpretation*, *outcome explanation*, and *model inspection*. (i) Model interpretation aims at understanding the global behavior of the black-box over the data domain. (ii) Outcome explanation is related to inferring the relationship between the features of specific sample input and its prediction in a local sense. (iii) Model inspection aims to provide descriptions of the properties of the model prediction process. In particular, model inspection is appropriate when the explanation goal is to provide a model debugging instrument. Instead, model interpretation and outcome explanation are suitable when the objective is to infer scientific knowledge. Focusing on the outcome explanation and model interpretation subproblems, a post-hoc explanation method translates to querying the black-box with input samples generated in a controlled manner or using random perturbations of the original train or test dataset.

The survey of [62] suggests categorizing post-hoc explanations into *model-agnostic* or *model-specific* methods. In [91], the authors define a method as model-agnostic or generalizable when it queries and uses the black-box only by inputting samples. In other words, the method does not exploit the internal peculiarities of the black-box to build the explanations. Thus, even though it is applied to explain a particular

class of ML models, they could provide interpretations for any model kind. Examples of model-agnostic methods are KernelSHAP [126], LIME [158], Anchors [159], RISE [151], or Shapley Sampling [182]. Instead, a model-specific method is non-generalizable. Specifically, it suits only for the particular class of ML model it was designed for. In other words, model-specific methods make assumptions about the inner working of the ML model, and they leverage idiosyncrasies of the ML model representations. Examples of model specific methods are LRP [18], DeepLift [171], DeepSHAP [126], or DASP [12]. The model-agnostic methods are usually characterized by a high computational cost, although they are preferable because of their easy applicability to different ML model classes. The main aim of model-specific explanation methods is to accelerate the computations of model-agnostic methods, especially when the explanation has time constraints.

Furthermore, a post-hoc explanation method is *gradient-based* when it assumes that the ML model to be explained provides a gradient for the output prediction. Gradient-based methods collocate between model-specific and model agnostic because they are not limited to one class of ML models, but apply to all the models trained using their gradients. An example is represented by Integrated Gradients [184]. The gradient-based methods need to be distinguished from the backpropagation-based (BP-based) methods [174], linked to DNN models. Indeed, the latter modifies the BP algorithm to propagate an importance score from the DNN output to the input of the DNN, instead of the gradient of the loss function. So BP methods should be deemed model-specific because they suit for DNNs. Examples of algorithms belonging to the BP class is Saliency [173, 68, 19], Input $\odot$ Gradient [172], DeconvNet [193], Guided BP [177], Guided Grad-CAM [168], or SmoothGrad [176], in addition to the already mentioned LRP, DeepLift, DeepSHAP. In the literature, the term "saliency map" refers to an explanation method when images compose the data domain. So far, saliency maps regard mainly convolutional DNNs. This name comes from the feature relevance values: they are shown as heatmap superimposed on the original sample image to explain. Many of the mentioned explanation algorithms have standard implementations in open-source code libraries [126, 7, 119].

Finally, the last solution to provide model interpretability consists of attributing an influence score to each of the training samples of the ML model at test time. Specifically, the idea of [117] is to ask which training sample would change the loss value of the test set the most when up-weighted or down-weighted in the loss computation during training.

The taxonomy of XAI presented up to now is preparatory to the discussion and experiments presented in the following sections. The disagreement recurring in the literature causes this section to be far from exhaustive, although based on the recent surveys about the topic.

1.1.3 Attribution Methods and Feature Selection

In the context of Supervised Learning algorithms, the problems of model interpretation and outcome explanation described in Section 1.1.2 relate to the problem of finding the most important features that have contributed to obtained prediction. The link holds in the global data domain where the model is defined, or in a local neighborhood of a specific sample. In detail, the feature importance problem consists of finding an algorithm to assign a quantity to each input feature of a learning problem, representing its importance for the model to reach a specific prediction in a global or local sense. The importance value is often referred to as *relevance* in the ML literature [118, 93, 36]. The feature ranking according to the assigned relevance translates the problem into feature selection: the latter consists of finding the best features to solve the learning task, given the assumed hypothesis class.

Although the two problems seem consequent, and some literature papers considers both views [64], the two formulations need a distinction. Indeed, a difference between the XAI approach and the feature selection problem is that the former is a set of operations conducted to understand the model predictions. Instead, the latter is mainly a preprocessing phase or is involved in the training algorithm [93]. In fact, in the feature selection literature, the feature selection phase precedes or is part of the actual training algorithm. This change of perspective can be due to the historical development of the new class of training algorithms. In particular, in the early stage of the history of ML, feature selection was a significant phase in the development of an ML algorithm. With the advent of SVMs or DNNs, the focus has moved to model behavior understanding. On the other side, literature about feature selection is more developed while the one on interpretability is more recent. Therefore, the latter takes inspiration from the former to provide theories, methods, and algorithms.

The past literature has discussed many concepts of relevance. The variety of definitions depends on the several answers to the question *relevant to what?* [36]. From the literature, this chapter reports the main definitions about the conditions that make a feature be deemed as relevant [118, 36]. Here, the discussion consider the example case of a binary classification task, to give a contextual view about the topic.

Notation 1.1.1 (Assumptions for Explanation Models). Formally, assume to have an input space or data domain $A \subseteq \mathbb{R}^n$ and a target space $\mathcal{Y} \subseteq \mathbb{R}^m$. Denote the target function $\boldsymbol{F}\colon A \longrightarrow \mathcal{Y}$, and suppose to have a dataset $\mathcal{D} = ((\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_p, y_p) \subseteq \mathbb{R}^n \times \mathcal{Y}$. The samples $\boldsymbol{x}$ are drawn from the random vector $\boldsymbol{X}$, according to its probability distribution q. Denote the set of all features as N; each component of the input random variable vector, that is, each feature, as X_i; the set of features $N \setminus \{X_i\}$ as N_i. The target samples $y \in \mathcal{Y}$ are drawn from the target random variable $Y = \boldsymbol{F}(\boldsymbol{X}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$ is white noise.

Furthermore, the target function $\boldsymbol{F}$ is approximated by a Supervised ML model $\widehat{\boldsymbol{F}} \approx \boldsymbol{F}$, mapping between the same spaces of $\boldsymbol{F}$. In the case of binary classification task, $\mathcal{Y} = \{-1, 1\}$.

Notably, the Bayes classifier is the algorithm that predicts the most probable class for a given instance, according to the q data generating distribution, assumed to be known. The optimal Bayes classifier is weakly monotonic in the number of features, i.e., its error or loss decreases or remains equal when adding a new feature. So the problem of computing the feature relevance seems not significant. However, two obstacles arise in practice: (i) the underlying distribution q is unknown, and (2) algorithms attempt to find a hypothesis $\widehat{\boldsymbol{F}}$ by approximating NP-hard optimization problems. The first obstacle is related to the bias-variance trade-off [99]: estimation of more parameters (bias reduction) has to balance with an accurate estimate of the same parameters (variance reduction). The second limitation is commonly intractable, and it poses additional computational challenges.

From the theoretical standpoint, two interesting definitions of a relevant feature are worthy to mention from [118]. A feature X_i is strongly relevant if the removal of X_i alone results in a nonzero increase of the Bayes classifier loss. A feature X_i is weakly relevant if it is not strongly relevant and there exists a features' subset $S \subset N_i$ such that the Bayes classifier loss on S is worse than the performance on $S \cup \{X_i\}$. Formally,

Definition 1.1.2 (Strong relevant feature). *A feature X_i is strongly relevant if and only if there exists some x_i, y_i, n_i for which $p(X_i = x_i, N_i = n_i) > 0$, and such that*

$$p(Y = y \mid X_i = x_i, N_i = n_i) \neq p(Y = y \mid N_i = n_i)$$

Definition 1.1.3 (Weakly relevant feature). *A feature X_i is weakly relevant if and only if it is not strongly relevant, and there exists $S \subset N_i$ for which exists x_i, y_i, s with $p(X_i = x_i, S = s) > 0$, and such that*

$$p(Y = y \mid X_i = x_i, S = s) \neq p(Y = y \mid S = s)$$

Remark 1.1.4. Note that the properties of strongly or weakly relevant features do not directly provide a quantity to be measured. In other words, they do not allow computing a feature relevance for model interpretation and outcome explanation. Instead, they state when a feature is relevant according to the learning task of study, which is the binary classification task in the presented case.

The determination of a relevance value to be assigned ex-post to the features used by a model make use of the definition of explanation model [126]. As stated clearly in [126], the best explanation model of a simple model is the model itself. Instead, for more complex models, the explanation model should be an approximating simpler model, at least locally. The simpler model could possibly use a new set of simplified input features Z_j, $j = 1, \ldots, t$, on which reconstructing original

input variables X_i according to a rebuilding function h, such that $X = h(Z)$. For example, the easiest case assumes that Z_j is a binary variable with a one-to-one correspondence between Z and X, and Z_j denoting the presence or absence of the corresponding input feature X_i. More formally, $Z_j \in \{0, 1\}$, that is $t = n$ and $i \longleftrightarrow j$. When the task is outcome explanation, the function h could depend also on the sample to be explained $\boldsymbol{x}^*$, so that $X = h_{\boldsymbol{x}^*}(Z)$. To illustrate the idea behind the meaning of the rebuilding function h, consider the example from [158], where X_i are the pixel of an image. The binary variables Z_j represent the presence or absence of a superpixel, that is, a region of connected pixels. In detail, the absence means the replacement of the component of the observed sample with a neutral assignment, like the gray color, while h maps the Z_j presence/absence variables to the original pixel features X_i. Notably, the paper of [48] justifies the simplified input space by proposing a unified view of explanation by removing features. In other words, the simplified input features are a formal representation of the event of feature removal, and the explanation model measures the relevance of each feature according to the consequent change observed in the ML model. All the above can be resumed in the following:

Definition 1.1.5 (Explanation model, informally). *Fix a sample $\boldsymbol{x}^* \in \mathbb{R}^n$ to be explained. Assume simplified input features $\{Z_j\}_{j=1,\dots,t}$ and a function $h\colon \mathbb{R}^t \to \mathbb{R}^n$ such that $X = h_{\boldsymbol{x}^*}(Z)$. Given a model $\widehat{\boldsymbol{F}}$, an explanation model is an interpretable model $\boldsymbol{G}_{\widehat{\boldsymbol{F}}}\colon \mathbb{R}^t \to \mathcal{Y}$ that approximates $\widehat{\boldsymbol{F}}$ locally near $\boldsymbol{x}^*$, assigning to each simplified feature Z_j an attribution or effect $\varphi_j^* \in \mathbb{R}$ for the explained sample.*

In particular, in the case of a regression task, the set $\mathcal{Y}$ becomes a subset of $\mathbb{R}^m$, and the above definition generalizes accordingly. The definition above is clearly undetermined for the term *interpretable* characterizing $\boldsymbol{G}$, anticipated by the discussion of Section 1.1.2. An attribution method will be an algorithm used to obtain an explanation model $\boldsymbol{G}_{\widehat{\boldsymbol{F}}}$. The attributions φ_j^* represent the relevance measures used for outcome explanation or model interpretation. All the attribution methods are post-hoc explanation methods, because they do not provide a transparent view of the prediction process of the original explained model $\widehat{\boldsymbol{F}}$.

Many attribution methods proposed in the literature for the outcome explanation assume a further property on the attributions on the simplified features. This property is called additive feature attribution, and it defines a class of algorithms named additive feature attribution methods [126, Definition 1].

Definition 1.1.6 (Additive feature explanation model). *The explanation model $\boldsymbol{G}_{\widehat{\boldsymbol{F}}}$ has the property of additive feature attributions if*

$$\boldsymbol{G}_{\widehat{\boldsymbol{F}}}(\boldsymbol{z}^*) = \varphi_0^* + \sum_{j=1}^{t} \varphi_j^* z_j^*,$$

where every φ_j^*, $j = 1, \dots, t$ is the attribution to the simplified feature Z_j^* for the explained sample $X^* = h(Z^*)$

In practice, the explanation model $\boldsymbol{G}$ has the property of being an affine function in the components of the simplifying input features, and it is such that the coefficients of the affine function are the simplified features attributions. Evidently, an additive feature explanation model can be deemed an interpretable one. The attribution methods belonging to this class are many of the most recent proposed in the literature, like LIME [158], SHAP and its variations KernelSHAP and DeepSHAP [126], DeepLIFT [171], LRP [18]. Note that additive feature attribution methods use both the model and the dataset to build the corresponding explanation model $\boldsymbol{G}$. This happens because the data generating distribution q is commonly unknown: to build $\boldsymbol{G}$, the method gets samples from q and query the model in the neighborhood of the sample $\boldsymbol{x}^*$ to explain.

Now, consider the case of a data distribution generated by a numerical model. Precisely, assume that the function $\boldsymbol{F}$ is a simulating model incorporating theoretical knowledge about a physical phenomenon. As stated in the previous Section 1.1.2, this is a frequent scenario in science or industry. In this case, the ML model is a statistical model approximating the simulator. An explanation model $\boldsymbol{G}_{\widehat{\boldsymbol{F}}}$ could provide insights about $\boldsymbol{F}$ if both the approximations $\boldsymbol{G}_{\widehat{\boldsymbol{F}}} \approx \widehat{\boldsymbol{F}} \approx \boldsymbol{F}$ are sufficiently precise. In particular, the insights comes from the relevance values ϕ_j. The relevance score could be employed to improve the simulator $\boldsymbol{F}$ similarly to feature selection, that is, by providing the most significant variables X_i inputted to $\boldsymbol{F}$ from a statistical perspective. This chapter illustrates a case study to show in practice an application of the pipeline presented until now.

1.1.4 The Need for Explanations Validation

The advances in explanation methods, in particular the post-hoc class, give rise to new challenges for validating the complete training and explanation of the developed model. An ML training algorithm requires validating the ML model through a validation set to solve the bias-variance trade-off and to check the model generalization capability on the test set. A general choice is validation methods through data splitting, selected according to the complexity and computational cost of the ML model developed, for example, cross-validation [99]. Nonetheless, the validation process of an ML model does not make the same model robust: the ML model can be subject to adversarial attacks. An adversarial attack is a tentative of artificially manipulating the output of an ML model by slightly modifying the input sample [186, 87]. The more the ML model is complex, the less it is manageable and the more it is prone to manipulations. This happens especially for high dimensional data domain and highly non-linear decision boundary or regression function, like the cases where DNNs are the most powerful. The likelihood of these attacks is threatening applications of ML in areas like security [188]. The possibility of adversarial attacks strengthens the urgency of validation methods that can guarantee the ML model's predictions. Again, explainability represents a candidate solution

to this problem. Explanations could bring out evidence of prediction errors by the ML model, thus providing a defense versus manipulations.

As formalized in the previous Section 1.1.3, a post-hoc explanations in the form of attribution methods provides two steps of approximations $\boldsymbol{G}_{\widehat{\boldsymbol{F}}} \approx \widehat{\boldsymbol{F}} \approx \boldsymbol{F}$. A usual model validation investigates the approximation $\widehat{\boldsymbol{F}} \approx \boldsymbol{F}$. As commented above, the model validation alone does not prevent an adversarial attack. In case of attribution methods, a second approximation $\boldsymbol{G}_{\widehat{\boldsymbol{F}}} \approx \widehat{\boldsymbol{F}}$ comes to help. But does the explanation model $\boldsymbol{G}_{\widehat{\boldsymbol{F}}}$ provide faithful attributions to the input features? At first glance, this question could be paradoxical, because the explanation model is motivated by the need for trustworthy predictions. But the introduction of the explanation model could raise other concerns about its faithfulness to the model to be explained. In other words, in the case of post-hoc explanations, the explanation model acts like an external authority that provides insights into the furnished predictions. Because many explanation methods rely upon heuristics, they do not guarantee their authoritativeness by themselves. This lack of robustness of explanation gives rise to new problems [78]. First, isolated explanations could be plausible but misleading, misguiding the possible discovery of knowledge from the ML model. In theory, such explanations could satisfy the GDPR laws, but they do not in practice [62]. Second, they can also open the door to manipulations of explanations. These manipulations could take the form of proper adversarial attacks to explanations, used to hide bias in the explained model. An example coming from [78] illustrates an undesired outcome. Suppose that an ML model is applied in healthcare to classify a radiology image as pathological or not. An explanation method might suggest that a specific image region is important for the malignant classification (e.g., region having high relevance in the saliency map). The clinician might then concentrate on that region for investigation or treatment, or look for comparable peculiarities in other subjects. It would result upsetting when a different region emerges from the saliency map in an almost identical image, indistinguishable from the original, and classified again as malignant. Thus, a robust predictor (both images correctly classified as malignant) but a fragile explanation would still cause undesirable behavior. In the example given above, if the explanation guides the doctor's interventions (e.g., location of a biopsy), then a non-robust explanation against adversarial attacks could be a life-threatening concern.

The authors of [78] focus on adversarial attacks that introduce tiny perturbations on the input sample to explain to fool explanation methods. Other authors focus on adversarial attacks in a scenario where model-agnostic algorithms make perturbations by design on the input sample to approximate the input data distribution q [175]. The authors of [13] further extend this possibility by fooling model-specific explanation methods for DNNs to explain any input image by a saliency map of the number 42. The last two cited papers leverage the lack of representativeness of samples in the approximation of the data generating distribution. This limit is due to the sampling procedure adopted by most algorithms, that use sample data

coming outside the support q to approximate q itself.

The works of [5, 4] represent the first steps towards giving guarantees about the faithfulness of explanation methods. They propose sanity checks to show that an explanation method is dependent on the parameters of the ML model to explain or the data the model has been trained on. Surprisingly, they found that some explanation algorithm mentioned here fails the check. Furthermore, the paper of [174] shows that some BP-based explanations method fails the same sanity check, and it proves a tentative theorem to show the cause of this behavior.

Established on the recent literature, the discussion above is intended to suggest a critical view of the explanations provided by the most used algorithms. Explanation methods expose the risk of a confirmation bias towards prejudices about the cause-effect link between input and prediction. Focusing on the case when the objective is to provide insights about the data generating process, the risk highlighted could undermine the conclusion driven by the explanations. Therefore, the validation of explanations requires basing the inferred hypothesis on a solid experiment. When a simulator represents the data generating process, that is $\boldsymbol{F}$, there are several alternatives for validating the outcome of the explanations compared with those presented before. In particular, the sanity checks presented before do not use the data generating process; instead, they concentrate on the ML model to validate explanations. In the case of outcome explanation or model interpretation, the developer could possibly modify the simulator $\boldsymbol{F}$ according to the most important features given by the explanation method to be validated and applied to $\widehat{\boldsymbol{F}}$. Thus, the obtained simplified simulator $\boldsymbol{F}'$ can be compared with $\boldsymbol{F}$ via suitable metrics, depending on the domain and problem of interest, to check for the hypothesis derived from the explanation method.

Chapter 2

Explainable Artificial Intelligence for Modeling Insights

This chapter investigates the possibility offered by interpretability and explainability methods for gaining insights about the data generating process. In particular, it applies those methodologies when the data generating process is a known numerical model simulating a physical phenomenon. According to the taxonomy of 1.1.2, the focus is on a model-specific backpropagation-based explanation method defined by the algorithm of Layer-wise Relevance Propagation (LRP) [18, 133]. The algorithm is employed both for outcome explanation and model interpretation, as defined in Section 1.1.2. The complete pipeline includes the usage of a simulator $\boldsymbol{F}$ and a neural network $\mathcal{N}$ approximating the simulated data generating process. Then, the LRP algorithm is applied to $\mathcal{N}$ to provide insights about $\boldsymbol{F}$. The pipeline handle the validation requirement by using a simplified simulator $\boldsymbol{F}'$ derived from $\boldsymbol{F}$. The comparison of the results obtained by the simplified simulator and the original simulator provide clues about the insights inferred by the explanation algorithm. In particular, the presentation highlights the peculiarities of the illustrated pipeline with a specific experimental case study, allowing a discussion of the result of a model interpretation algorithm.

The explanation algorithm is thoroughly presented in Section 2.1, which provides a detailed and formal description of LRP. Then, Section 2.2 reviews the experimental case study of Discrete Fracture Networks (DFNs). Specifically, it formalizes a regression task, and it presents the Backbone identification problem. Section 2.3 presents the experimental case study and the results of the application of the proposed pipeline to the DFN Backbone identification problem. Finally, Section 2.4 discusses the obtained results, and it proposes a generalization of the pipeline for the analysis of physical phenomena.

2.1 Focus on Layer-Wise Relevance Propagation

This section focuses on Layer-wise Relevance Propagation (LRP) [18]. LRP defines a subclass of algorithms of XAI oriented to outcome explanation, as expressed in Section 1.1.2.

Before describing LRP, fix the notation for the presentation of Neural Networks (NNs) analyzed using LRP. Given a Fully-connected Neural Network (FNN) $\mathcal{N}$ made of $L+1 \in \mathbb{N}$ layers, denote them as $U^{(0)}, \ldots, U^{(L)}$ where:

- $U^{(0)}$ is the input layer;
- $U^{(L)}$ is the output layer;
- $U^{(1)}, \ldots, U^{(L-1)}$ are the hidden layers.

Let $\mathcal{N}$ be a FNN trained for approximating a function $\boldsymbol{F} : A \subseteq \mathbb{R}^n \to \mathbb{R}^m$, then let $\widehat{\boldsymbol{F}}$ denote the function corresponding to $\mathcal{N}$ at the end of the training; therefore, assuming $\mathcal{N}$ is a sufficiently good approximation, it holds $\boldsymbol{F} \approx \widehat{\boldsymbol{F}}$ or, more precisely, $\boldsymbol{F}(\boldsymbol{x}) \approx \widehat{\boldsymbol{F}}(\boldsymbol{x})$ for each $\boldsymbol{x} \in A \subseteq \mathbb{R}^n$.

Given a trained NN with function $\widehat{\boldsymbol{F}}$, LRP looks at a prediction $\widehat{\boldsymbol{F}}(\boldsymbol{x})$ and assigns relevance scores to the components x_k of $\boldsymbol{x}$, taking into account the weights and the architecture of the NN. The computed relevance scores indicate how much each x_k contributed in computing the prediction $\widehat{\boldsymbol{F}}(\boldsymbol{x})$. For a given NN model, and an input $\boldsymbol{x}$, the method assumes to have a vector $R \in \mathbb{R}^m$, defined relevance score, related to $\widehat{\boldsymbol{F}}(\boldsymbol{x})$, LRP propagates the score R backward through the NN from the output layer $U^{(L)}$ to the input one $U^{(0)}$. The propagation happens layer-wise, and it redistributes R among all the input units of $U^{(0)}$. The redistribution of R is defined by a propagation rule that takes into account the forward propagation of $\boldsymbol{x}$ through the NN and, therefore, depends on the weights and the network connectivity. The final result describes the relevance of each component x_k of $\boldsymbol{x}$. In particular, it allows highlighting the features having higher influence in the computation of the corresponding prediction made by the NN. Actually, these components are the list of values that come up to the input units from the propagation of R.

In literature, LRP is always applied with respect to one input at a time [133]: given one input $\boldsymbol{x}$ and a score R, LRP computes the relevance of the components x_k of $\boldsymbol{x}$ for the prediction of the output. Then, the relevance score depends on the input, so the most relevant components can vary changing the input sample taken into account. An extension proposed here is that the authors not only compute the relevance scores of all the inputs in a given dataset, but they also aggregate this information (see Subsection 2.1.3 and Section 2.3). The result is an approximation of the expected relevance score vector that furnishes a description about how the NN looks at the data domain space. Thus, the extension makes possible to orient the analysis not only to outcome explanation but also to model interpretation, as defined in the Subsection 1.1.2.

2.1.1 The Propagation Rule

To understand the LRP method, the mechanism that regulates the propagation of the score R backward through the NN is described here. This sequence of operations is called the *propagation rule* of the method, and it can vary according to the NN architecture. LRP has been firstly described for a classification task, and this section proposes an extension to a regression task. In addition, here the main characteristics of the propagation rule are introduced and the α-β rule is described. The same rule will be used in the experiments of the case of study presented in Section 2.3.

One of the most important aspects that characterizes the propagation rule is the criterion behind the choice of the score R with respect to the input $\boldsymbol{x}$, at the beginning of the backward propagation process, to be assigned to the neurons of the output layer $U^{(L)}$. The literature of LRP, similarly to all the BP-based explanation methods, is focused on classification task. In particular, they choose R at the beginning of the propagation to be the *logit score*, that is, the value before the softmax activation function of the output layer $U^{(L)}$ [18]. However, in this work the authors consider only the simplest case, where the starting score is equal to the output predicted by the NN with respect to $\boldsymbol{x}$. This makes easy to generalize LRP to a regression task, and it does not modify the behavior of LRP in a classification task. In the situation where the output layer $U^{(L)}$ of the network is characterized by more than one unit, the criterion is obviously generalized such that it assigns to each output unit $u_j \in U^{(L)}$ a relevance score $R_j^{(L)}$ equal to the j-th component of the prediction vector corresponding to input $\boldsymbol{x}$, i.e.:

$$R_j^{(L)}(\boldsymbol{x}) = (\widehat{\boldsymbol{F}}(\boldsymbol{x}))_j \,. \tag{2.1}$$

Then the total starting score is

$$R(\boldsymbol{x}) = \sum_{u_j \in U^{(L)}} R_j^{(L)}(\boldsymbol{x}) =: R^{(L)}(\boldsymbol{x}) \,. \tag{2.2}$$

For the ease of notation, from now on the dependency of the relevance scores on the input $\boldsymbol{x}$ is omitted.

Therefore, assuming for simplicity to have a NN without skip or residual connections, the LRP method defines a rule to propagate these scores from $U^{(L)}$ to $U^{(L-1)}$ and, more generally, from each layer to the previous one. The rule comprises the definition of messages, for each pair $(u_i, u_j) \in U^{(\ell)} \times U^{(\ell+1)}$, for each $\ell = 0, \ldots, L-1$, such that the message $R_{i \leftarrow j}^{(\ell,\ell+1)} \in \mathbb{R}$ is the amount of score $R_j^{(\ell+1)}$ that spread to unit $u_i \in U^{(\ell)}$ from unit $u_j \in U^{(\ell+1)}$. Actually, the message $R_{i \leftarrow j}^{(\ell,\ell+1)}$ represents how much the output of unit u_i sent to u_j is relevant for the prediction computation $\widehat{\boldsymbol{F}}(\boldsymbol{x})$; then, the relevance score $R_i^{(\ell)}$ of u_i with respect to $\widehat{\boldsymbol{F}}(\boldsymbol{x})$ is

given by the sum of all the incoming messages:

$$R_i^{(\ell)} = \sum_{u_j \in U^{(\ell+1)}} R_{i \leftarrow j}^{(\ell,\, \ell+1)}, \tag{2.3}$$

for each $\ell = 0, \ldots, L-1$. The messages can also be named partial relevances. Therefore, the relevance of the component x_i of the input $\boldsymbol{x} \in \mathbb{R}^n$ is given by the quantity $R_i^{(0)}$ computed starting from $R(\boldsymbol{x})$.

Due to the empirical origin of the LRP method, in literature (e.g., see [133]) the computation of the messages is usually described through examples that show many possible arbitrary formulas for the definition of $R_{i \leftarrow j}^{(\ell,\, \ell+1)}$. To the best of the authors knowledge, no formal definitions exist. Then, to facilitate the understanding of the problem, in the Subsection 2.1.2 a more general and formal definition of the messages $R_{i \leftarrow j}^{(\ell,\, \ell+1)}$ is introduced, to characterize the propagation rule. The content of Subsection 2.1.2 can be useful to the interested reader that is new to the LRP algorithm.

2.1.2 Details about Layer-wise Relevance Propagation

In this Subsection, a novel general and formal definition of the LRP messages is introduced, extending what already illustrated in Subsection 2.1.1. The message $R_{i \leftarrow j}^{(\ell,\, \ell+1)}$, from unit $u_j \in U^{(\ell+1)}$ to unit $u_i \in U^{(\ell)}$ for the LRP method, is defined as

$$R_{i \leftarrow j}^{(\ell,\, \ell+1)} := \rho\left(c(x_i^{(\ell)}, w_{ij}^{(\ell+1)}),\, R_j^{(\ell+1)}\right), \tag{2.4}$$

where $x_i^{(\ell)}$ is the output of unit $u_i \in U^{(\ell)}$ and $w_{ij}^{(\ell+1)}$ is the weight of the edge (u_i, u_j) in the FNN. The functions $\rho : \mathbb{R}^2 \to \mathbb{R}$ and $c : \mathbb{R}^2 \to \mathbb{R}$ are arbitrary functions, denoted as *relevance function* and *contribution function*, respectively.

The relevance function ρ has the role to decide how much of the relevance score $R_j^{(\ell+1)}$ has to be propagated backward to unit u_i: the higher the output of ρ (i.e. the message $R_{i \leftarrow j}^{(\ell,\, \ell+1)}$), the more relevant is u_i with respect to u_j. The partial relevance of u_i should clearly depend on the score $R_j^{(\ell+1)}$, which is the relevance of u_j, but also on how much u_i contributed to unit u_j in the forward passage. This contribution is measured by the function c, taking into account both $x_i^{(\ell)}$ and $w_{ij}^{(\ell+1)}$. In literature ([18, 133]), the most used choice of ρ is the multiplication, such that the role of c is reduced to the computation of an appropriate factor for rescaling $R_j^{(\ell+1)}$. Moreover, the function c can be different for each message. For example, c can be characterized by different parameter values varying the unit u_j considered (see parameters $z_j^{\pm}$ in α-β rule later).

The propagation rule, that the LRP method defines to compute the message $R_{i \leftarrow j}^{(\ell,\, \ell+1)}$ through ρ and c, is assumed to satisfy the following properties:

1. *Conservation* [133]: the sum of the scores propagated from each layer to the preceding ones remains equal, that is:

$$R^{(0)} = \cdots = R^{(L-1)} = R^{(L)} = R(\boldsymbol{x})\,, \tag{2.5}$$

where, for each $\ell = 0, \ldots, L$, define $R^{(\ell)}$ as

$$R^{(\ell)} = \sum_{u_i \in U^{(\ell)}} R_i^{(\ell)}\,. \tag{2.6}$$

The introduction of the conservation property is important because it highlights that LRP actually compute a decomposition of the outputs (assuming Equation (2.1)) in terms of the input variables [18, 133].

2. *Coherence*: the general criterion behind the user choices of the functions ρ and c for the propagation rule. Definitions of ρ and c are not formally considered or argued in the literature, due to the empirical and heuristic nature of both the propagation rules addressed, and the problems related to the typical LRP applications. Then, trying to introduce a more detailed formalization, in this work a propagation rule is defined to be coherent if:

   - exists a measure or signed measure μ_ρ with respect to the Borel σ-algebra on $\mathbb{R}^2$ (here denoted as $\mathcal{B}(\mathbb{R}^2)$) such that $\rho(\xi_1, \xi_2) = \mu_\rho((0\,,\,\xi_1) \times (0\,,\,\xi_2))$, for each $(\xi_1, \xi_2) \in \mathbb{R}^2$;
   - exists a measure or signed measure μ_c with respect to the σ-algebra $\mathcal{B}(\mathbb{R}^2)$ such that $c(\xi_1, \xi_2) = \mu_c((0\,,\,\xi_1) \times (0\,,\,\xi_2))$, for each $(\xi_1, \xi_2) \in \mathbb{R}^2$.

   The coherence property, introduced by the authors of this work, has the purpose of providing an outline on how to build a well-defined propagation rule for LRP to highlight the most relevant input features. Indeed, not every pair of arbitrary functions ρ and c return scores that describe correctly the relevance of the inputs to the outputs, even if the conservation property is guaranteed. Defining ρ and c as functions characterized by the measures $\mu_\rho\,, \mu_c$, respectively, the authors argue that the relevance scores obtained with LRP better characterize the relationship between input and outputs in the NN.

Once a propagation rule is defined, the total starting score $R^{(L)}$ propagates from the output layer to the input units $u_i \in U^{(0)}$, and values $R_i^{(0)}$ let the user understand the relevance of the components of the given input $\boldsymbol{x}$ behind the FNN prediction $\widehat{\boldsymbol{F}}(\boldsymbol{x})$.

The propagation rule, defined by the two properties of conservation and coherence, is not uniquely identified. Therefore, different propagation rules have been proposed in literature [133]. In this work, the rule applied in Section 2.3 belongs to the class of α-β rule. For the ease of notation, denote as z_{ij} the product

$$z_{ij} = x_i w_{ij}\,, \tag{2.7}$$

where, for simplicity, the layer dependencies previously denoted with superscripts like (ℓ) and $(\ell+1)$ have been dropped. Then, α-β rule defines the quantity

$$z_{ij}^{\pm} := \max(\pm\, z_{ij}\,,\, 0) \tag{2.8}$$

as the signed local contribution of u_i with respect to u_j, and the quantity

$$z_j^{\pm} := b_j^{\pm} + \sum_{u_i \in U^{(\ell)}} z_{ij}^{\pm}\,, \tag{2.9}$$

as the signed pre-activation of u_j, where b_j is the bias of u_j and $b_j^{\pm} = \max(\pm\, b_j\,,\, 0)$.

Furthermore, the definition of the message from $u_j \in U^{(\ell+1)}$ to $u_i \in U^{(\ell)}$ in the α-β rule [133, section 5.1] is characterized by

$$\begin{aligned} R_{i\leftarrow j}^{(\ell,\ell+1)} = \rho\left(c_j(x_i\,, w_{ij})\,,\, R_j^{(\ell+1)}\right) = c_j(x_i\,, w_{ij}) \,\cdot\, R_j^{(\ell+1)} = \\ \left(\alpha\,\frac{z_{ij}^{+}}{z_j^{+}} \;-\; \beta\,\frac{z_{ij}^{-}}{z_j^{-}}\right) R_j^{(\ell+1)}\,, \end{aligned} \tag{2.10}$$

where, for each fixed $\alpha\,, \beta \in \mathbb{R}^{+}$, one should have $\alpha - \beta = 1$ in order to satisfy the conservation property. To avoid numerical instability, a small number (e.g. $\epsilon = 10^{-9}$) is added to the denominators of Equation (2.10). Observe also that the α-β rule is characterized by a coherence property, where μ_ρ and μ_{c_j} are such that:

$$\mu_\rho((0\,,\, \xi_1) \times (0\,,\, \xi_2)) = \xi_1\, \xi_2 \tag{2.11}$$

and

$$\mu_{c_j}((0, \xi_1) \times (0, \xi_2)) = \begin{cases} \frac{\alpha}{z_j^{+}}\, \xi_1\, \xi_2\,, & \text{if } \; \xi_1\, \xi_2 \geq 0 \\ -\frac{\beta}{z_j^{-}}\, \xi_1\, \xi_2\,, & \text{if } \; \xi_1\, \xi_2 < 0 \end{cases}. \tag{2.12}$$

Specifically, in Section 2.3 of this work, the parameter values are fixed to $\alpha = 1$ and $\beta = 0$. This setting implies that the message is defined as

$$R_{i\leftarrow j}^{(\ell\,,\,\ell+1)} = \begin{cases} (x_i w_{ij}/z_j^{+}) R_j^{(\ell+1)}\,, & \text{if } x_i w_{ij} \geq 0 \\ 0\,, & \text{if } x_i w_{ij} < 0 \end{cases}. \tag{2.13}$$

2.1.3 Expected Relevance Score

The LRP method has been widely used in applications for explanation of NNs concerning images, but rarely for explanation of regression NNs (e.g. [18, 37]). Probably due to this reason, to the best of the authors' knowledge, the usage of LRP as model interpretation method proposed in this paper has never been considered before.

The main idea behind model interpretation performed using LRP is the following: compute the *expected relevance scores*

$$\mathbb{E}_{\boldsymbol{x}\sim q}[R_1^{(0)}]\,,\ldots,\mathbb{E}_{\boldsymbol{x}\sim q}[R_n^{(0)}] \tag{2.14}$$

for the components $x_1\,,\ldots,x_n$, respectively, of a random input vector $\boldsymbol{x}\in\mathbb{R}^n$ with distribution q, with respect to a given FNN $\mathcal{N}$ with function $\widehat{\boldsymbol{F}}:\mathbb{R}^n\to\mathbb{R}^m$. Then, the most important features for $\widehat{\boldsymbol{F}}$ are the ones characterized by a higher expected relevance score. Furthermore, assuming that $\widehat{\boldsymbol{F}}(\boldsymbol{x})\approx\boldsymbol{F}(\boldsymbol{x})$ for each $\boldsymbol{x}\in A\subseteq\mathbb{R}^n$, LRP allows performing indirectly (through $\mathcal{N}$) a feature selection also with respect to the target function $\boldsymbol{F}:A\subseteq\mathbb{R}^n\to\mathbb{R}^m$.

From a practical point of view, the vector of expected relevance scores is approximated by

$$\begin{gathered}\bar{\boldsymbol{r}}=\bar{\boldsymbol{r}}^{(0)}:=\mathbb{E}_{\boldsymbol{x}\sim q}[\boldsymbol{r}^{(0)}]=\mathbb{E}_{\boldsymbol{x}\sim q}\left[\left[R_1^{(0)}\,,\ldots,R_n^{(0)}\right]^\top\right]=\\ \left[\mathbb{E}_{\boldsymbol{x}\sim q}[R_1^{(0)}]\,,\ldots,\mathbb{E}_{\boldsymbol{x}\sim q}[R_n^{(0)}]\right]^\top\end{gathered} \tag{2.15}$$

computing the vector of mean relevance scores with respect to a given set $\mathcal{S}\subset\mathbb{R}^n$ of s samplings of $\boldsymbol{x}$, i.e.:

$$\bar{\boldsymbol{r}}(\mathcal{S})=\bar{\boldsymbol{r}}^{(0)}(\mathcal{S}):=\mathbb{E}_{\mathcal{S}}[\boldsymbol{r}^{(0)}]=\frac{1}{s}\sum_{\boldsymbol{x}\in\mathcal{S}}\boldsymbol{r}^{(0)}=\frac{1}{s}\sum_{\boldsymbol{x}\in\mathcal{S}}\left[R_1^{(0)}\,,\ldots,R_n^{(0)}\right]^\top\approx\bar{\boldsymbol{r}}\,, \tag{2.16}$$

where $\bar{\boldsymbol{r}}(\mathcal{S})=\bar{\boldsymbol{r}}$ for $s=|\mathcal{S}|\to+\infty$.

2.2 Case study: Discrete Fracture Network

Discrete Fracture Networks (DFNs) [6, 42, 70] are popular models adopted for performing flow simulations in underground fractured media. Each fracture of the DFN network is represented by a 2-dimensional polygon into a 3-dimensional domain, and it is characterized by its own geometrical and hydro-geological features (namely: position, size, orientation, fracture transmissivity, ...). In this section, a new strategy is discussed. The strategy relies on flux-regression Neural Networks trained on datasets generated via DFN flow simulations, as in [35] and Layer-wise Relevance Propagation. The aim is to identify backbones of the DFN, namely, suitable subnetworks of the DFN where the transport characteristics approximate the ones of the original network [178]. Further details about the Backbone Identification problem are given in Subsection 2.2.2.

While most applications of LRP concern NNs trained on image datasets for classification tasks (e.g. [18, 37]), this experiment applies LRP to a NN trained on physical simulation data for a regression task. Furthermore, LRP usage is here characterized by an innovative way of application; indeed, LRP is not run on input

data one by one, looking for the most relevant features for each single prediction of the NNs, but an approximation of the expected relevance scores of all the features in the domain space is computed. In this way, LRP is presented as a feature selection method with respect to the numerical model of the DFN, and the backbone fractures of the DFN are identified as the ones with higher expected relevance score. Finally, the validation of the effectiveness of the LRP-based feature selection method consist of checking the quality of the identified backbone, running the DFN simulations on the corresponding subnetwork.

The method proposed herein can be very useful in those applications concerning the backbone of a DFN and the flux behaviour, varying the fracture transmissivities; in particular, the backbone obtained with this method can be extremely effective for clogging problems and waste storage problems. Indeed, since the backbones returned by the proposed method are identified in the framework of UQ, they are statistically robust with respect to changes in fracture transmissivities. For example, a user is able to statistically know which fractures are more relevant for flux and should be avoided for waste storage problems, or which fractures are most critical in flux propagation and can be important in clogging problems.

2.2.1 Numerical Model

Here, for the reader's convenience, the DFN modeling problem is briefly described. For full details, the interested reader could refer to [27, 28].

A DFN is a discrete model used to describe and characterize a network of underground fractures in a fractured rock medium as a set of 2D polygons in the 3D space $\mathbb{R}^3$ (see Figure 2.1). Each polygon represents a fracture, and it is labelled with an index in a set I; then, each fracture is denoted by $\mathcal{F}_i$, with $i \in I$. A DFN is composed by the union of all the fractures:

$$\cup_{i \in I} \mathcal{F}_i.$$

Each fracture is endowed with its own size and orientation in the 3D space and with its own transmissivity parameter κ_i for the flux characterization; all these data are typically sampled from suitable distributions. Segments given by the intersection of two or more fractures are called *traces*, and they characterize the connectivity of the network; notably, DFNs can be represented as graphs with fractures as nodes and traces as edges (see Figure 2.2).

The flow simulations in a DFN are characterized not only by its geometry, but also by the hydrogeological properties conferred to the fractures, such as the transmissivity. The transmissivity parameter κ_i of $\mathcal{F}_i$, for each $i \in I$, represents the flow facilitation through the fracture, and it is fundamental for the flow characterization in the DFN; the Section 2.2.3 is focused on the NN regression problem for predicting the outflowing fluxes of a DFN given the transmissivities of its fractures.

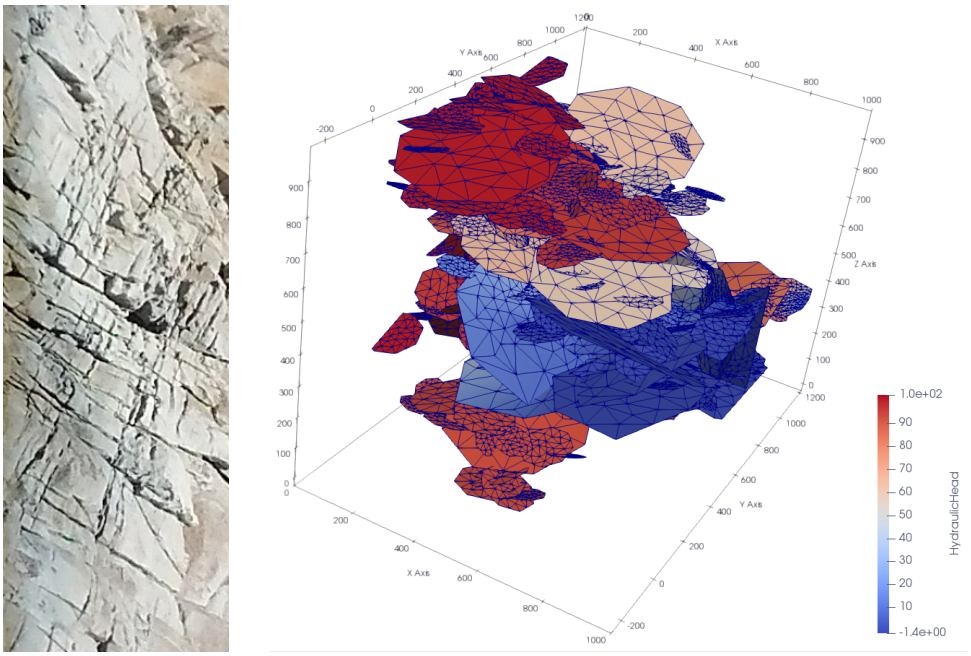

Figure 2.1: External surface of a natural fractured medium (left) and a DFN (right). As an example, the opening in the rock could be the upper side of the right 3D plot. Each colored meshed polygon on the right is a fracture, and polygon intersections are traces.

2.2.2 The Backbone Identification problem

Given a modelled DFN, a backbone of the DFN is a suitable subnetwork where the transport characteristics approximate the ones of the original network [178]. These subnetworks can be used in many applications and furnish precious and fundamental information for clogging problems and for waste storage problems simulated through DFNs (e.g. geological storage of CO_2). In [9], backbones are identified through particle tracking methods that find the fractures where most of the transport of particles occurs. However, the computational domain characterizing a DFN can be quite large and exhibit a great deal of geometrical complexity, therefore transport and flow simulations turn out to be very costly, even if recent literature has proposed several approaches to overcome these problems. To mention a few, recall papers based on reformulations as lower-dimensional problems [146, 145, 55]; based on the use of partially conforming meshes [153, 152, 63, 154]; other interesting geometrical approaches are proposed in [71, 72, 76, 109, 111]; approaches consisting in a reformulation of the problem as a PDE-constrained optimization problem [27, 28, 29, 25, 31, 24], that can be used in conjunction with several space discretizations [30, 23]. Nonetheless, computational simulation on a large DFN is still a costly task, and it may be prohibitive to perform numerous

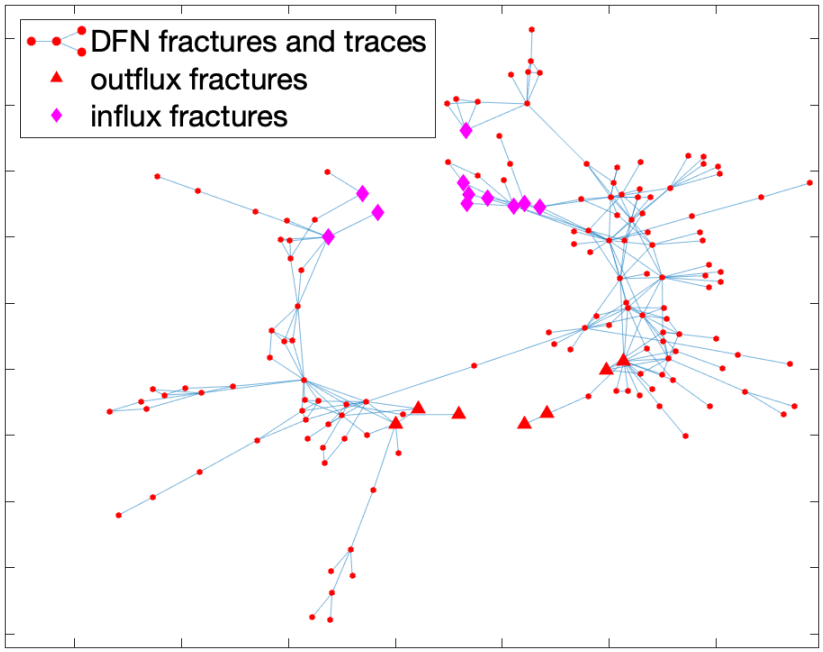

Figure 2.2: The graph representation of a DFN: nodes depict the fractures, traces are the edges.

simulations.

Due to the expensive cost of particle tracking simulations, other backbone identification methods based on graph topology and Machine Learning (ML) have been developed [178, 107, 106, 179]. These methods usually train the learning algorithms as binary classifiers for single fractures, predicting whether they are backbone fractures or not, on datasets built using particle flow simulations. The great advantage of these methods is that, given a sufficiently large dataset of classified fractures for the training, the backbone identification process is fast and accurate. That is extremely useful in the framework of Uncertainty Quantification (UQ), where numerous simulations are required. Indeed, one of the main issues related to DFNs is the lack of deterministic information about geometrical and hydro-geological fracture features. This information is typically only known by means of probability distribution, and data needed for actual simulations are commonly sampled from the available distributions.

All the cited backbone identification methods take into account the time spent by particles to flow from the source to the sink of the network as criterion of the identified backbone. The time spent is assumed to be such that the first passage time of the particles through the backbone-reduced subnetwork of the DFN is approximately equal to the one of the full DFN. However, in some problems the quantity of interest (QoI) may be, for example, the total flux exiting the DFN. This case study focus on the total flux outflowing the network for identifying the

backbone. Then, for a given DFN, the target is to identify a backbone such that its exiting flux approximates the one of the full DFN. The method is tested and applied in the framework of a DFN with fixed geometry but with stochastic fracture transmissivities. In particular, the method illustrated is able to identify a backbone of the given DFN sufficiently good (with respect to the approximation of the QoI) for every possible sample of fracture transmissivities. The experiments showing these results are provided in Section 2.3

2.2.3 Regression Problem Setting

The problems addressed in this paper are characterized as follows. See Figure 2.3 and 2.4 for the two examples taken from [35] used to generate the experimental datasets. In the following, the details and assumptions of the data generation are resumed. Consider a DFN consisting of n fractures, with fixed geometrical prop-

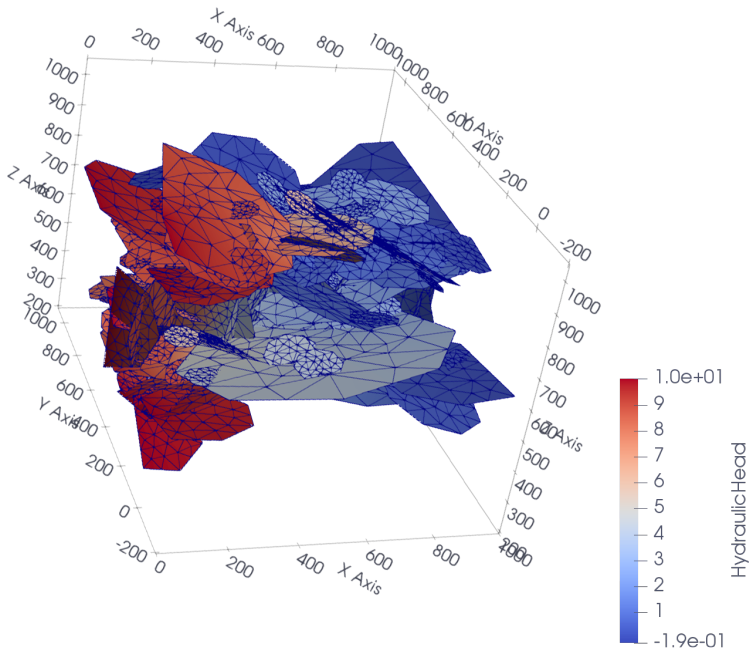

Figure 2.3: 3D view of DFN158.

erties, immersed in a cubic matrix block with a 1 000 meters long edge. Assume the boundary conditions to be such that two opposite faces of the block represent an inlet and outlet face, respectively. Impose a Dirichlet boundary condition on the pressure H fixed at $H = 10$ on fracture edges created intersecting the DFN with the leftmost face of the domain, corresponding to $x = 0$, and $H = 0$ on the edges obtained intersecting the DFN with the rightmost face, that is, $x = 1\,000$. The fractures intersecting such faces are called inflow and outflow fractures, respectively. All other fracture edges are insulated with homogeneous Neumann condition. The

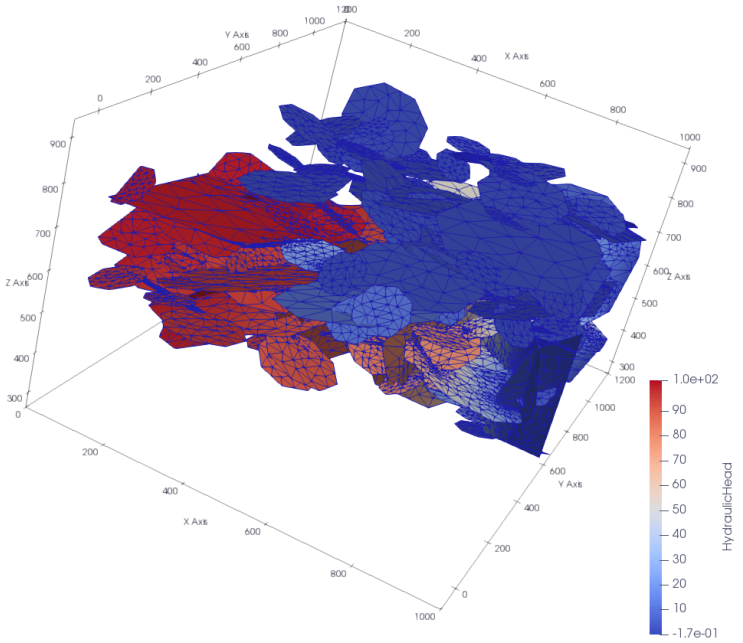

Figure 2.4: 3D view of DFN202.

boundary conditions and the geometry of the DFN mainly affect the flux directionality, while the transmissivities have a great impact on the flow intensity on each outflow fracture. The target of the regression problem here considered is to predict the exiting fluxes of each sample and to approximate the exiting flux distributions among the outflow fractures. For a complete description of the DFN model, refers to [35].

The fracture transmissivities are assumed to be isotropic parameters $\kappa_1, \ldots, \kappa_n$ modelled as random variables with log-normal distribution [106, 165]:

$$\log_{10} \kappa_i \sim \mathcal{N}(-5,1/3)\,. \tag{2.17}$$

In particular, the two test cases considered in the experiments are characterized by $n = 158$ and $n = 202$ fractures named DFN158 and DFN202, respectively (see also [35]). The fractures are assumed to be octagons, and have been randomly generated using the following distribution for the geometrical features [185, 105]: fracture radii have been sampled with respect to a truncated power law distribution, with exponent $\gamma = 2.5$ and upper and lower cut-off $r_u = 560$ and $r_0 = 50$, respectively; the fracture orientations have been sampled from a Fischer distribution having mean direction $\boldsymbol{\mu} = (0.0065, -0.0162, 0.9998)$ and dispersion parameter 17.8; uniform distribution has been used for mass centers. The resulting number of outflow fractures for DFN158 and DFN202 is $m = 7$ and $m = 14$, respectively.

Dataset characterization

Fix the following notation. Let $\boldsymbol{\kappa} = [\kappa_1, \dots, \kappa_n]^\top \in \mathbb{R}^n$ be the vector collecting the transmissivities of all fractures of the given DFN and let $\boldsymbol{\varphi} = [\varphi_1, \dots, \varphi_m]^\top \in \mathbb{R}^m$ be the vector collecting all the exit flows. Let $\boldsymbol{F} : A \subseteq \mathbb{R}^n \to \mathbb{R}^m$ be a function defined by

$$\boldsymbol{\varphi} = \boldsymbol{F}(\boldsymbol{\kappa}), \tag{2.18}$$

that is the function that provides the vector of outflows associated to the transmissivity input $\boldsymbol{\kappa}$. Consider $D \in \mathbb{N}$ samples $\boldsymbol{\kappa}_k \in \mathbb{R}^n$, $k = 1, \dots, D$, drawn according to distribution (2.17). The dataset $\mathcal{D}$[1] used for the creation of the training set, the test set and the validation set is

$$\mathcal{D} = \{(\boldsymbol{\kappa}_k, \boldsymbol{\varphi}_k) \in \mathbb{R}^n \times \mathbb{R}^m \mid \boldsymbol{F}(\boldsymbol{\kappa}_k) = \boldsymbol{\varphi}_k \,,\ \forall\ k = 1, \dots, D\} \,. \tag{2.19}$$

The test set $\mathcal{P}$ is created by randomly picking approximately 30% of the elements in $\mathcal{D}$. The remaining elements are then randomly split into two subsets $\mathcal{T}$ and $\mathcal{V}$, representing the training set and the validation set, respectively, and such that $|\mathcal{V}| \sim 20\% \, |\mathcal{D} \setminus \mathcal{P}|$.

2.3 Experiments

This section shows the experimental setting and the results obtained by employing the Layer-Wise Relevance Propagation algorithm to identify the Backbone on the Discrete Fracture Network model described in Section 2.2.

2.3.1 Multi-Task Architecture

This section recall the architecture introduced in [35] for the approximation of a function $\boldsymbol{F} : A \subseteq \mathbb{R}^n \to \mathbb{R}^m$. These NNs are characterized by a tree-shaped structure (see Figure 2.5), obtained by extending the one described in [86, chapter 7.7] for multi-task learning. In particular, given a hyperparameter $d \in \mathbb{N}$, the NN architecture is given by:

- One input layer $U^{(0)}$ of n units;
- A sequence of d hidden layers $U^{(1)}, \dots, U^{(d)}$, each one made of n units with softplus ($f(x) = \log(1 + e^x)$) activation function, such that $U^{(\ell-1)}$ is fully connected to $U^{(\ell)}$, for each $\ell = 1, \dots, d$. Call *trunk* of the NN the sequence $U^{(0)}, \dots, U^{(d)}$;

[1] The datasets of DFN158 and DFN202 can be downloaded from https://smartdata.polito.it/discrete-fracture-network-flow-simulations/ (transmissivity st.dev. parameter 0.33)

- m sequences of d hidden layers $U_j^{(d+1)}, \dots, U_j^{(2d)}$, each one made of n softplus units, followed by one output layer $U_j^{(2d+1)}$ made of one linear unit for each $j = 1 \dots, m$. These layers are such that $U^{(d)}$ is fully connected to $U_j^{(d+1)}$ and $U_j^{(\ell-1)}$ is fully connected to $U_j^{(\ell)}$, for each $\ell = d+2, \dots, 2d+1$, for each $j = 1, \dots, m$. Call *branches* of the NN all the m sequences $U_j^{(d+1)}, \dots, U_j^{(2d+1)}$.

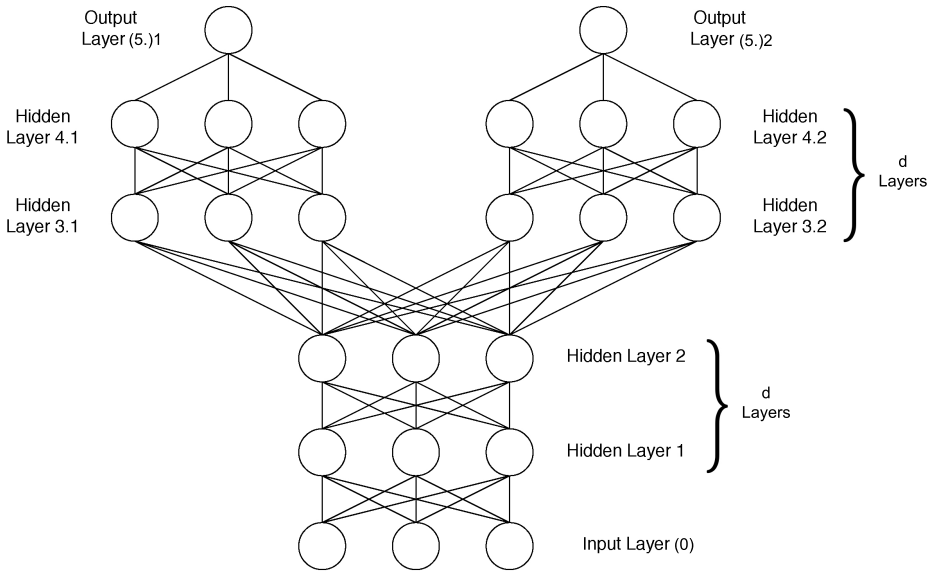

Figure 2.5: Example of NN built for vector valued regression concerning flux prediction ($n = 3$, $m = 2$, $d = 2$). For simplicity, biases have not been represented.

The choice of using softplus functions for the hidden layers was made after a preliminary investigation in [35], comparing the performances obtained also with other activation functions.

Consider a NN $\mathcal{N}_n^* \in \{\mathcal{N}_{158}^*, \mathcal{N}_{202}^*\}$ (see Subsection 2.3.2) and a general input vector $\widetilde{\boldsymbol{\kappa}}$ of (normalized) transmissivities for $\mathcal{N}_n^*$. Then, since $\widehat{\boldsymbol{F}}(\widetilde{\boldsymbol{\kappa}}) \approx \boldsymbol{F}(\boldsymbol{\kappa})$ is a valuable approximation (Tables 2.1-2.3), the application of LRP method to $\mathcal{N}_n^*$ with respect to a given $\widetilde{\boldsymbol{\kappa}}$ returns a vector of relevance scores

$$\boldsymbol{r}^{(0)} = \left[R_1^{(0)}, \dots, R_n^{(0)}\right]^\top$$

that is a vector characterizing the relevance of fractures $\mathcal{F}_i$ in the DFN during the computation of the fluxes $\boldsymbol{\varphi} = \boldsymbol{F}(\boldsymbol{\kappa})$, for each $i = 1, \dots, n$. In particular, due to the conservation property of LRP (see the Section 2.1.2), observe that initialization of the scores as in Equation (2.1) allows the relevance scores of the most relevant fractures $\mathcal{F}_i$ to increase when the sum of the predicted fluxes $\sum_{j=1}^m \widehat{\varphi}_j = R(\widetilde{\boldsymbol{\kappa}})$ is higher.

Given the interpretation of LRP relevance scores as fracture relevance in the DFN for the single simulation $\boldsymbol{F}(\boldsymbol{\kappa})$, the idea can be extended. Using LRP as feature selection method with respect to $\mathcal{N}_n^*$ (see Subsection 2.1.3) and assuming $\widehat{\boldsymbol{F}}(\widetilde{\boldsymbol{\kappa}}) \approx \boldsymbol{F}(\boldsymbol{\kappa})$ for each transmissivity vector $\boldsymbol{\kappa} \in \mathbb{R}^n$, the vector of expected relevance scores $\bar{\boldsymbol{r}} \in \mathbb{R}^n$ could be interpreted as a measure of the expected relevance of the fractures $\mathcal{F}_1, \dots, \mathcal{F}_n$ in the DFN, for any random $\boldsymbol{\kappa}$ sampled.

As a consequence, if this interpretation of LRP relevance scores is correct, a collection of the most relevant fractures (e.g., the ones with $\mathbb{E}_{\boldsymbol{\kappa}\sim q_\kappa}[R_i^{(0)}]$ greater than an arbitrary threshold) can be interpreted as a possible backbone of the DFN, where the target Quantity of Interest (QoI) to be preserved is the total flux exiting from the DFN. For this reason, this section analyzes the fluxes obtained running simulations on subnetworks of both DFN158 and DFN202, obtained selecting the fractures through LRP applied on $\mathcal{N}_{158}^*$ and $\mathcal{N}_{202}^*$, respectively, and comparing them with full simulations on the whole DFNs. From these comparisons, it results a validation of both the LRP-based feature selection's quality (see Subsection 2.3.4) and the subnetworks as backbones.

2.3.2 Neural Network Training and Performances

Consider the two test cases described in the previous Subsection 2.2.3 (DFN158 and DFN202). Recall that they are respectively characterized by $n = 158$ and $n = 202$ total fractures, and $m = 7$ and $m = 14$ outflow fractures.

The multi-task architecture previously described is used to design and train suitable NNs for each DFN. The training of NNs is conducted varying hyperparameters already tested in [35]. The configuration of hyperparameters yield four different NNs for each DFN:

- *depth parameter* $d \in \{1,3\}$ (the depth of the NN being equal to $2d$);
- *mini-batch size* $B \in \{10,30\}$;
- a number n of units for the input layer and hidden layers coinciding with the number of fractures;
- the tree-shaped structure has $m = 7$ branches for DFN158 and $m = 14$ branches for DFN202.

In the following, refer to these NNs and options as

$$\mathcal{N}_{n,d}^B, \quad \forall\, d \in \{1,3\}, \quad \forall\, B \in \{10,30\}, \tag{2.20}$$

All the NNs are trained and tested with respect to a dataset $\mathcal{D}_n$ (see Section 2.2.3) of 10 000 pairs $(\boldsymbol{\kappa}_k, \boldsymbol{\varphi}_k) \in \mathbb{R}^n \times \mathbb{R}^m$, in order to make predictions of the outflowing fluxes of DFN158 and DFN202. The training is made using the optimizer `Adam` [114], with a maximum number of epochs $c_{\max} = 1\,000$, mini-batch size B and two

regularization methods: early stopping method, with patience parameter $p^* = 150$, and method of minimum validation error. Then, for each fixed $n = 158{,}202$, the two networks $\mathcal{N}_n^*$ with the best performances are selected. The selection takes into account a grid search approach with respect to the values of d and B, and it uses as performance measure the mean value of the global relative errors of the predictions of $\mathcal{N}_{n,d}^B$ on the test set $\mathcal{P}_n$ (see Table 2.1). For simplicity, this quantity is referred to as $\mathbb{E}[e^r(\mathcal{N}_{n,d}^B\,;\,\mathcal{P}_n)]$, defining the vector of relative errors of a prediction $\widehat{\boldsymbol{\varphi}}$ with respect to the total exiting flux [35] as

$$\boldsymbol{e}^r(\widehat{\boldsymbol{\varphi}}) = \frac{|\widehat{\boldsymbol{\varphi}} - \boldsymbol{\varphi}|}{\sum_{i=1}^m \varphi_i} = \frac{1}{\sum_{i=1}^m \varphi_i}\left[\, |\widehat{\varphi}_1 - \varphi_1|\,, \ldots, |\widehat{\varphi}_m - \varphi_m|\,\right]^\top . \tag{2.21}$$

The results of the selection over the grid search are $\mathcal{N}_{158}^* := \mathcal{N}_{158,3}^{30}$ and $\mathcal{N}_{202}^* := \mathcal{N}_{202,3}^{30}$.

	DFN158		DFN202	
	$d = 1$	$d = 3$	$d = 1$	$d = 3$
$B = 10$	0.0104	0.0085	0.0060	0.0070
$B = 30$	0.0099	**0.0082**	0.0057	**0.0055**

Table 2.1: Mean relative errors $\mathbb{E}[e^r(\mathcal{N}_{n,d}^B\,;\,\mathcal{P}_n)]$ for several values of depth parameter d and mini-batch size B for all the test cases ($n = 158, 202$).

The approach here adopted for building and training the NNs is the same used in [35], but with a main difference: a pre-processing phase for the input data has been introduced. Indeed, NN performances often increase when a normalization of input data is applied to have zero mean and standard deviation equal to 1 [141]. Hence, the preprocessing consist of the application of a function $g : \mathbb{R} \to \mathbb{R}$ to standardize $\log(\kappa_i)$:

$$g(\kappa_i) = \widetilde{\kappa}_i \sim \mathcal{N}(0{,}1)\,,\ \forall\ i = 1, \ldots, n \tag{2.22}$$

and the training of the NNs is conducted as in [35] with respect to a normalized version of $\mathcal{D}_n$, that is the dataset $\widetilde{\mathcal{D}}_n$ characterized by normalized inputs such that

$$\widetilde{\mathcal{D}}_n = \{(\widetilde{\boldsymbol{\kappa}}_k\,,\,\boldsymbol{\varphi}_k) \in \mathbb{R}^n \times \mathbb{R}^m \mid (\boldsymbol{\kappa}_k, \boldsymbol{\varphi}_k) \in \mathcal{D}_n\}\ , \tag{2.23}$$

where $\widetilde{\boldsymbol{\kappa}}_k = \boldsymbol{g}(\boldsymbol{\kappa}_k)$ and $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^n$ is the element-wise application of the function g to the components of $\boldsymbol{\kappa}_k$.

Tables 2.2 and 2.3 report the measurements of the Jensen-Shannon divergence (D_{JS}) for the actual flux distributions and the predicted flux distributions given by the NNs $\mathcal{N}_n^*$ with respect to the inputs of the test set $\widetilde{\mathcal{P}}_n$ (see Figure 2.6 for visualizing an example of distribution comparison). They report an additional relative

dissimilarity measure for distributions, the ratio $\mathcal{D}_{\mathrm{KL}}/\mathcal{E}$, for the ease of interpretation and comparison of the values. Indeed, the D_{JS} values can be considered low for some pairs of distributions, but not for others. The new dissimilarity measure introduced is defined as the ratio between the Kullback-Liebler divergence (D_{KL}) and the entropy ($\mathcal{E}$) of the actual distribution (see [35] and [86, chapter 3.13]), namely:

$$\frac{D_{\mathrm{KL}}(P \parallel Q)}{\mathcal{E}(P)} = \frac{\mathbb{E}_{x\sim P}\left[\log\left(P(x)/Q(x)\right)\right]}{\mathbb{E}_{x\sim P}\left[\log P(x)\right]}\,, \tag{2.24}$$

where P is the actual flux's probability distribution of a fracture and Q is the one of corresponding predictions. The advantages of using ratio (2.24) derive from the relationship between the D_{KL} and the cross-entropy $\mathcal{E}(P,Q)$ [86, chapter 3.13], since:

$$\frac{D_{\mathrm{KL}}(P \parallel Q)}{\mathcal{E}(P)} = \frac{\mathcal{E}(P,Q) - \mathcal{E}(P)}{\mathcal{E}(P)} = \frac{\mathcal{E}(P,Q)}{\mathcal{E}(P)} - 1\,, \tag{2.25}$$

where

$$\mathcal{E}(P,Q) := \mathbb{E}_{x\sim P}\left[\log Q(x)\right]\,. \tag{2.26}$$

In fact, $\mathcal{E}(P,Q)$ measures the average information needed to describe the entropy $\mathcal{E}(P)$ (that is, the average information rate of P) using a random sampling from Q. Then the ratio (2.24) represents the relative information error, measuring the relative missing information lost to describe the average information rate of P using a random sampling from Q.

	$\mathcal{F}_8$	$\mathcal{F}_{12}$	$\mathcal{F}_{14}$	$\mathcal{F}_{78}$	$\mathcal{F}_{90}$	$\mathcal{F}_{98}$	$\mathcal{F}_{107}$
D_{JS}	0.0050	0.0018	0.0063	0.0012	0.0196	0.2144	0.0060
$D_{\mathrm{KL}}/\mathcal{E}$	0.0009	0.0003	0.0010	0.0002	0.0033	0.0379	0.0010

Table 2.2: DFN158. Jensen-Shannon divergence and dissimilarity measure between actual and predicted flux distributions.

2.3.3 LRP Generalization to Multitask Neural Networks

In this Subsection, the LRP method is generalized to FNNs having the multitask architecture described in Subsection 2.3.1. In principle, the application of LRP to multitask FNNs can be easily performed, since any multitask FNN can be translated into an equivalent non-multitask FNN characterized by some null weights. However, this sort of translation is not advisable for at least two reasons: firstly, it violates the principle of parsimony in terms of coding and computational load; secondly, one hinders the model interpretability by losing its tights with the topology of

	$\mathcal{F}_8$	$\mathcal{F}_{15}$	$\mathcal{F}_{18}$	$\mathcal{F}_{31}$	$\mathcal{F}_{61}$	$\mathcal{F}_{73}$	$\mathcal{F}_{93}$
D_{JS}	0.0050	0.0941	0.0226	0.0033	0.0021	0.0014	0.0324
$D_{\mathrm{KL}}/\mathcal{E}$	0.0007	0.0177	0.0040	0.0005	0.0003	0.0003	0.0055
	$\mathcal{F}_{115}$	$\mathcal{F}_{156}$	$\mathcal{F}_{162}$	$\mathcal{F}_{173}$	$\mathcal{F}_{176}$	$\mathcal{F}_{180}$	$\mathcal{F}_{187}$
D_{JS}	0.0256	0.0928	0.0059	0.0016	0.0327	0.0069	0.0570
$D_{\mathrm{KL}}/\mathcal{E}$	0.0042	0.0165	0.0014	0.0002	0.0039	0.0016	0.0097

Table 2.3: DFN202. Jensen-Shannon divergence and dissimilarity measure between actual and predicted flux distributions.

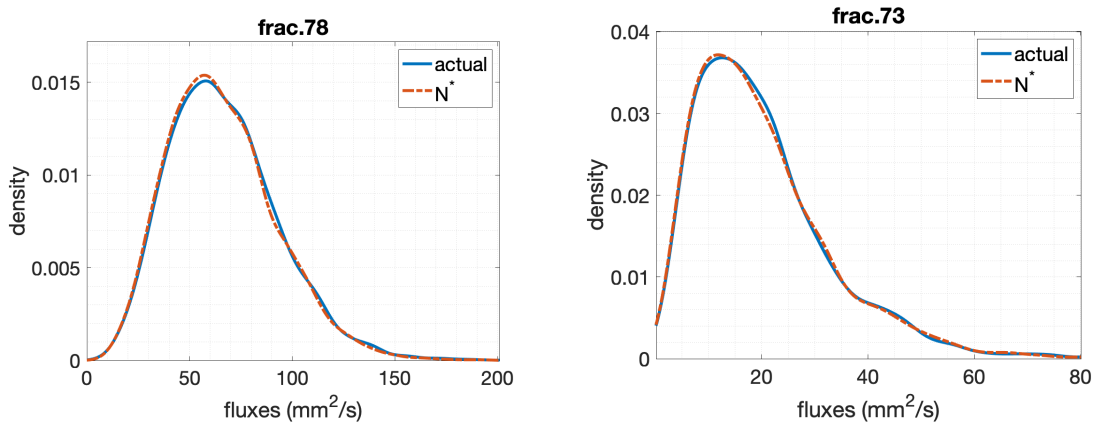

Figure 2.6: DFN158 case (left) and DFN202 case (right). Example of two comparisons between probability density functions of the actual flux distribution (continuous line) and the predicted flux distribution (dotted line) for one of the outflowing fractures, done by $\mathcal{N}_n^*$.

the underlying DFN. It is, therefore, worthwhile generalizing the LRP method to multi-task architectures.

Consider a multi-task FNN $\mathcal{N}$ with the same architecture as the one introduced in Subsection 2.3.1, characterized by n inputs, m branches and outputs, and depth $2d$. Introduce the following notation for the scores involving the layers of the branches.

- Let L be such that $L = 2d + 1$. For each $h = 1, \dots, m$, the starting score assigned to the h-th output is denoted as

$$R_1^{(L\,;\,h)} = \widehat{\varphi}_h = \left(\widehat{\boldsymbol{F}}(\widetilde{\boldsymbol{\kappa}})\right)_h , \tag{2.27}$$

where $\widehat{\boldsymbol{F}}$ is the function associated to $\mathcal{N}$ and $(\widetilde{\boldsymbol{\kappa}}\,,\,\widehat{\boldsymbol{\varphi}})$ is a given input-output pair defined as in Subsection 2.3.2. Then, the total starting score $R(\widetilde{\boldsymbol{\kappa}}) =$

$\sum_{h=1}^{m} \widehat{\varphi}_h$ is re-defined as

$$R(\widetilde{\boldsymbol{\kappa}}) = \sum_{h=1}^{m} R_1^{(L\,;h)} =: R^{(L)}\,. \tag{2.28}$$

- For each $\ell = d+1, \ldots, 2d$, the scores computed with respect to units of the layer $U_h^{(\ell)}$ (belonging to the h-th branch of $\mathcal{N}$) are denoted as $R_{i\leftarrow j}^{(\ell,\ell+1\,;h)}$. Then, analogously to (2.3) and (2.6), $R_i^{(\ell\,;h)}$, $R^{(\ell\,;h)}$ and $R^{(\ell)}$ are defined, respectively, as

$$R_i^{(\ell\,;h)} = \sum_{u_j \in U_h^{(\ell+1)}} R_{i\leftarrow j}^{(\ell,\ell+1\,;h)}\,, \tag{2.29}$$

$$R^{(\ell\,;h)} = \sum_{u_i \in U_h^{(\ell)}} R_i^{(\ell\,;h)} \tag{2.30}$$

and

$$R^{(\ell)} = \sum_{h=1}^{m} R^{(\ell\,;h)}\,. \tag{2.31}$$

In particular, the above equation is used also for the case $\ell = L = 2d+1$, even if the result is $R^{(L\,;h)} = R_1^{(L\,;h)}$.

- For $\ell = d$ and for each $h = 1, \ldots, m$, the message propagating scores from $u_j \in U_h^{(d+1)}$ to $u_i \in U^{(d)}$ is denoted as $R_{i\leftarrow j}^{(d,d+1\,;h)}$. Then, the total score $R_i^{(d)}$ propagated to u_i is given by

$$R_i^{(d)} = \sum_{h=1}^{m} \sum_{u_j \in U_h^{(d+1)}} R_{i\leftarrow j}^{(d,d+1\,;h)} \tag{2.32}$$

Generalizing now the α-β rule (2.10) to define the scores $R_{i\leftarrow j}^{(\ell,\ell+1\,;h)}$, for each $\ell = d, \ldots, L-1$ and for each $h = 1, \ldots, m$, observe that the conservation and coherence properties are satisfied for the multitask NN $\mathcal{N}$. Then, for the generality of parameters n, m, and d considered, the LRP method characterized by the α-β rule can be applied to multitask NNs characterized by the architecture described in section 2.3.1.

2.3.4 Validation of the Expected Relevance Scores

Since aim at using the expected relevance scores returned by LRP to select an arbitrary number $p \in \mathbb{N}$, $p \leq n$, of most relevant features in the domain of $\boldsymbol{F}$, the validation criterion introduced in this work is different from the ones usually adopted for LRP.

A common criterion used to evaluate an XAI algorithm used for outcome explanation, like LRP, is the one that here is named *excluding criterion*. This criterion

consists in analyzing how the predicted NN outcome changes if the p most important features of an input $\boldsymbol{x}$, identified by the XAI algorithm, are modified to an uninformative neutral value, obtaining an altered input $\boldsymbol{x}'$ [126, 171]. Then, given a second altered input $\boldsymbol{x}_{\text{rand}}$ obtained from $\boldsymbol{x}$ setting p random components to the neutral value, the excluding criterion analyzes how the predictions $\widehat{\boldsymbol{F}}(\boldsymbol{x}')$ and $\widehat{\boldsymbol{F}}(\boldsymbol{x}_{\text{rand}})$ change with respect to $\widehat{\boldsymbol{F}}(\boldsymbol{x})$: if the p most important input features of $\boldsymbol{x}$ have been properly identified by the algorithm, the distance between $\widehat{\boldsymbol{F}}(\boldsymbol{x}')$ and $\widehat{\boldsymbol{F}}(\boldsymbol{x})$ is significantly greater than the distance between $\widehat{\boldsymbol{F}}(\boldsymbol{x}_{\text{rand}})$ and $\widehat{\boldsymbol{F}}(\boldsymbol{x})$. In computer vision, where LRP has been mainly applied, each input feature is a pixel color and the neutral value is usually assumed to be the gray color.

However, in the DFN context the excluding criterion is not easily applicable due to the difficulty to define a suitable neutral value, and to the obstacles to extend the criterion from a local XAI method for outcome explanation to a global one for model interpretation, based on the expected relevance scores. Therefore, a novel validation criterion is proposed here, called *retaining criterion*. The retaining criterion involves the usage of the DFN simulator represented by the approximated function $\boldsymbol{F}\colon A \subseteq \mathbb{R}^n \to \mathbb{R}^m$. Let $\boldsymbol{F}_{|p} : A_{|p} \subseteq \mathbb{R}^p \to \mathbb{R}^m$ be a restriction of $\boldsymbol{F}$ with respect to the p most relevant features of a given input $\boldsymbol{x}$, identified by the XAI algorithm; analogously, let $\boldsymbol{x}_{|p} \in A_{|p}$ be the restriction of $\boldsymbol{x} \in A \subseteq \mathbb{R}^n$. Then, the criterion consists in comparing $\boldsymbol{F}_{|p}(\boldsymbol{x}_{|p})$ and $\boldsymbol{F}(\boldsymbol{x})$: if $\boldsymbol{F}_{|p}(\boldsymbol{x}_{|p}) \approx \boldsymbol{F}(\boldsymbol{x})$, the algorithm has correctly identified the p most relevant features of $\boldsymbol{x}$.

The retaining criterion can also be extended easily to a global XAI algorithm like the one defined in Section 2.1.3. In this case, the p most relevant features are the p ones with highest expected relevance score and, therefore, are fixed for each input $\boldsymbol{x} \in \mathbb{R}^n$. Then, the expected relevance scores returned by LRP perform a good feature selection of p features if $\boldsymbol{F}_{|p}(\boldsymbol{x}_{|p}) \approx \boldsymbol{F}(\boldsymbol{x})$ for each $\boldsymbol{x} \in A \subseteq \mathbb{R}^n$.

In general, observe that the new retaining criterion is characterized by the following advantages over the excluding criterion:

- absence of a neutral value usage and definition;
- easily extendable to a global XAI algorithm for model interpretation evaluation;
- the possibility to compare the effects of different choices of p directly looking at actual simulator outputs instead of model predictions.

Then, the retaining criterion is adopted for evaluating the quality of the backbones identified with the new method described in the next section 2.3.5.

2.3.5 The Direct Method for Backbone Identification

Consider the NNs $\mathcal{N}^*_{158}, \mathcal{N}^*_{202}$ and the corresponding datasets $\widetilde{\mathcal{D}}_{158}\,, \widetilde{\mathcal{D}}_{202}$ (see Equation (2.23)), each one with cardinality $|\widetilde{\mathcal{D}}_{158}| = |\widetilde{\mathcal{D}}_{202}| = 10\,000$ (see Subsection

2.3.2). For each $\mathcal{N}_n^* \in \{\mathcal{N}_{158}^*, \mathcal{N}_{202}^*\}$ compute the vector of mean relevance scores with respect to the set $\widetilde{\mathcal{D}}_n$, i.e. the vector

$$\bar{\boldsymbol{r}}_n \;:=\; \bar{\boldsymbol{r}}\,(\widetilde{\mathcal{D}}_n) \;=\; \mathbb{E}_{\widetilde{\mathcal{D}}_n}[\boldsymbol{r}^{(0)}] \;=\; \frac{1}{|\widetilde{\mathcal{D}}_n|}\sum_{(\widetilde{\boldsymbol{\kappa}},\widetilde{\boldsymbol{\varphi}})\in\widetilde{\mathcal{D}}_n} \boldsymbol{r}^{(0)}\,, \tag{2.33}$$

using an LRP method characterized by the α-β rule with $\alpha = 1$ and $\beta = 0$ (see Equation (2.13), and Section 2.1.2). Then, looking at the values of $\bar{\boldsymbol{r}}_n$, and recalling that $\bar{\boldsymbol{r}}_n \approx \mathbb{E}_{\boldsymbol{\kappa}\sim q_\kappa}[\boldsymbol{r}^{(0)}]$, it is possible to create a hierarchy for the fractures of the DFN such that $\mathcal{F}_i$ is less relevant than or as relevant as $\mathcal{F}_j$ if

$$(\bar{\boldsymbol{r}}_n)_i \;=\; \mathbb{E}_{\widetilde{\mathcal{D}}_n}[R_i^{(0)}] \;\leq\; \mathbb{E}_{\widetilde{\mathcal{D}}_n}[R_j^{(0)}] \;=\; (\bar{\boldsymbol{r}}_n)_j\,, \tag{2.34}$$

for each $i, j \in \{1, \dots, n\}$. In Figure 2.7 the sorted set of fractures with the corresponding mean relevance scores $(\bar{\boldsymbol{r}}_n)_i$ are visualized, both for DFN158 and for DFN202.

A first observation about the element values of $\bar{\boldsymbol{r}}_n$, for each n = 158,202, is that all the boundary fractures of the DFNs with exiting flux belong to the set of fractures in the top 25% with the highest relevance scores. This observation has non-trivial consequences: it is an important clue that the NNs $\mathcal{N}_n^*$ learned to approximate $\boldsymbol{F}$ coherently with the topology of the network of fractures characterizing the DFNs. Indeed, although one may think as an obvious consequence that NNs mainly look at the inputs corresponding to the fractures with exiting flux, remember that the NNs $\mathcal{N}_n^*$ have no information about relationships between inputs and outputs, except for the coupling they have access to during the training. In particular, assuming that $\mathcal{F}_i$ is a boundary exit fracture, no information about the strict physical-based relationship between the transmissivity κ_i and the computed flux have been given to the NN.

The only exception to the general behavior observed for the outflux fractures is the fracture $\mathcal{F}_{156}$ in DFN202: $\mathcal{F}_{156}$ is indeed an exit fracture, but looking at the actual flux statistics for DFN202, $\mathcal{F}_{156}$ is characterized by an extremely low flux, especially with respect to the ones of the other outflowing boundary fractures. With reference to the box in Figure 2.7 (bottom), the fracture corresponds to the leftmost column with crossing-lines texture. In general, observe that the mean relevance score for fractures with exiting fluxes is characterized by a non-negligible dependence on the mean value of the flux (Tables 2.4, 2.5). Indeed, an average monotonically increasing trend is observed and reported in Figure 2.8.

Continue the analysis and focus on the sub-networks of both DFN158 and DFN202 given by the set of fractures in the top 25% , 50% ,75% relevance scores, respectively. First, some interesting observation about the topology of the graphs that characterize the networks can be done: comparing the graph of the full DFNs and the graphs of the sub-DFNs given by the 25% , 50% ,75% most relevant fractures (Figures 2.9 and 2.10) the observations are the following:

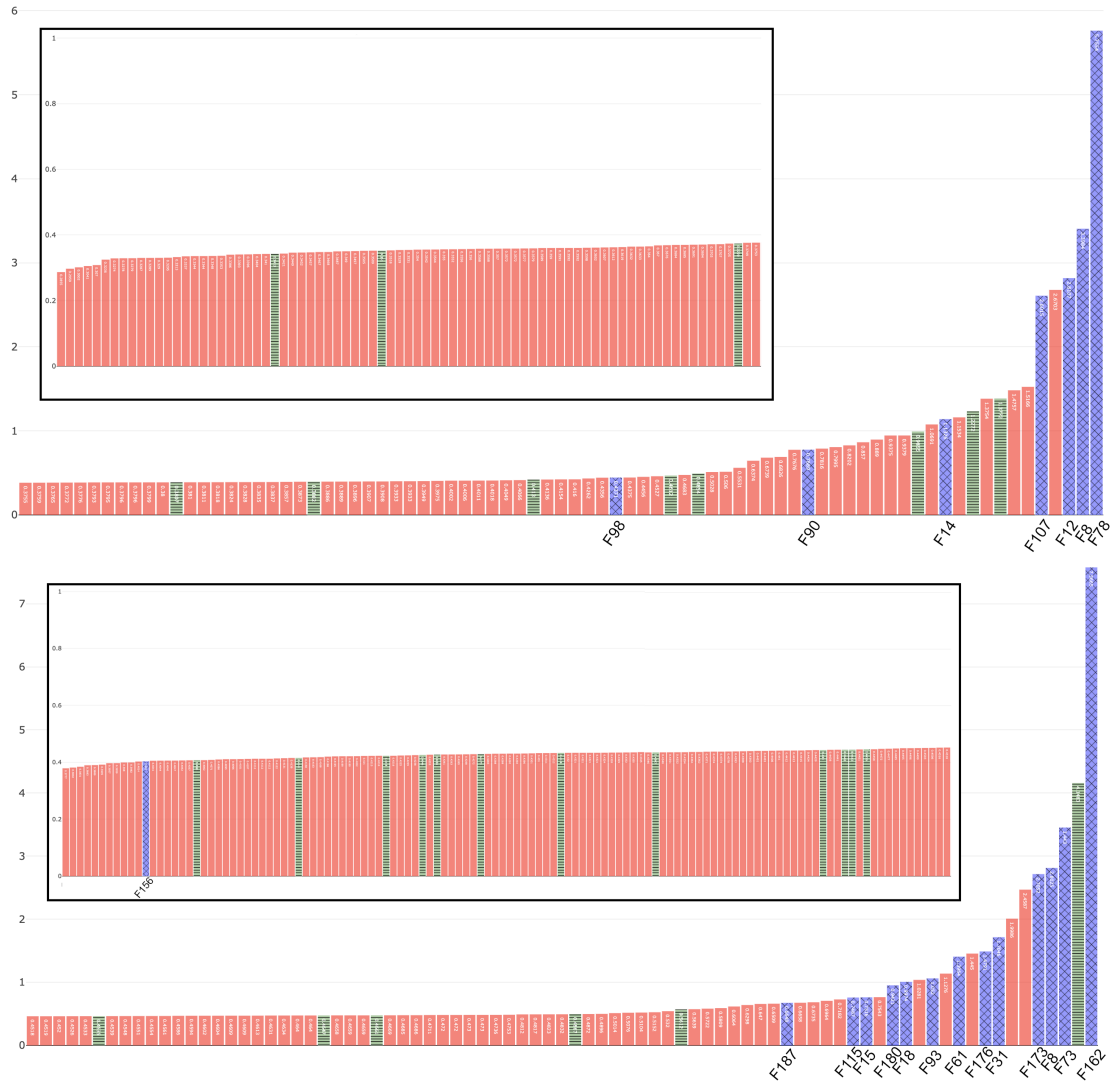

Figure 2.7: Mean relevance scores $(\bar{\boldsymbol{r}}_n)_i$ (y axis) and corresponding fractures $\mathcal{F}_i$ (x axis), sorted in ascending order w.r.t. the score values. Top: DFN158; bottom: DFN202. The box in the top-left corner contains the first 60% of the sorted mean relevance scores. Boundary fractures of the DFN have been highlighted with a crossing-lines texture (exiting flux) and a horizontal-line texture (entering flux). Only outflowing fractures are labeled.

1. The less relevant fractures are in general those belonging to dead-end branches of the networks, since they are the first fractures removed when pruning the DFN graphs to keep only the 75% most relevant ones;

2. Pruning further the DFN graphs (50% of most relevant fractures), other fractures are removed, which are not dead-end fractures but belong to source-sink

	$\mathcal{F}_8$	$\mathcal{F}_{12}$	$\mathcal{F}_{14}$	$\mathcal{F}_{78}$	$\mathcal{F}_{90}$	$\mathcal{F}_{98}$	$\mathcal{F}_{107}$
$\mathbb{E}_{\widetilde{\mathcal{D}}_{158}}[R_i^{(0)}]$	0 .4371	0.7689	1.1326	2.6015	2.8107	3.3984	5.7554
$\mathbb{E}_{\widetilde{\mathcal{D}}_{158}}[\varphi_j]$	0.5372	1.8071	7.0173	13.4984	24.6603	26.8003	67.2993

Table 2.4: DFN158. Mean relevance scores and mean flux values (mm^2/s) of outflux boundary fractures (on $\widetilde{\mathcal{D}}_{158}$). Columns sorted w.r.t. the mean relevance score (ascending order)

	$\mathcal{F}_{156}$	$\mathcal{F}_{187}$	$\mathcal{F}_{115}$	$\mathcal{F}_{15}$	$\mathcal{F}_{180}$	$\mathcal{F}_{18}$	$\mathcal{F}_{93}$
$\mathbb{E}_{\widetilde{\mathcal{D}}_{202}}[R_i^{(0)}]$	0.4020	0.6648	0.7487	0.7516	0.9421	0.9974	1.0520
$\mathbb{E}_{\widetilde{\mathcal{D}}_{202}}[\varphi_j]$	0.2118	1.4390	2.6950	1.3839	1.9175	2.9906	2.4751
	$\mathcal{F}_{61}$	$\mathcal{F}_{176}$	$\mathcal{F}_{31}$	$\mathcal{F}_{173}$	$\mathcal{F}_8$	$\mathcal{F}_{73}$	$\mathcal{F}_{162}$
$\mathbb{E}_{\widetilde{\mathcal{D}}_{202}}[R_i^{(0)}]$	1.3964	1.4791	1.7046	2.7052	2.8025	3.4430	7.5665
$\mathbb{E}_{\widetilde{\mathcal{D}}_{202}}[\varphi_j]$	5.2791	6.5818	6.5228	11.4715	27.0487	19.9980	56.0594

Table 2.5: DFN202. Mean relevance scores and mean flux values (mm^2/s) of outflux boundary fractures (on $\widetilde{\mathcal{D}}_{202}$). Columns sorted w.r.t. the mean relevance score (ascending order)

paths (i.e., paths in the graph that start from any inlet fracture and end in any outlet fracture); removing this fractures is likely to reduce the number of source-sink paths. However, it is worth noting that at least one source-sink path is always left. In particular, the bottleneck fractures (i.e., the cut nodes of the graphs) belonging to source-sink paths are not removed. A clear example is the single fracture that keeps connected the two main halves of DFN158 (see Figure 2.9);

3. Pruning too much the graphs (25% of most relevant fractures) does not guarantee to preserve the connectivity between outflow and inflow fractures and some bottleneck nodes can be removed from the graphs (see Figure 2.10);

4. In the DFN158 case, the LRP algorithm seems to be more sensitive to the actual physical relevance of the fractures in the flux problem than in the DFN202 case. Indeed, for DFN202, in the set of 25% most relevant fractures, five of the outflowing ones belong to a connected component without inlet fractures; these fractures ($\mathcal{F}_{15}, \mathcal{F}_{18}, \mathcal{F}_{115}, \mathcal{F}_{180}, \mathcal{F}_{187}$), disconnected from any source of flux, are deemed by the LRP algorithm as less relevant than the other outflowing ones that are at the end of a source-sink path. Moreover,

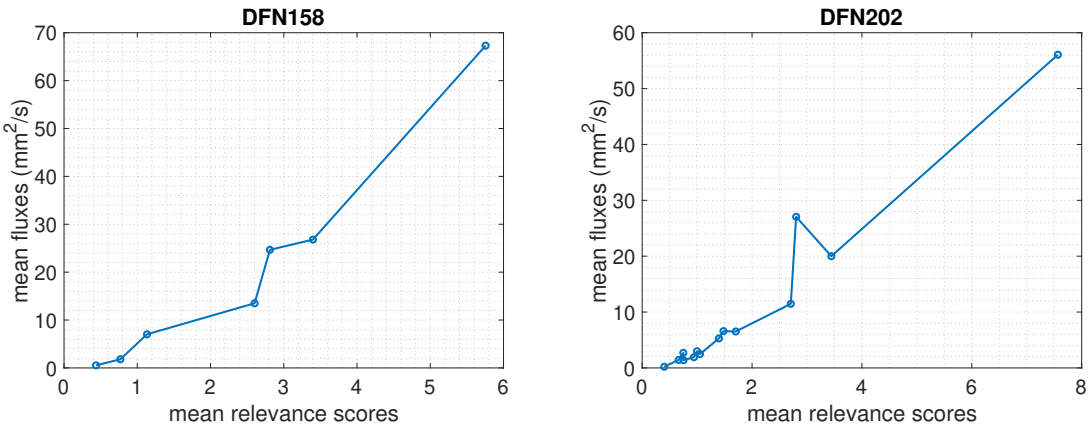

Figure 2.8: DFN158 case (left) and DFN202 case (right). Plot of Mean relevance scores versus mean flux values (mm^2/s) of boundary fractures with exiting flux, computed with respect to $\widetilde{\mathcal{D}}_n$.

fractures $\mathcal{F}_{15}, \mathcal{F}_{18}, \mathcal{F}_{115}, \mathcal{F}_{180}, \mathcal{F}_{187}$ are characterized by a mean relevance score lesser than one (see Table 2.4).

The explanation of this behavior is likely to dwell into the higher number of bottleneck fractures characterizing DFN202 with respect to DFN158. Indeed, the more the bottleneck nodes, the more little differences in the relevance scores can bring to inaccurate fracture removals.

As a final analysis to confirm that the process outlined is able to detect a subnetwork of the DFN that can be considered its backbone, numerical simulations are run on all the sub-DFNs previously introduced. The candidate backbones sub-DFNs are denoted as DFN158$|_{25\%}$, ... , DFN202$|_{75\%}$). More precisely, these simulations run using as input parameters the same transmissivity values of the full DFNs, but restricted to the remaining fractures of the sub-DFN.

After running these simulations for the sub-DFNs, the total flux exiting from the sub-networks is compared with the one of the corresponding full DFN, observing very similar behaviors (see Figure 2.11) that are confirmed by the values reported in Tables 2.6 and 2.7. In general, for DFN158 the sub-DFNs approximate better the behavior of the full-DFN total flux than in the DFN202 case; the reason is attributable to the observations written at item (4). Coherently with the physics of the problem, it can be observed a general decrease of the total exiting flux while pruning the DFN fractures in the graph; however, in both cases (DFN158 and DFN202) a good conservation of the mean total flux and a almost perfect conservation of the standard deviation of the same quantity. In particular, looking at Tables 2.6 and 2.7 the removal of the 75% of the fractures make lose only 14.38% of the flux in DFN158 and 29.01% of the flux in DFN202.

The simulations run on all the sub-DFNs proved that backbones for both DFN158

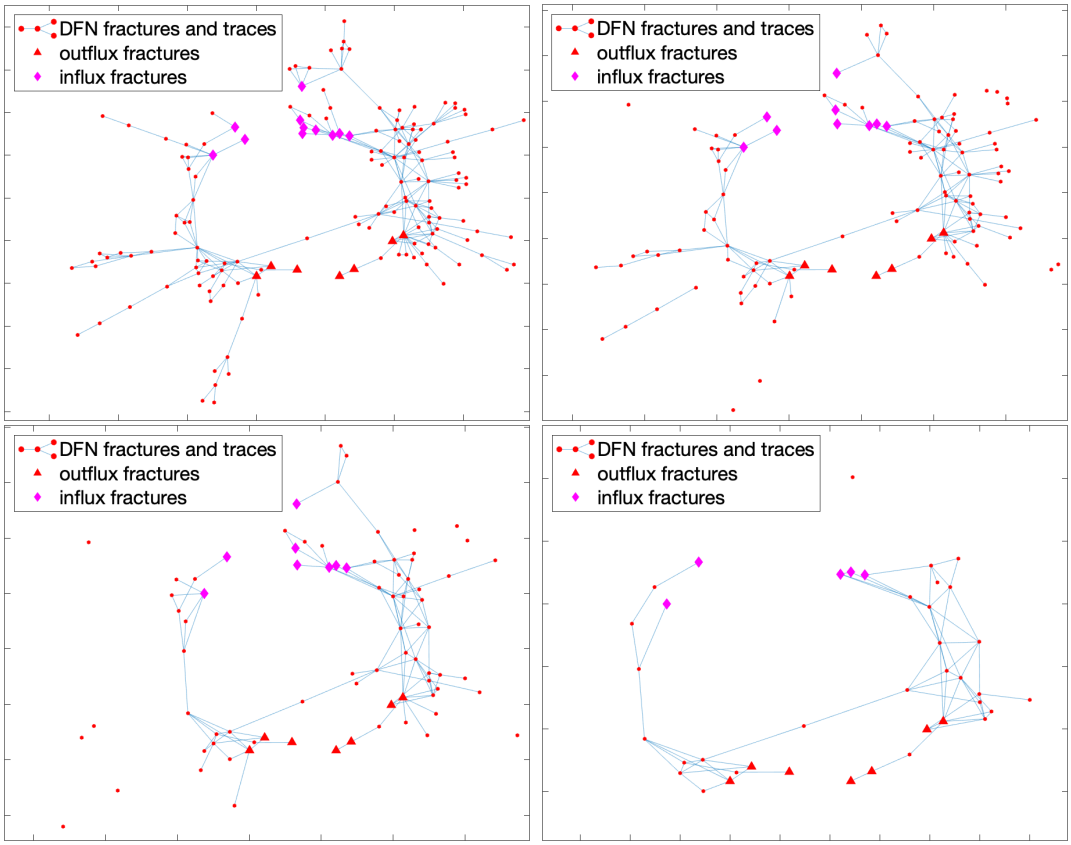

Figure 2.9: Graphs representing networks of fractures. From top to bottom, left to right: DFN158, DFN158$|_{75\%}$, DFN158$|_{50\%}$, DFN158$|_{25\%}$. Triangles/diamonds are fractures with exiting/entering fluxes.

	DFN158	DFN158$|_{75\%}$	DFN158$|_{50\%}$	DFN158$|_{25\%}$
$\mathbb{E}[\sum\varphi]$	141.6738	136.8680 (96.61%)	133.5348 (94.26%)	121.2997 (85.62%)
$\sigma[\sum\varphi]$	27.5962	27.5345 (99.78%)	27.3233 (99.01%)	27.0040 (97.85%)
$D_{\mathrm{KL}}/\mathcal{E}$	-	0.0028	0.0077	0.0451

Table 2.6: DFN158. Mean value and standard deviation of the total flux for the DFN and its sub-DFNs (percentages are computed w.r.t. the DFN values). Last row shows $D_{\mathrm{KL}}/\mathcal{E}$ for the flux distributions of sub-DFNs with respect to the one of the DFN.

and DFN202 have been discovered thanks to the NNs $\mathcal{N}^*_{158}$ and $\mathcal{N}^*_{202}$, respectively. In fact, the identified sub-networks of fractures in the DFNs approximate the flux behaviours of the full networks with respect to 10 000 samplings of transmissivity

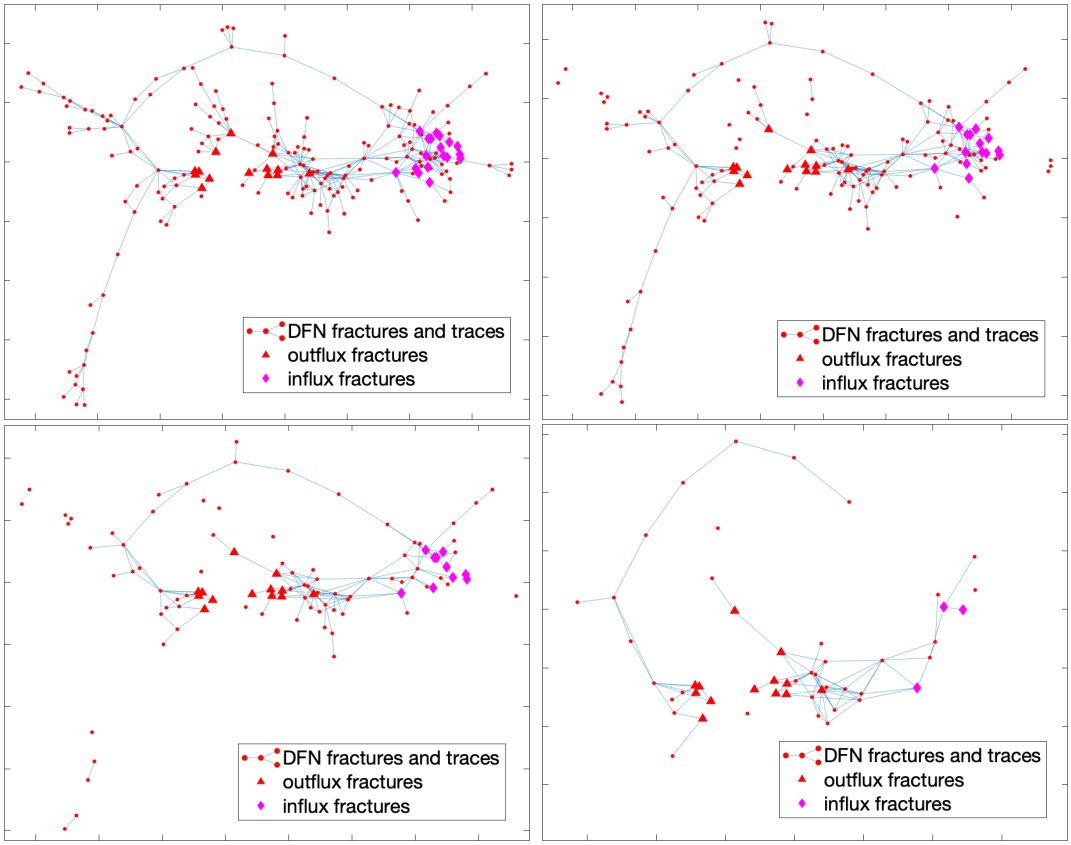

Figure 2.10: Graphs representing networks of fractures. From left to right: DFN202, DFN202$|_{75\%}$, DFN202$|_{50\%}$, DFN202$|_{25\%}$. Triangles/diamonds stars are fractures with exiting/entering fluxes.

	DFN202	DFN202$|_{75\%}$	DFN202$|_{50\%}$	DFN202$|_{25\%}$
$\mathbb{E}[\sum\varphi]$	146.0742	142.9892 (97.89%)	128.2563 (87.80%)	103.6958 (70.99%)
$\sigma[\sum\varphi]$	37.4519	37.2404 (99.44%)	36.3729 (97.12%)	37.4810 (100.08%)
$D_{\mathrm{KL}}/\mathcal{E}$	-	0.0007	0.0217	0.1045

Table 2.7: DFN202. Mean value and standard deviation of the total flux for the DFN and its sub-DFNs (percentages are computed w.r.t. the DFN values). Last row shows $D_{\mathrm{KL}}/\mathcal{E}$ for the flux distributions of sub-DFNs with respect to the one of the DFN.

vectors $\boldsymbol{\kappa} \in \mathbb{R}^n$ (n = 158,202). Then, it is possible to identify a backbone for a given DFN with a given NN that approximates sufficiently well the function $\boldsymbol{F}$ for the actual exiting fluxes computation.

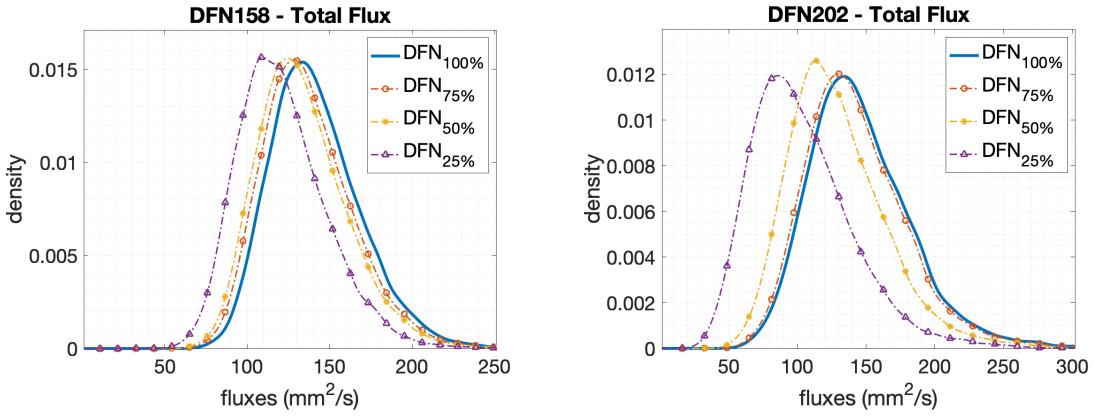

Figure 2.11: Total flux comparisons, for both DFN158 (left) and DFN202 (right), with respect to the probability density functions of their sub-DFNs.

The backbones built using the method illustrated in this section are characterized by the property of being robust with respect to different choices of transmissivity vectors, preserving approximately the total flux behavior. Thus, this backbones theoretically could be able to conserve also the first passage time of particles for transport problems (QoI conserved by backbones in [178, 9, 107, 106, 179]), since this quantity depends on the fluxes characterizing the DFN. However, further studies should be done to confirm this hypothesis, but they are not part of the purpose of this work.

2.3.6 The Refined Method for Backbone Identification

Subsection 2.3.5 has shown that by computing the expected relevance scores with the LRP method for $\mathcal{N}^*_{158}$ and $\mathcal{N}^*_{202}$, it is possible to identify backbones for DFN158 and DFN202 characterized by a good approximation of the total exiting flux of the corresponding full DFNs. However, sorting the DFN fractures with respect to the scores, it has been observed that not important fractures (e.g. the ones belonging to DFN dead-ends branches) sometimes have a higher score than other fractures, especially in lower scores cases (see Figure 2.12, see item (4), p.41). Therefore, the backbone identification method described in Subsection 2.3.5 can be refined by taking into account the fractures of the dead-end branches that, for a better evaluation of the abilities of the new LRP-based feature selection method, were intentionally not removed from the network.

The main idea behind the refinement follows: fix an arbitrary *percentage step* p; create iteratively a sequence of sub-DFNs by alternating two main steps repeated until reaching a minimum percentage threshold τ of the starting total fractures:

- a graph-pruning step that removes dead-end branches from the current sub-DFN. Moreover, the sorted sequence of expected relevance scores is updated

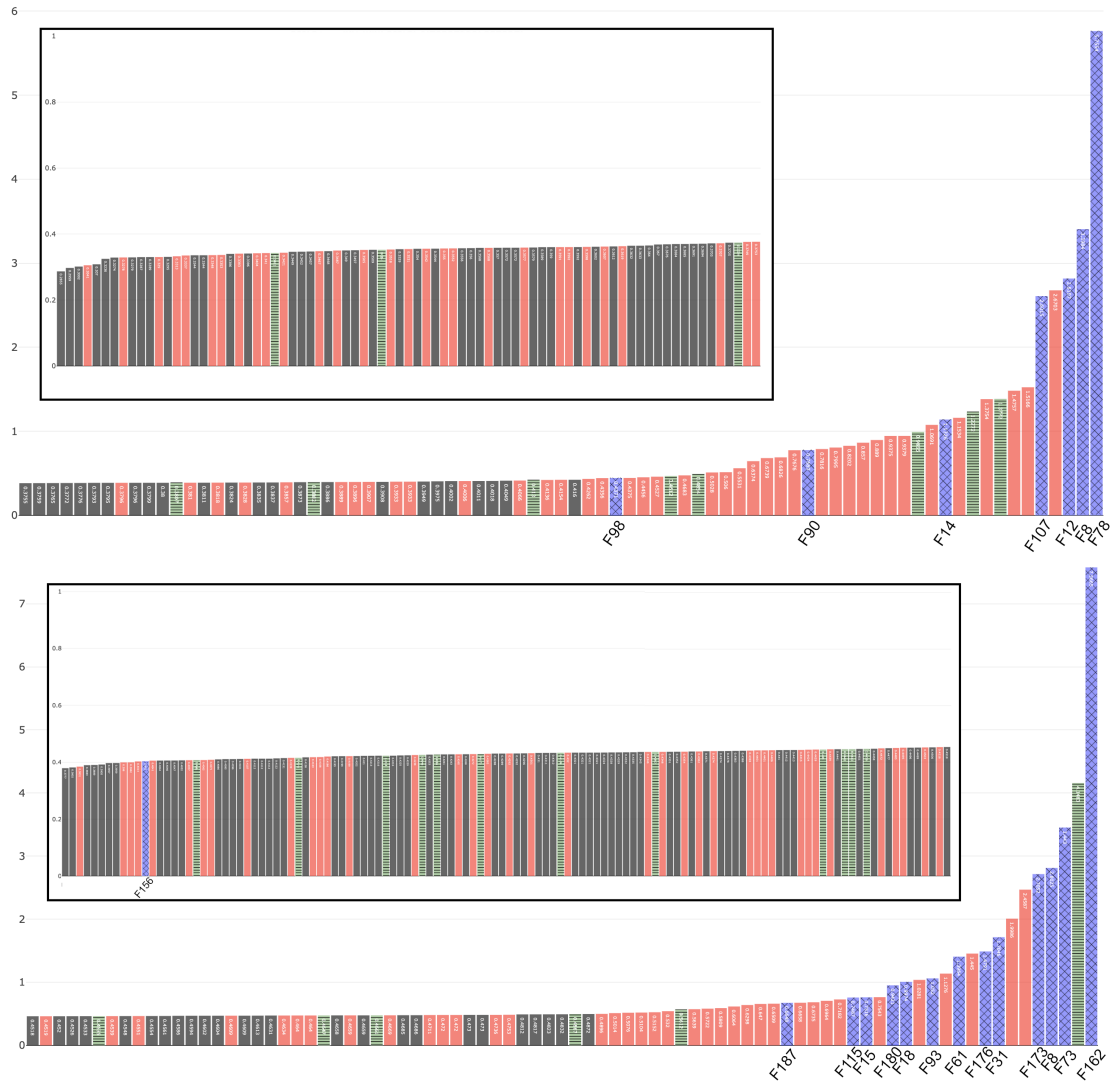

Figure 2.12: Mean relevance scores $(\bar{\boldsymbol{r}}_n)_i$ (y axis) and corresponding fractures $\mathcal{F}_i$ (x axis), sorted in ascending order w.r.t. the score values. Top: DFN158; bottom: DFN202. Box in top-left corner: the first half of the sorted mean relevance scores. DFN boundary fractures have been highlighted with a crossing-lines texture (exiting flux) and a horizontal-line texture (entering flux). DFN fractures belonging to dead-end branches are the darkest ones. Only outflowing fractures are labeled.

by removing the same fractures removed from the sub-DFN;

- a graph-pruning step keeping a specific percentage of fractures according to the updated expected relevance scores (as described in Subsection 2.3.5);

More specifically, the whole refinement procedure is described by Algorithm 2.3.1.

Algorithm 2.3.1 (Refined Backbone Identification).

Data: *G (graph of the full DFN), $p \in (0,1)$ (percentage step), L (fracture/node sequence, sorted in increasing order with respect to the relevance score value), $\tau \in (0,1)$ (minimum percentage threshold).*

Outputs: *$\mathcal{L}$ (sequence containing the subs-sequences of L and characterizing the sub-DFNs).*

Procedure *Refined Backbones(G, L, p, τ):*

1. *$\mathcal{L} \leftarrow$ empty sequence;*
2. *$N_0 \leftarrow$ number of nodes in G;* (num. of fractures of the full DFN)
3. *$\rho \leftarrow 1$;* (percentage counter)
4. *$R \leftarrow$ subset of G's nodes s.t. they belong to dead-end branches;*
5. *$L \leftarrow$ remove all the $v \in R$ from L;* (L still sorted w.r.t. relevance scores)
6. *$G \leftarrow$ sub-graph of G given by nodes in L;*
7. *$N \leftarrow$ number of nodes in G;*
8. ***while** $N/N_0 > \tau$ **do:***
9. ***while** $\rho \geq N/N_0$ **do:*** (in case $(|R|/N_0) > p$)
10. $\rho \leftarrow \rho - p$;
11. ***end while***
12. ***if** $\rho < \tau$ **do** break;*
13. *$L \leftarrow$ last `round`(ρN_0) elements of L;* (L still sorted w.r.t. relevance scores)
14. *$G \leftarrow$ sub-graph of G given by nodes in L;*
15. *repeat lines 4-7;*
16. *add G to $\mathcal{L}$;*
17. ***end while***

In Algorithm 2.3.1, the step corresponding to the removal of the dead-ends, both from the graph and the sequence of relevance scores, is described in lines 4-7; the step that keeps only a particular number of fractures (i.e. $[\rho N_0]$) with higher relevance score (as in Subsection 2.3.5) is described in lines 13-14. In general, the algorithm returns, for each DFN, a sequence of sub-DFNs

$$\left(\mathrm{DFN}|_{(\rho_i - \epsilon_i)100\%}\right)_{i=1,\dots,s}, \tag{2.35}$$

		while-step 1		while-step 2	
	remove dead-ends	$[\rho_1 N_0]$ most rel. fractures	remove dead-ends	$[\rho_2 N_0]$ most rel. fractures	remove dead-ends
DFN158	87 (55.06%)	79 (50%)	**77 (48.73%)**	40 (25.32%)	**39 (24.68%)**
DFN202	108 (53.47%)	101 (50%)	**99 (49.01%)**	51 (25.25%)	**46 (22.77%)**

Table 2.8: Summary of outcome of Algorithm 2.3.1 ($p = \tau = 0.25$) showing the number of nodes in G at each step (percentages are w.r.t. the initial number of nodes). Bold values characterize the returned sub-DFNs.

where s is the number of main iterations executed by the algorithm, ρ_i is the value of ρ at the i-th iteration and ϵ_i is the percentage of fracture removed at line 15 at the i-th iteration. The stopping criteria of the algorithm are such that ρ_s is always greater than the threshold τ but $\tau > \rho_s - p$; on the other hand, $(\rho_s - \epsilon_s)$ can be lesser than τ, if the dead-end fractures in $\text{DFN}|_{\rho_s}$ are more than $(\rho_s - \tau)N_0$. In this last case, the stopping criterion of the main while loop is reached. Concluding the description of the algorithm, the role of the inner while loop (line 9) is only to skip the values of ρ such that ρN_0 is greater than the current number of nodes in G.

One of the most important observation for Algorithm 2.3.1 is that such a kind of refinement actually has a negligible computational cost, since the computation of dead-end branches of a graph is not expensive; all the remaining operations are the same ones illustrated in Subsection 2.3.5.

Now, the refined method can be applied to DFN158 and DFN202. Then, a comparison of the results with the previous ones is conducted. For both DFN158 and DFN202, Algorithm 2.3.1 run with $p = \tau = 0.25$, such that it should return a sequence of sub-DFNs

$$(\text{DFN}|_{(0.75-\epsilon_1)100\%}, \text{DFN}|_{(0.50-\epsilon_2)100\%}, \text{DFN}|_{(0.25-\epsilon_3)100\%}) \tag{2.36}$$

comparable with those computed in Subsection 2.3.5. However, since both in DFN158 and in DFN202 the number of fractures belonging to dead-end branches is greater than 25% of the total number of fractures, Algorithm 2.3.1 does not return the first element of sequence (2.36). A brief summary of the operations performed by the refined method is described in Table 2.8.

Looking at the sub-DFNs graphs obtained with the refined method (see Figure 2.13), observe that from all the influx fractures exist a path that ends in an outflux fracture. This simple fact is actually very important; indeed, it is an evidence of the efficiency of the refined method introduced, since the original method was not able to preserve a source-sink path for each influx fracture of the sub-DFNs (e.g., recall the observations made for $\text{DFN202}|_{25\%}$).

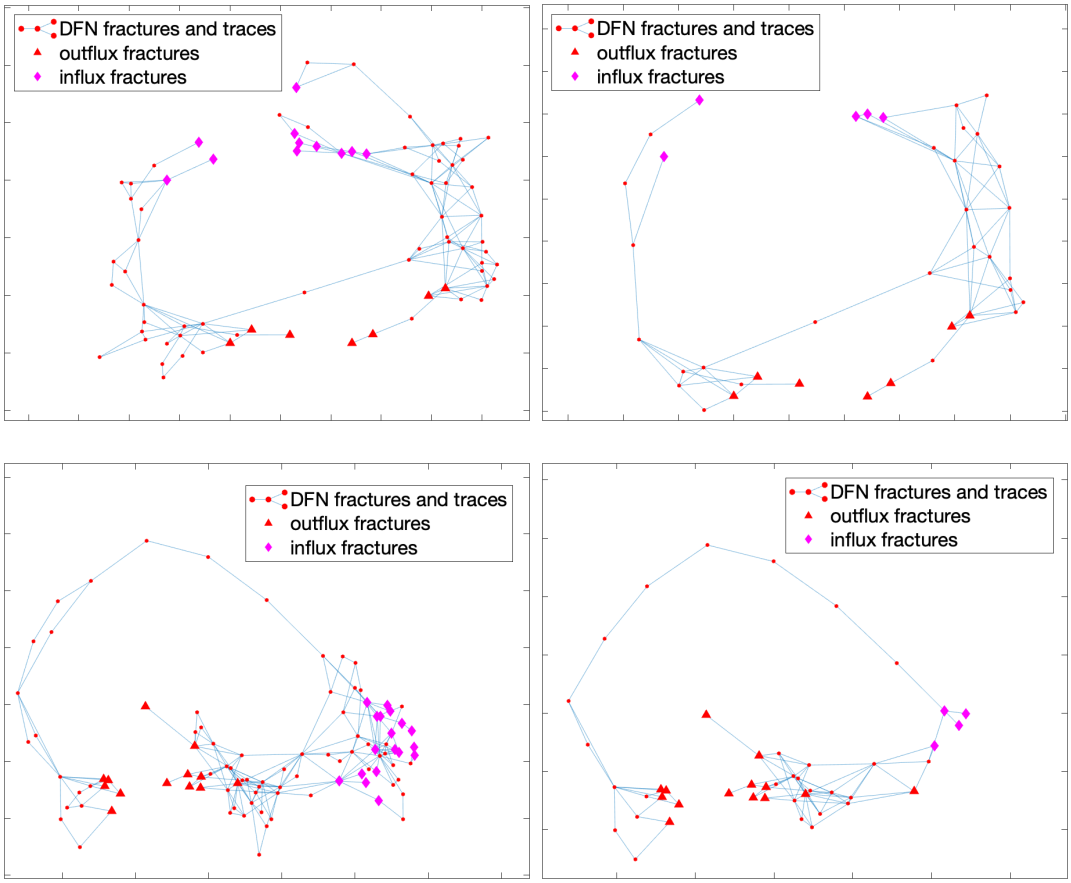

Figure 2.13: Graphs representing the network of fractures of: $\text{DFN158}|_{48.73\%}$ (top-left), $\text{DFN158}|_{24.68\%}$ (top-right), $\text{DFN202}|_{49.01\%}$ (bottom-left), $\text{DFN202}|_{22.77\%}$ (bottom-right). Triangles/diamonds are fractures with exiting/entering fluxes.

Conclude the comparison analysing the total fluxes exiting from the new sub-DFNs. The values in Tables 2.9 and 2.10 show that the sub-DFNs obtained with the refined method are characterized by a better flux approximation of the total flux exiting from the full DFN (see also Figures 2.14 and 2.15 for a visual comparison). Therefore, in the end, the method characterized by Algorithm 2.3.1 is a good refinement of the method proposed in Subsection 2.3.5, returning subnetworks of the DFN that can be considered its backbones.

2.4 Conclusions and Future Directions

This chapter has highlighted the compelling opportunities offered by the class of algorithms named eXplainable Artificial Intelligence (XAI). In particular, it has

	DFN158	$\text{DFN158}\vert_{48.73\%}$	$\text{DFN158}\vert_{24.68\%}$
$\mathbb{E}[\sum\varphi]$	141.6738	138.2734 (97.60%)	124.8932 (88.15%)
$\sigma[\sum\varphi]$	27.5962	27.5544 (99.85%)	26.9265 (97.57%)
$D_{\mathrm{KL}}/\mathcal{E}$	-	0.0014	0.0314

Table 2.9: DFN158. Mean value and standard deviation of the total flux for the DFN and its sub-DFNs computed with Algorithm 2.3.1 (percentages are computed w.r.t. the DFN values). Last row shows $D_{\mathrm{KL}}/\mathcal{E}$ for the flux distributions of sub-DFNs with respect to the one of the DFN.

	DFN202	$\text{DFN202}\vert_{49.01\%}$	$\text{DFN202}\vert_{22.77\%}$
$\mathbb{E}[\sum\varphi]$	146.0742	142.9952 (97.89%)	112.4792 (77.00%)
$\sigma[\sum\varphi]$	37.4519	37.5844 (100.35%)	38.0651 (101.64%)
$D_{\mathrm{KL}}/\mathcal{E}$	-	0.0008	0.0695

Table 2.10: DFN202. Mean value and standard deviation of the total flux for the DFN and its sub-DFNs computed with Algorithm 2.3.1 (percentages are computed w.r.t. the DFN values). Last row shows $D_{\mathrm{KL}}/\mathcal{E}$ for the flux distributions of sub-DFNs with respect to the one of the DFN.

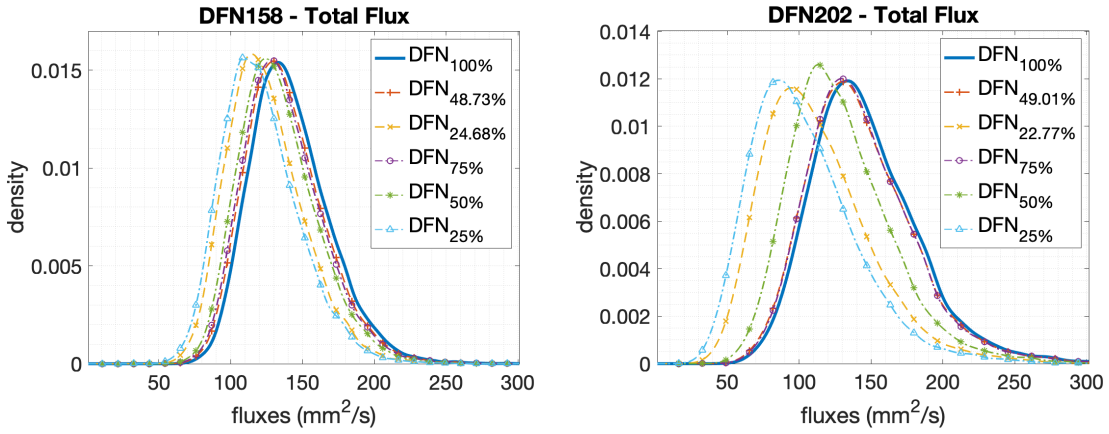

Figure 2.14: Total flux comparisons, for both DFN158 (top) and DFN202 (bottom), with respect to the probability density functions of all their sub-DFNs (computed with both two methods described in this work).

shown how algorithms and methods of XAI could provide insights about the natural world. This work has deeply described and formalized the Layer-wise Relevance

Propagation (LRP) algorithm for outcome explanation. In addition, it has proposed an extension to globally investigate a studied ML model on all its data domain. The described pipeline illustrates the attractive possibilities offered by the interaction between a numerical model simulating a physical phenomenon and the XAI algorithm applied to NNs. In particular, if the data source is a numerical simulator, an NN approximating the simulator paired with an XAI algorithm can enrich the investigation of the phenomenon of interest.

The DFN case study represents an example of this possibility. Indeed, the experiments show that the multitask NN is able to correctly approximate the original simulator $\boldsymbol{F}$, and the LRP algorithm identifies the subset of significant fractures causing the outflow. From a statistical perspective, the excluded fractures are irrelevant for the flow propagation. The proposed extension of LRP as a model interpretation method has made possible the mentioned interaction, allowing a model analysis on all the domain space of the input variables. The evidences are confirmed by the validation procedure, which employs a simplified simulator inferred by the features extracted according to the result of the global LRP algorithm. Moreover, the simplified simulator represents a down scale of the problem. It provides a faster tool to analyze the same quantities of interest. In addition, the refined procedure have improved the LRP algorithm capabilities of selecting the relevant fractures for the outflow. Future research can explore the limitations of the global LRP method and employ more recent algorithms inside a similar pipeline. Furthermore, an open question remains how precisely measure the quality of the computed relevance.

These experiments are in line with the principles that define Theory-guided Data Science [112] recalled in the introductory Chapter 1. The proposed pipeline is generalizable to other ML models and XAI algorithms. In addition, the discussed framework is suitable for other domains and problems in science and engineering. Furthermore, it opens appealing impacts on industrial applications whenever the simulator is the primary object to infer the occurring events.

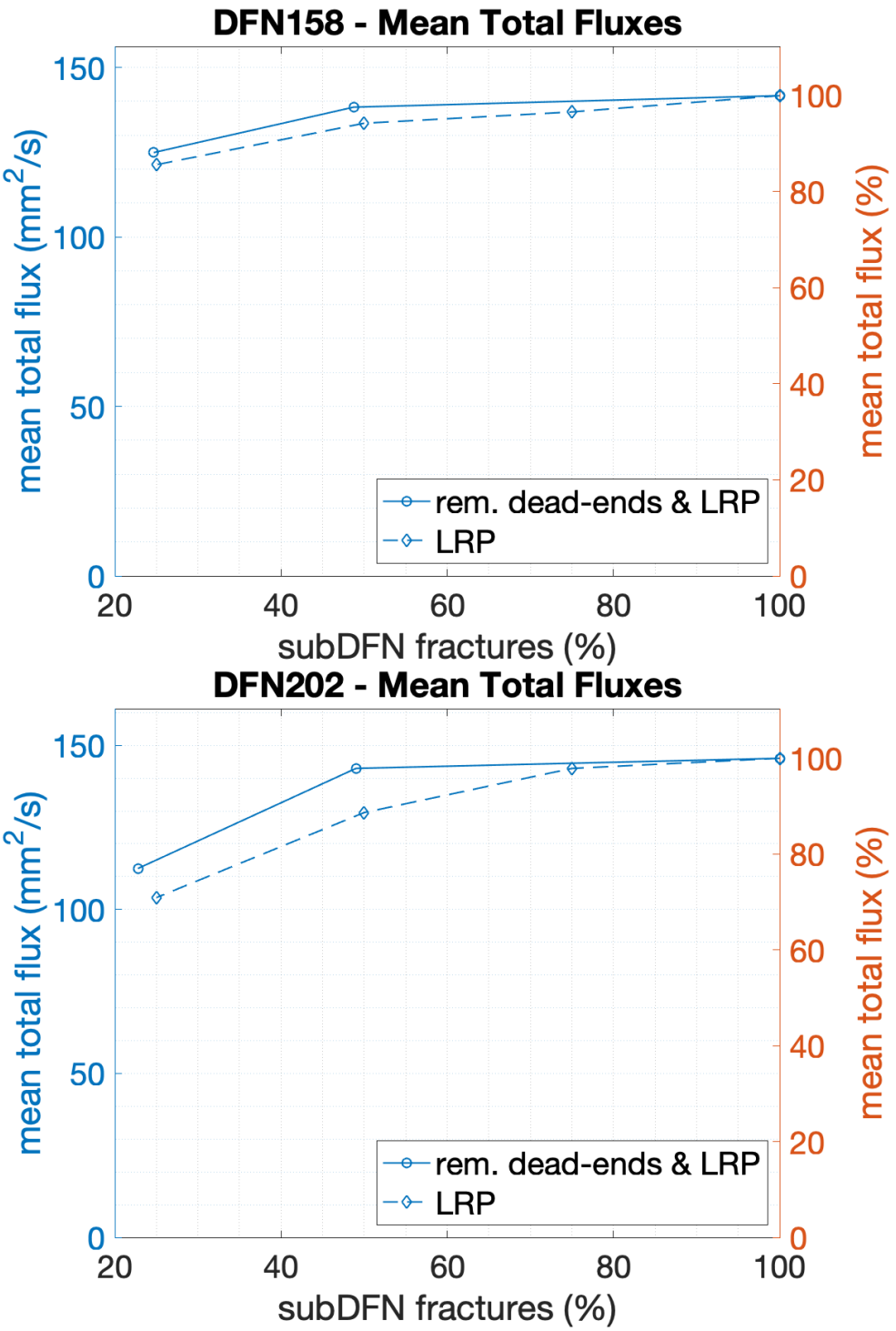

Figure 2.15: Mean total flux comparison for both DFN158 (left) and DFN202 (right), with respect to their sub-DFNs. Continuous line characterizes the values corresponding to the sub-DFNs obtained with the refined method. Dotted lines characterize the values corresponding to the sub-DFNs obtained with the not refined method.

Chapter 3

Inserting knowledge into Neural Networks

3.1 A New Architecture for Graph Regression Tasks

Graphs are frequently used to describe and study many phenomena, such as transportation systems, epidemic or economic-default spread, electrical circuits, and social interactions; the literature typically refers to the use of graph theory to analyze such phenomena with the term "network analysis" [38].

Recently, the neural network (NN) community have proposed new key contributions to network analyses; in particular, Deep Learning (DL) approaches [122] can be extended to graph-structured data via graph neural networks (GNNs). The origin of GNNs dates back to the late 2000s [88, 131, 166], when their processing was still too computationally expensive [191]. Nonetheless, the huge success of Convolutional Neural Networks (CNNs) [86] inspired a new family of GNNs, re-defining the notion of convolution for graph-structured data and developing the graph convolutional networks (GCNs). According to the taxonomy defined in [191], two main families of GCNs can be observed: the spectral-based GCNs [41, 53, 115], which are based on the spectral graph theory, and the spatial-based GCNs [77, 95, 134, 143], which aggregates the information of neighbor nodes'. In particular, the spatial-based GCNs are nowadays preferred in many applications, thanks to their flexibility and efficiency [191].

Typically, GCNs are used to perform the following tasks on graph data [191]: (i) semi-supervised node regression or classification; (ii) edge classification or link prediction; and (iii) graph classification. Nonetheless, even if GCNs have been proven to be good instruments to learn graph data, some challenges still exist. The two main challenges for GCNs are [191]: first, to build deep architectures with good performances; second, to be scalable for large graphs. The first issue is the

most problematic one; indeed, the success of DL lies in its depth, but the literature suggests that going deeper into a GCN is not usually beneficial [191]. Moreover, experimental results for the spectral-based GCNs showed that performances dropped considerably as the number of graph convolutional layers increased [123].

This chapter presents a new type of spatial-based graph convolutional layer designed for regression tasks on graph-structured data, a framework for which previous GCNs are not well suited. Given a graph G with n nodes, a regression task on graph-structured data based on G consists of approximating a function $\boldsymbol{F} : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^m$, $m \leq n$, depending on the adjacency matrix of G. In addition, $\boldsymbol{F}$ takes as input a vector of feature values assigned to the nodes of G, and it returns the m values related to a fixed subset of m nodes. This type of regression task has applications in many interesting fields, such as circulation with demand (CwD) problems (see [116, chap. 7.7]), network interdiction models (NIMs) [57], and flux regression problems in underground fractured media [26, 35]. A classic multi-layer perceptron (MLP), or its suitable variants, can perform this regression task on the graph data with a good performance [35, 26], implicitly learning the node relationships during the training (see [32, 34]). On the other hand, the current GCNs in the literature are not comparable to MLPs for such a regression task; indeed, as mentioned above, they are designed mainly for other kinds of tasks and, in practice, they cannot exploit deep architectures. Then, the idea is to define a new graph convolutional layer that exploits the graph structure to improve the training of the NN (compared to an MLP), and that makes it possible to build deep NN architectures. The new convolution operation for graph data defined here belongs to the class of message passing NNs [40, 80] and it is directly inspired by the convolution of CNNs [86, chap. 9]. Nonetheless, similarities with other spatial-based GNN exists, like the graph layers of NN For Graph (NN4G) [131], or the Diffusion-Convolutional NN (DCNN) [15].

Informally, the simplest version of our graph layer is characterized by a filter with one weight w_i associated with each graph node v_i. Then, the output feature of a node is computed by summing up the input features of the node itself and of its neighbors, where each one is multiplied by the corresponding node weights. The new type of graph layer is called Graph Informed (GI) layer. Indeed, given a scalar value w_j associated with each graph vertex v_j, a GI layer looks like a fully-connected (FC) layer where, for each unit v_i connected with a unit v_j of the previous layer, the weight w_{ji} is equal to 0 if (v_j, v_i) is not an edge of the graph and $i \neq j$, otherwise $w_{ji} = w_j$ (see Equation (3.4) in Section 3.2). These NNs are defined as Graph-Informed Neural Networks (GINNs). The feature map representation of deeper layers in GINNs is built by filtering out the connections among NN units of subsequent layers that are not connected in the graph describing the task. Thus, the process of representation learning is informed by the graph connectivity. Numerical experiments show the potentiality of the GI layers, which involves training deep NNs made up of a sequence of GI layers. In particular, the numerical experiments

showed that GINNs performs better than MLPs for the studied regression tasks. An emerging observation from the experiments concerns the NN depth: in contrast with the previous GCNs, the depth seems beneficial for GINNs.

The work is organized as follows: in Section 3.2, the GI layers are formally introduced and defined, explaining their fundamental operations, properties, similarities, and differences with respect to other spatial-based graph convolutional layers. Section 3.3 presents a case study derived from the problem of maximum-flow: in the classic formulation, given a flow network described by a graph with a source, a sink and edges capacities, the objective is to compute the maximum flow value between the source and the sink. The classic formulation is extended to the stochastic maximum-flow problem, where the capacities of the flow network are modeled by random variables. The stochastic maximum-flow problem is translated to a graph regression task, to provide the first experimental case study. The second case study is the real-world application to Discrete Fracture Networks, already presented in Section 2.2: the flux regression problem of underground networks of fractures. Section 3.4 is dedicated to the numerical experiments; in particular, it analyzes the regression performance of the GINNs on the two presented case studies, and it compares the results with the MLPs baseline on the same problems. Section 3.5, summarizes the obtained results, draw conclusions and discusses the possibilities of future research.

3.2 Formalizing the Graph-Informed Layer

In this section, staring from the adjacency matrix of a graph, the mathematical formulation of the new GI layer is introduced. In particular, this part details the formalism that defines the function $\mathcal{L}^{GI}$, describing a GI layer L^{GI}. From now on, the function describing a generic NN layer will be denoted as the characterizing function of the layer.

Definition 3.2.1 (Graph-Informed layer: basic form). *Let $A \in \mathbb{R}^{n\times n}$ be the adjacency matrix characterizing a given graph $G = (V, E)$ without self-loops, and let $\widehat{A}$ be the matrix $\widehat{A} := A + \mathbb{I}_n$, where $\mathbb{I}_n \in \mathbb{R}^{n\times n}$ is the identity matrix. Then, a graph-informed (GI) layer L^{GI}, with respect to the graph G, is an NN layer with n units connected to a layer with outputs in $\mathbb{R}^n$ with a characterizing function $\mathcal{L}^{GI}\colon \mathbb{R}^n \to \mathbb{R}^n$ defined by*

$$\mathcal{L}^{GI}(\boldsymbol{x}) = \boldsymbol{f}\left(\widehat{W}^{\top}\,\boldsymbol{x} + \boldsymbol{b}\right), \tag{3.1}$$

where:

- *Given a vector $\boldsymbol{w} \in \mathbb{R}^n$ of weights associated with the vertices V, the defined filter of L^{GI} the matrix $\widehat{W}$ is obtained by multiplying the i-th row of $\widehat{A}$ by the weight w_i, i.e.,*

$$\widehat{W} := \mathrm{diag}(\boldsymbol{w})\hat{A}\,, \tag{3.2}$$

where $\mathrm{diag}(\boldsymbol{w})$ *is the diagonal matrix with a diagonal that corresponds to vector* $\boldsymbol{w}$*;*

- *Given the layer activation function* $f\colon \mathbb{R} \to \mathbb{R}$*,* $\boldsymbol{f}$ *denotes the element-wise application of* f*;*
- $\boldsymbol{b} \in \mathbb{R}^n$ *is the vector of biases.*

Broadly speaking, given a directed graph $G = (V, E)$, with n nodes and an adjacency matrix $A \in \mathbb{R}^{n\times n}$, the main idea behind a GI layer is to generalize the convolutional layer filters to the graph-structured features. Indeed, the objective is to endow the layer with the implicit relationship between the features of the adjacent graph nodes, and also to take advantage of the sparse interaction- and parameter-sharing properties typical of convolutional NNs (see [86, chapter 9.2]).

Convolutional layers rely on the identification of images as lattices of pixels. The main idea for the GI layer formulation is to adapt convolutional layer concepts to graphs that are not characterized by a lattice structure. The filter mechanisms of the convolutional layers is here generalized to graph-structured data, introducing the concept of graph-based filters. In practice, for each node of the graph G, consider a weight w_j which is associated to node $v_j \in V$ and re-define the convolution operation as

$$x'_i = \sum_{j\in \mathrm{N_{in}}(i)\cup\{i\}} x_j\, w_j + b_i\,, \tag{3.3}$$

where

- x_j denotes the input feature of node $v_j \in V$, for each $j = 1, \dots, n$;
- $\mathrm{N_{in}}(i)$ is the set of indices j, such that there exists an incoming edge $(v_j, v_i) \in E$;
- b_i is the bias corresponding to node v_i;
- x'_i is the output feature associated to v_i, computed by the filter (see Figure 3.1).

For a undirected graph G, Equation (3.3) does not change, since a undirected edge $\{v_j, v_i\}$ is equivalent to two directed edges, (v_j, v_i) and (v_i, v_j). Indeed, Definition 3.2.1 holds for both directed and undirected graphs.

Similar to the convolutional layers, which act on the current pixel and on all its neighbors for computing the output feature, in (3.3) the layer acts on x_i and on the values associated with the incoming neighbors of v_i for the computation of x'_i (see Figure 3.1). Nonetheless, despite the inspiration received from convolutional layers, a GI layer L^{GI} described by (3.3) can be seen also as a constrained FC layer, where the weights are such that

$$w_{ji} = \begin{cases} w_j\,, & \text{if } (v_j, v_i) \in E \\ w_i\,, & \text{if } j = i \\ 0\,, & \text{otherwise} \end{cases}, \tag{3.4}$$

for each $i, j = 1, \ldots, n$.

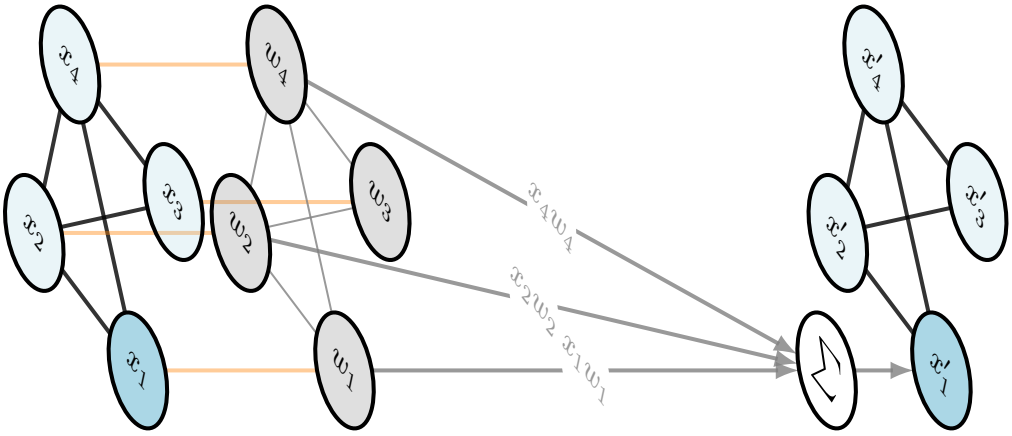

Figure 3.1: Case of undirected graph with $n = 4$ nodes. Example of the action of a filter $\boldsymbol{w} \in \mathbb{R}^4$ (grey "layer" of the plot) of a GI layer, applied to feature x_1 of the first graph-node v_1, for the computation of x'_1; for simplicity, the bias is not illustrated. The orange edges describe the multiplication of feature x_i, of node v_i, with the filter's weight w_i, for each $i = 1, \ldots, 4$.

In the next sections, generalizations of the action of these kinds of layers are provided in order to make possible to:

(i) receive any arbitrary number $K \geq 1$ of input features for each node;

(ii) return any arbitrary number $F \geq 1$ of output features for each node.

3.2.1 Generalization to K Input Node Features

Equation (3.1) describes the simplest case of GI layers, where just one feature is considered for each vertex of the graph for both the inputs and the outputs. The previous definition can be generalized by taking into account a larger number of features tackled by L^{GI}.

Definition 3.2.2 (Graph-Informed layer with K input features per node). *Let G, A, and $\widehat{A}$ be as in Definition 3.2.1. Then, a GI layer with $K \in \mathbb{N}$ input features is an NN layer with n units connected to a layer with outputs in $\mathbb{R}^{n \times K}$ with a characterizing function $\mathcal{L}^{GI} : \mathbb{R}^{n \times K} \to \mathbb{R}^n$ defined by*

$$\mathcal{L}^{GI}(X) = \boldsymbol{f}\left(\widetilde{W}^\top \text{vertcat}(X) + \boldsymbol{b}\right), \tag{3.5}$$

where:

- $X \in \mathbb{R}^{n \times K}$ *is the input matrix (i.e., the output of the previous layer) and* $\text{vertcat}(X)$ *denotes the vector in* $\mathbb{R}^{nK}$ *obtained by concatenating the columns of* X*;*
- *Given the matrix* $W \in \mathbb{R}^{n \times K}$*, the defined filter of* L^{GI}*, whose columns* $\boldsymbol{w}_{\cdot 1}, \ldots, \boldsymbol{w}_{\cdot K} \in \mathbb{R}^n$ *are the vectors of weights associated with the* k*-th input feature of the graph's vertices, the matrix* $\widetilde{W} \in \mathbb{R}^{nK \times n}$ *is defined as*

$$\widetilde{W} := \begin{bmatrix} \widehat{W}^{(1)} \\ \vdots \\ \widehat{W}^{(K)} \end{bmatrix} = \begin{bmatrix} \text{diag}(\boldsymbol{w}_{\cdot 1})\widehat{A} \\ \vdots \\ \text{diag}(\boldsymbol{w}_{\cdot K})\widehat{A} \end{bmatrix} \in \mathbb{R}^{nK \times n}. \tag{3.6}$$

The idea behind the generalization from Definitions 3.2.1 to 3.2.2 is rather simple. Let L be an NN layer with outputs in $\mathbb{R}^{n \times K}$, $K \geq 1$. Therefore, a generic output of L is a matrix $X \in \mathbb{R}^{n \times K}$, whose row $i \in \{1, \ldots, n\}$ describes the K features $x_{i1}, \ldots, x_{iK}$ of node v_i; on the other hand, each column $\boldsymbol{x}_{\cdot 1}, \ldots, \boldsymbol{x}_{\cdot K}$ of X is equivalent to the output of an NN layer with outputs in $\mathbb{R}^n$.

Therefore, the generalization consists of summing up the action of the K "basic" single-input filters $\boldsymbol{w}_{\cdot 1}, \ldots, \boldsymbol{w}_{\cdot K}$, where each one is applied to $\boldsymbol{x}_{\cdot 1}, \ldots, \boldsymbol{x}_{\cdot K}$, respectively; then, the bias vector is added to this sum, and the activation function is applied. However, this approach is equivalent to (3.5), i.e., defining one filter W obtained from the concatenation of the basic filters. Indeed:

$$\sum_{k=1}^{K} \widehat{W}^{(k)\top} \boldsymbol{x}_{\cdot k} = \widetilde{W}^{\top} \text{vertcat}(X). \tag{3.7}$$

Remark 3.2.3 (Parallelism with convolutional layers). It is worth noting that the operations summarized in (3.5) are an adaptation of the convolutional layer operations to the graph-based inputs. Indeed, the input $X \in \mathbb{R}^{n \times K}$ is equivalent to an $n \times 1$ image with K channels, while $\boldsymbol{w}_{\cdot k}$ is equivalent to the part of the convolutional filter corresponding to the k-th channel of the input image. Then, the output $\mathcal{L}^{GI}(X) \in \mathbb{R}^n$ is equivalent to the feature map of the convolutional layers.

3.2.2 Generalization to F Output Node Features

A further generalization (3.5) is possible by increasing the number of output features per node returned by the GI layer. This operation is equivalent to building a GI layer characterized by a number $F \geq 1$ of matricial filters, where each one used to compute one of the output features. In a nutshell, the output of these general GI layers is a matrix $Y \in \mathbb{R}^{n \times F}$ whose l-th column $\boldsymbol{y}_{\cdot l} \in \mathbb{R}^n$, $l = 1 \ldots, F$, describes the l-th feature of the nodes of G.

Definition 3.2.4 (Graph-Informed layer: general form). *Let $G, A, \widehat{A}$ be as in Definition 3.2.1. Then, a GI layer with $K \in \mathbb{N}$ input features and $F \in \mathbb{N}$ output features is an NN layer with nF units connected to a layer with outputs in $\mathbb{R}^{n\times K}$ with a characterizing function $\mathcal{L}_{GI} : \mathbb{R}^{n\times K} \to \mathbb{R}^{n\times F}$ defined by*

$$\mathcal{L}^{GI}(X) = \boldsymbol{f}\left(\widetilde{\mathbf{W}}^{\top}\mathrm{vertcat}(X) + B\right), \tag{3.8}$$

where:

- *Define the filter of L^{GI}, the tensor $\mathbf{W} \in \mathbb{R}^{n\times K\times F}$, given by the concatenation along the third dimension of the weight matrices $W^{(1)}, \dots W^{(F)} \in \mathbb{R}^{n\times K}$, corresponding to the F output features of the nodes. Each column $\boldsymbol{w}_{\cdot k}^{(l)} \in \mathbb{R}^n$ of $W^{(l)}$ is the basic filter describing the contribution of the k-th input feature to the computation of the l-th output feature of the nodes, for each $k = 1, \dots, K$, and $l = 1, \dots, F$;*
- *The tensor $\widetilde{\mathbf{W}} \in \mathbb{R}^{nK\times F\times n}$ is defined as the concatenation along the second dimension (i.e., the column dimension) of the matrices $\widetilde{W}^{(1)}, \dots, \widetilde{W}^{(F)}$, such that*

$$\widetilde{W}^{(l)} := \begin{bmatrix} \widehat{W}^{(l,1)} \\ \vdots \\ \widehat{W}^{(l,K)} \end{bmatrix} = \begin{bmatrix} \mathrm{diag}(\boldsymbol{w}_{\cdot 1}^{(l)})\widehat{A} \\ \vdots \\ \mathrm{diag}(\boldsymbol{w}_{\cdot K}^{(l)})\widehat{A} \end{bmatrix} \in \mathbb{R}^{nK\times n}, \tag{3.9}$$

  for each $l = 1, \dots, F$. Before the concatenation, the matrices $\widetilde{W}^{(1)}, \dots, \widetilde{W}^{(F)}$ are reshaped as tensors in $\mathbb{R}^{nk\times 1\times n}$ (see Figure 3.2);
- *the operation $\widetilde{\mathbf{W}}^{\top}\mathrm{vertcat}(X)$ is a tensor–vector product (see Remark 3.2.6);*
- *$B \in \mathbb{R}^{n\times F}$ is the matrix of the biases, i.e., each column $\boldsymbol{b}_{\cdot l}$ is the bias vector corresponding to the l-th output feature of the nodes.*

Notation 3.2.5. From now on, for the sake of simplicity, for each matrix $X \in \mathbb{R}^{n\times K}$, denote by $\boldsymbol{x}$ the vector $\mathrm{vertcat}(X) \in \mathbb{R}^{nK}$.

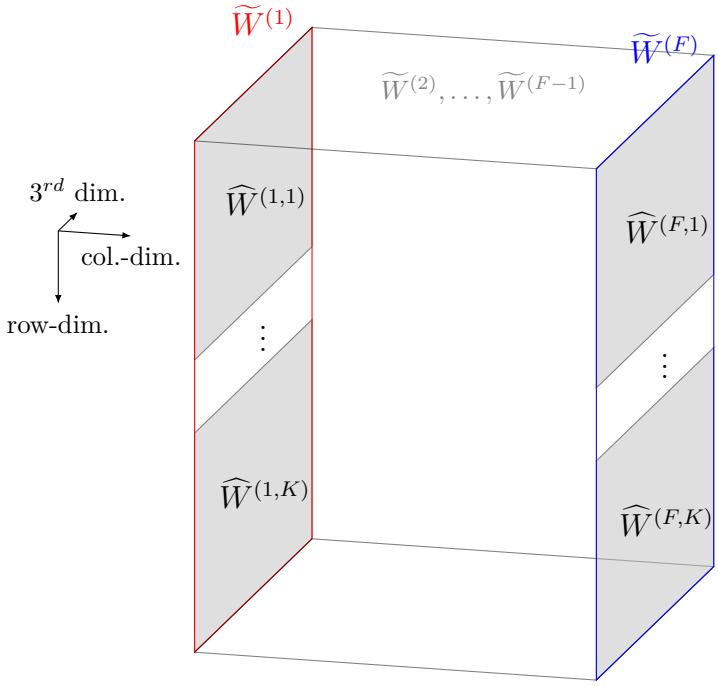

Figure 3.2: Tensor $\widetilde{\mathbf{W}}$ obtained concatenating along the second dimension of the matrices $\widetilde{W}^{(1)}, \ldots, \widetilde{W}^{(F)} \in \mathbb{R}^{nK \times n}$. Before the concatenation, the matrices are reshaped as tensors in $\mathbb{R}^{nk \times 1 \times n}$.

The generalization of (3.5) to the case of F output features is built as a function that, for each $X \in \mathbb{R}^{n \times K}$, a matrix is returned $Y \in \mathbb{R}^{n \times F}$ whose l-th column $\boldsymbol{y}_{\cdot l}$, for $l = 1, \ldots, F$, is defined as the application of (3.5) with respect to a proper filter $W^{(l)} \in \mathbb{R}^{n \times K}$. Indeed, given

$$\boldsymbol{y}_{\cdot l} = \boldsymbol{f}\left(\widetilde{W}^{(l)\top}\boldsymbol{x} + \boldsymbol{b}_{\cdot l}\right) \tag{3.10}$$

where $\boldsymbol{b}_{\cdot l} \in \mathbb{R}^n$ is the bias vector associated to the l-th filter, obtain

$$Y = \begin{bmatrix} \boldsymbol{y}_{\cdot 1} & \cdots & \boldsymbol{y}_{\cdot F} \end{bmatrix} =$$

$$\begin{bmatrix} \boldsymbol{f}\left(\widetilde{W}^{(1)\top}\boldsymbol{x} + \boldsymbol{b}_{\cdot 1}\right) & \cdots & \boldsymbol{f}\left(\widetilde{W}^{(F)\top}\boldsymbol{x} + \boldsymbol{b}_{\cdot F}\right) \end{bmatrix} = \boldsymbol{f}\left(\widetilde{\mathbf{W}}^{\top}\boldsymbol{x} + B\right).$$

Simply stated, the generalization to the F output features can be interpreted as a repetition of (3.5), with respect to F different filters and biases, grouping the results in a matrix Y.

Remark 3.2.6. Recall that the matrix-tensor product of a matrix $M \in \mathbb{R}^{p \times q}$ by a tensor $\mathbf{T} \in \mathbb{R}^{q \times r \times s}$ is given by

$$M \cdot \mathbf{T} = \mathbf{P} \in \mathbb{R}^{p \times r \times s},$$

where the (i, j, k)-th component p_{ijk} of the tensor $\mathbf{P}$ is defined as

$$p_{i\,j\,k} = \sum_{h=1}^{q} m_{i\,h} t_{h\,j\,k}\,,$$

where m_{ih} and t_{hjk} are components of M and $\mathbf{T}$, respectively. Analogously, it is possible to extend this product to tensor–matrix or tensor–tensor pairs.

Moreover, for a three-way tensor as $\widetilde{\mathbf{W}}$, recall that the transpose is defined such that the (i, j, k)-th element of $\widetilde{\mathbf{W}}^\top$ is equal to the (k, j, i)-th element of $\widetilde{\mathbf{W}}$.

Remark 3.2.7 (Total number of parameters). The total number of parameters in a GI layer, with a characterizing function (3.8), is $nKF + nF$, i.e., the number of weights plus the number of biases. Remember that the number of parameters of a fully-connected layer, with an input shape n and an output shape M is $nM + M$; then, in the case of $M = n$ and $(KF + F) < (n + 1)$, observe that the GI layers have a smaller number of parameters to be trained. This observation is important in the case of very large graphs G (i.e., $n \gg 1$).

3.2.3 GINNs compared to other Graph Neural Networks

To the best of the authors' knowledge, the GI layers introduced above define a novel typology of spatial GCNs. As stated in [40], the design and study of GNN layers is currently a vibrant area of research for deep learning, and a complete summary and comparison between different models is challenging and out of the scope of this chapter. Nonetheless, the GINN layer clearly belongs to Message Passing NNs, firstly defined by [80] and recalled in [40]. In addition, as already commented above, it belongs to the convolutional class of message passing NNs, like other previous works [53, 115, 190]. Formally, define as $\boldsymbol{h}^{(\ell)}$ the output vector of the ℓ-th layer of a NN architecture, where $\boldsymbol{h}^{(0)} = X \in \mathbb{R}^{n \times K}$ is the input feature matrix. As reported by [40], graph convolutional layers computes the component i corresponding to the node v_i of $\boldsymbol{h}^{(\ell+1)}$ as:

$$h_i^{(\ell+1)} = g_1(h_i^{(\ell)}, \bigoplus_{j \in \mathcal{N}(i)} c_{ij}\, g_2(h_j^{(\ell)})), \tag{3.11}$$

where g_1 and g_2 are learnable function; $\bigoplus$ is an aggregation operator over the neighborhood of i, like the summation; c_{ij} is a constant representing an importance measure of v_j to v_i. The above formulation is quite general, and it comprises the GINN formulation. Indeed, recalling the weight vector $\boldsymbol{w}$ of Equation (3.2), the GINN layer formulation can be stated in terms of Equation (3.11) by fixing $\bigoplus$ as the summation, $c_{ij} = 1$, and:

$$\begin{aligned} g_2(h_j^{(\ell)}) &= w_j^{(\ell)}\, h_j^{(\ell)} \\ g_1(h_i^{(\ell)}, \alpha) &= f(w_i^{(\ell)}\, h_i^{(\ell)} + \alpha) \quad \text{with a generic } \alpha \in \mathbb{R}, \end{aligned}$$

where f is the layer activation function. Differently from the GINN layer, the mentioned previous approaches [53, 115, 190] restate the convolution operator using the spectral representation of the graph, and specifically the spectral decomposition of the graph Laplacian associated with the graph G. As stated by [53], the spectral decomposition is introduced because of the need of faster computations of the graph convolutional operations.

A comparison that deserves special care is the one with NN4G, that is reputed the first GNN [191]. Indeed, Equation (3.1) partially resembles the NN4G layer (see [191, sec. V.B.] and [131]). Restated according to the formalism of this section, the NN4G characterizing function of a hidden layer ℓ, with $\ell \geq 2$, results in:

$$h_i^{(\ell)} = f(\sum_{s=1}^{K} \theta_s^{(\ell)} X_{is} + \sum_{\kappa=1}^{\ell-1} \sum_{j \in \mathcal{N}(i)} w_{ij}^{(\kappa,\ell)} h_i^{(\kappa)},$$

where X_{is} is the feature s attributed to the node i of the feature input matrix X and $\theta_s^{(\ell)}$ are learnable parameters. The first term links the input feature matrix with the actual layer ℓ, so it resembles the residual connections later revisited for the ResNet architecture [101]. The second summation is a form of skip connections, that is, the extra connections between non-consecutive layers, later revisited for the DenseNet architecture [103]. If $\kappa = \ell - 1$, the third summation resembles the formulation of the GINN as in Equation (3.3). In other words, it appears that the simpler GINN architecture can be rebuilt by removing from NN4G the mentioned forms of residual and skip connections. However, the formulation given in this chapter by Equation (3.1) is generalized to the tensor form (3.8), that is, to multiple input/output features, unlike the NN4G layers. It is worth noting that a tensor form, such as (3.8), is very useful to manage graph-structured regression problems with more than one feature per node.

Analogously, there are few similarities between the simple L^{GI} layer of Equation (3.1) and the diffusion-convolutional NNs (DCNNs) of [15], but these GCNs are still different from the GINNs. Indeed, DCNN layers are based on a degree-normalized transition matrix, computed from the adjacency matrix, which expresses the probability of going from node v_i to node v_j in one step [15], while GINN makes use of the adjacency matrix by itself to filter the connections between consecutive layers. Other similarities between the GINNs and GNNs models can be observed in for the works of [59, 60], where the adjacency matrix is used to describe the flow of information. Nonetheless, in [59], the NN is built connecting a set of simpler NNs, according to the adjacency matrix. In [60], the interconnected NNs are trained similarly to a physics-informed NN (see [155, 156]).

3.2.4 Additional Properties for GI Layers

The GI layers, in their general formulation (3.8), can be endowed with additional operations. As is commonly done for the convolutional layers, the possibility to

endow the GI layers with a pooling operation is added. However, this operation is different from the one typically used in convolutional layers. Indeed, define a pooling for GI layers that aggregates the information in the columns of the output matrix, i.e., the values of the F output features of each graph vertex. Given a *reducing* operation (e.g., the mean, the max, the sum, etc.), labeled as rdc, and applied to each row of the matrix returned by (3.8), the pooling operation for GI layers modifies (3.8) in the following way:

$$\mathcal{L}^{(GI\,;\,\mathrm{rdc})}(X) = \mathrm{rdc}\left(\boldsymbol{f}\left(\widetilde{\mathbf{W}}^{\top}\boldsymbol{x} + B\right)\right), \tag{3.12}$$

where rdc is applied row-wise. For example, let $Y \in \mathbb{R}^{n\times F}$ denote the argument of the pooling operation in (3.12), namely $Y = \boldsymbol{f}\left(\widetilde{\mathbf{W}}^{\top}\boldsymbol{x} + B\right)$; the max-pooling operation for a GI layer is such that:

$$\mathcal{L}^{(GI\,;\,\max)}(X) = \begin{bmatrix} \max\{y_{11}, \ldots, y_{1F}\} \\ \vdots \\ \max\{y_{n1}, \ldots, y_{nF}\} \end{bmatrix} \in \mathbb{R}^{n}\,. \tag{3.13}$$

Note that the pooling operation can be generalized to the application of sub-groups to filters, instead of to all the filters. In this case, the pooling operation returns a matrix $Y \in \mathbb{R}^{n\times F'}$, with $F' < F$.

Another operation that is defined for GI layers is the application of a mask on the graph, such that the layer returns values only for a subset $\{v_{i_1}, \ldots, v_{i_m}\}$ of the chosen nodes. Let $I = \{i_1, \ldots, i_m\} \subseteq \{1, \ldots, n\}$ label the subset of nodes on which output of the GI layer focuses on. Then, a GI layer with a mask operation defined by the set I returns a sub-matrix $Y' \in \mathbb{R}^{m\times F}$ of the matrix $Y \in \mathbb{R}^{n\times F}$ defined by (3.8), obtaining extracting rows with index in I; namely,

$$\mathcal{L}^{(GI\,;\,I)}(X) = \begin{bmatrix} \left(\boldsymbol{f}\left(\widetilde{\mathbf{W}}^{\top}\boldsymbol{x} + B\right)\right)_{i_1\,\cdot} \\ \vdots \\ \left(\boldsymbol{f}\left(\widetilde{\mathbf{W}}^{\top}\boldsymbol{x} + B\right)\right)_{i_m\,\cdot} \end{bmatrix} \in \mathbb{R}^{m\times F}\,. \tag{3.14}$$

Note that the mask operation does not use the adjacency matrix of the graph, and it is employed in the numerical experiments only in the output layer of the evaluated GINNs.

This section ends with the following proposition, characterizing the relationship between the input and the output features of the graph nodes with respect to a GINN with a subset of H consecutive GI layers. Before presenting the statement, briefly recall some useful definitions of graph theory.

Given an undirected graph $G = (V, E)$, two nodes v_i and v_j are said adjacent if $(v_i, v_j) \in E$. A path over the graph G between two nodes v_i and v_j is a sequence of

distinct nodes starting at v_i and ending in v_j, expressed as $(v_{p_0} = v_i, \dots, v_{p_S} = v_j)$, and such that for all $s = 0, \dots S-1$, v_s is adjacent to v_{s+1}. The length of the path is the length of the sequence minus 1, that is $S-1$. The shortest path between two nodes v_i and v_j is the path of least length between the same nodes. Then, in the proposition below, the distance between two nodes v_i, v_j in the graph G, denoted by $\text{dist}_G(v_i, v_j)$, is the length of the shortest path between the same nodes. In particular, if two nodes belongs to different connected components of the graph, their distance is said to be infinity. In addition, if the graph $G = (V, E)$ is directed, define the associated undirected graph $G_a = (V_a, E_a)$ such that $V_a = V$ and $E_a = \{(v_i, v_j) \mid (v_i, v_j) \in E \text{ or } (v_j, v_i) \in E\}$. The proposition below focuses on an undirected graph, but it can be generalized to a directed graph G via its associated undirected graph G_a. The proof of the statement is straightforward.

Proposition 3.2.8 (Number of consecutive GI layers and node interactions). *Consider an undirected graph G, with the associated adjacency matrix $A \in \mathbb{R}^{n \times n}$. Let $H \in \mathbb{N}$, $H \geq 1$ and consider a GINN with a subset of H consecutive GI layers $L_1^{GI}, \dots, L_H^{GI}$, built according to A and with L_h^{GI} connected to L_{h+1}^{GI}, for $h = 1, \dots, H-1$. Then, the input feature corresponding to node v_i in L_1^{GI} contributes to the computation of the output feature corresponding to v_j in L_H^{GI} if $H \geq \text{dist}_G(v_i, v_j)$.*

The proposition above introduces a dependency of a GINN's depth on the complexity of the graph $G = (V, E)$. Let $\boldsymbol{F} : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^m$ be a function defined on the n vertices of G, and returning a vector of m values associated with the vertices $v_{i_1}, \dots, v_{i_m} \in V$. Let $\widehat{\boldsymbol{F}} : \mathbb{R}^n \to \mathbb{R}^m$ be the characterizing function of a GINN that approximates $\boldsymbol{F}$. If the output feature of vertex $v_j \in \{v_{i_1}, \dots, v_{i_m}\}$ through $\boldsymbol{F}$ depends on the input feature of vertex v_i, then the GINN needs at least $\text{dist}_G(v_i, v_j)$ consecutive GI layers to guarantee that the input feature of v_i contributes in making predictions for the output feature of v_j.

3.3 Case study: Maximum Flow

This section describes the maximum-flow problem case study: given a flow network, that is, a graph with a source, a sink, and capacities defined on the edges, the goal is to find the maximum flow value that can reach the sink [116, Chapter 7.1]. In particular, the interest is on the stochastic maximum-flow problem, i.e. a problem where the edge capacities are modeled as random variables and the target is to find the distribution of the maximum-flow (e.g., see [58]).

The stochastic maximum-flow problem is a sufficiently general problem graph regression algorithm, and it has many interesting applications in network analyses, such as the circulation with demand (CwD) problems (see [116, Chapter 7.7] and network interdiction models (NIMs) [57]. Put simply, a CwD problem should identify whether the maximum flow satisfies a given demand, varying the supply

provided by the source and the capacities of the edges; a NIM describes a game in which one or more agents modify the edge capacities to minimize/maximize the maximum flow of the network. These models have many interesting real-world applications, such as the administration of city traffic, the optimization of goods distributions, and identifying vulnerabilities in an operational system.

In this section, the maximum-flow problem is briefly described (Section 3.3.1), in particular the stochastic maximum-flow problem is presented 3.3.2. Finally, the maximum-flow regression problem is illustrated (Section 3.3.3). The regression problem will be the basis for an experimental comparison between the GINN architecture presented in Section 3.2.

3.3.1 The Maximum-Flow Problem

Flow networks are useful models to describe transportation networks, i.e., networks where some sort of traffic flows from a source to a sink along the edges, using the nodes as switches to let the traffic move from an edge to another one (see [ch. 7.1] in [116]). Here, the definition of a flow network is briefly recalled.

Definition 3.3.1 (Flow Network). *A flow network $\mathcal{G} = (G, s, t, c)$ is a directed graph $G = (V, E)$, of nodes V and edges $E \subseteq V \times V$, such that:*

- *The two nodes $s, t \in V$, $s \neq t$, are defined as the source and the sink of the network, respectively;*
- *c is a real-valued non-negative function defined on the edges, $c : E \to \mathbb{R}_{\geq 0}$, assigning to each edge $e \in E$ a capacity $c_e := c(e)$.*

A flow network $\mathcal{G}$ can be endowed with a flow function.

Definition 3.3.2 (Flow). *Let $\mathcal{G}$ be a flow network. An s-t flow (or just flow) on $\mathcal{G}$ is a function*

$$\varphi : E \to \mathbb{R}_{\geq 0}$$

that satisfies the following properties:

- *The capacity condition: for each $e \in E$, it holds $0 \leq \varphi(e) \leq c_e$;*
- *The conservation condition: for each $v \in V \backslash \{s, t\}$, the amount of flow entering v must be equal to the amount of flow leaving v, i.e.,*

$$\sum_{e \in E_{in}(v)} \varphi(e) = \sum_{e \in E_{out}(v)} \varphi(e)\,, \quad \forall\, v \in V \setminus \{s, t\}\,,$$

where $E_{in}(v) \subset V$ is the subset of the incoming edges of v, and $E_{out}(v) \subset V$ is the subset of outcoming edges of v;

- *The amount of flow leaving the source s must be greater than, or equal to, the one entering s, i.e., $\sum_{e\in E_{in}(s)} \varphi(e) \leq \sum_{e\in E_{out}(s)} \varphi(e)$.*

For the sake of notation, for each $v \in V$ set

$$\varphi_{\text{in}}(v) = \sum_{e\in E_{\text{in}}(v)} \varphi(e), \quad \varphi_{\text{out}}(v) = \sum_{e\in E_{\text{out}}(v)} \varphi(e), \quad \Delta\varphi_v = \varphi_{\text{out}}(v) - \varphi_{\text{in}}(v)\,,$$

and call the flow value of a vertex v the quantity $\Delta\varphi_v$.

Then, due to the conservation condition, it holds that $\Delta\varphi_v = 0$, for each $v \in V \setminus \{s, t\}$, and $\Delta\varphi_s \geq 0$. Note that the flow value of the source s is equal to the opposite of the flow value of the sink t, i.e., $\Delta\varphi_t = -\Delta\varphi_s$; for this reason, $\Delta\varphi_s$ is referred to as the flow value of the network.

One of the most common issues concerning a flow network $\mathcal{G}$ is to find a flow that maximizes the effective total flow of the sink t, i.e., to find φ^*, such that

$$\varphi^* = \arg\max_{\varphi} |\Delta\varphi_t|\,.$$

Such a kind of problem is called maximum-flow problem, and it can be solved through linear programming or many other algorithms (e.g., [127, 83, 45, 113, 82]). From a practical point of view, the relationship between the maximum-flow problem and the minimum-cut problem on a flow network $\mathcal{G}$ is particularily important (see [ch. 7.2] in [116] for more details).

Remark 3.3.3 (Flow networks and undirected graphs). Definitions 3.3.1 and 3.3.2 can be extended to the more complicated case of undirected graphs. Indeed, as observed in Section 3.2, the undirected edges $\{v_j, v_i\}$ of a graph are equivalent to the two directed edges (v_j, v_i) and (v_i, v_j). Then, a flow network based on an undirected graph $G = (V, E)$ can be defined as a flow network $\mathcal{G} = (G', s, t, c)$, where $G' = (V, E')$ is a directed graph, such that $(u, v), (v, u) \in E'$ if $\{u, v\} \in E$, whose capacity is defined on the edges of G, i.e., $c : E \to \mathbb{R}_{\geq 0}$. As a result, a flow φ defined on such a flow network is a function $\varphi : E' \to \mathbb{R}_{\geq 0}$ characterized by a slightly different capacity condition; namely, φ is such that

$$0 \leq \varphi((u, v)) + \varphi((v, u)) \leq c(\{u, v\})\,, \quad \forall\ \{u, v\} \in E\,.$$

Another approach is to introduce an arbitrary ordering, denoted by "$<$", on the graph nodes and define a directed graph $G' = (V, E')$, such that $(u, v) \in E'$ if $\{u, v\} \in E$ and $u < v$. In this case, a flow φ on the flow network $\mathcal{G} = (G', s, t, c)$, with c defined on the edges E' is a function $\varphi : E' \to \mathbb{R}$, such that the capacity condition is

$$0 \leq |\varphi(e)| \leq c(e)\,, \quad \forall\ e \in E'\,,$$

and where the entering/exiting behavior of the flows is described by the sign of $\varphi(e)$ and not by the edge direction; i.e., for $(u, v) \in E'$, if $\varphi((u, v)) > 0$ the flow $\varphi((u, v))$ enters in v, whereas if $\varphi((u, v)) < 0$ then $\varphi((u, v))$ enters in u. This latter approach is mainly adopted by software implementations.

3.3.2 The Stochastic Maximum-Flow Problem

The idea of the flow network, flow, and the maximum-flow problem can be easily extended to a stochastic framework, in which edge capacities are modeled as random variables.

Definition 3.3.4 (Stochastic flow network). *A stochastic flow network $\mathcal{G} = (G, s, t, p)$ is a directed graph $G = (V, E)$ of nodes V and edges $E \subseteq V \times V$, such that:*

- *The two nodes $s, t \in V$, $s \neq t$, are defined as the source and the sink of the network, respectively;*
- *p is a real-valued non-negative probability distribution for the edge capacities of the network.*

Let $\mathcal{G}(\boldsymbol{c})$ denote the flow network (G, s, t, c) with edge capacities returned by c sampled from p. More specifically, let $e_1, \ldots, e_{|E|}$ be all the edges of G; then $\mathcal{G}(\boldsymbol{c}) = (G, s, t, c)$ if:

- $\boldsymbol{c} \in \mathbb{R}^{|E|}$ is a vector whose c_i is sampled from p;
- The function c is such that $c(e_i) = c_i$, for each $i = 1, \ldots, |E|$.

Denote by $\varphi^{(\boldsymbol{c})}$ a flow defined on the flow network $\mathcal{G}(\boldsymbol{c})$.
The stochastic maximum-flow problem consists of finding the flow

$$\varphi^{*(\boldsymbol{c})} = \arg\max_{\varphi^{(\boldsymbol{c})}} \left|\Delta\varphi_t^{(\boldsymbol{c})}\right|, \tag{3.15}$$

for each fixed vector $\boldsymbol{c}$.

Alternatively, in stochastic maximum-flow problems, one may seek the flow distribution and/or its moments, or the maximum flow value entering the sink t, i.e.,

$$\left|\Delta\varphi_t^{*(\boldsymbol{c})}\right| = \max_{\varphi^{(\boldsymbol{c})}} \left|\Delta\varphi_t^{(\boldsymbol{c})}\right|, \tag{3.16}$$

3.3.3 The Maximum-Flow Regression Problem

A maximum-flow regression problem, with respect to a given stochastic flow network, consists of finding a function that, for each capacity vector $\boldsymbol{c}$, returns an approximation of the maximum flow $\left|\Delta\varphi_t^{*(\boldsymbol{c})}\right|$ or an approximation of all the flows reaching the sink t.

Let $\mathcal{G}$ be a stochastic flow network of $n = |E|$ edges and, without a loss of generality, let $e_1, \ldots, e_m \in E$, $m \leq n$ be all the incoming edges of the sink t. Let $\boldsymbol{F} : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^m$ be a function, such that

$$\boldsymbol{F}(\boldsymbol{c}) = [\varphi^{*(\boldsymbol{c})}(e_1), \ldots, \varphi^{*(\boldsymbol{c})}(e_m)]^\top := \boldsymbol{\varphi}, \tag{3.17}$$

for each capacity vector $\boldsymbol{c} \in \Omega \subseteq \mathbb{R}^n$ with the elements sampled from the distribution p of the given network $\mathcal{G}$.

For the sake of simplicity, drop from now on the dependency from $\boldsymbol{c}$ and the star symbol from the elements of $\boldsymbol{\varphi}$, denoted by $\varphi_1, \ldots, \varphi_m$ the m elements of the vector $\boldsymbol{\varphi} = \boldsymbol{F}(\boldsymbol{c})$. Moreover, assuming the convention of the non-negative flow functions on the graph (see Section 3.3.1), denote by φ the ℓ_1-norm of $\boldsymbol{\varphi}$; specifically:

$$\sum_{j=1}^{m} \varphi_i = \sum_{j=1}^{m} |\varphi_i| = \| \boldsymbol{\varphi} \|_1 =: \varphi \,. \tag{3.18}$$

Then, the target maximum flow with respect to $\boldsymbol{c}$ coincides with φ; indeed, $|\Delta {\varphi_t}^{*(\boldsymbol{c})}| = \sum_{j=1}^{m} \varphi^{*(\boldsymbol{c})}(e_j) = \sum_{j=1}^{m} \varphi_i = \varphi$.

Given the target function $\boldsymbol{F}$ defined by (3.17), consider the maximum-flow regression problem with respect to $\mathcal{G}$ looking for an NN with a characterizing function $\widehat{\boldsymbol{F}} : \mathbb{R}^n \to \mathbb{R}^m$, such that $\widehat{\boldsymbol{F}}(\boldsymbol{c})$ approximates $\boldsymbol{F}(\boldsymbol{c})$ for each capacity vector $\boldsymbol{c}$. Namely, setting $\widehat{\boldsymbol{\varphi}} = \widehat{\boldsymbol{F}}(\boldsymbol{c})$ and $\boldsymbol{\varphi} = \boldsymbol{F}(\boldsymbol{c})$, seek $\widehat{\boldsymbol{\varphi}} \approx \boldsymbol{\varphi}$. To train an NN with respect to $\boldsymbol{F}$, build a dataset $\mathcal{D}$ of pairs $(\boldsymbol{c}\,, \boldsymbol{\varphi}) \in \mathbb{R}^n \times \mathbb{R}^m$, with $\boldsymbol{\varphi} = \boldsymbol{F}(\boldsymbol{c})$, where the capacity vectors are sampled with respect to the distribution p of $\mathcal{G}$; then, $\mathcal{D}$ is split into a training set $\mathcal{T}$, a validation set $\mathcal{V}$, and a test set $\mathcal{P}$ of arbitrary cardinalities.

Once an NN is trained, its regression performances are evaluated by computing two performance measures on the test set $\mathcal{P}$: the edge-wise average mean relative error (MRE_{av}), and the mean relative error on the predicted maxflow (MRE_{φ}). These two errors represent the mean relative error (weighted with respect to the true maxflow) of the predicted flows of the m edges $e_1, \ldots, e_m$ and the mean relative error of the predicted maxflow $\widehat{\varphi} := \sum_{i=1}^{m} \widehat{\varphi}_j$ (i.e., the sum of the elements of $\widehat{\boldsymbol{\varphi}} = \widehat{\boldsymbol{F}}(\boldsymbol{c})$), respectively. For each prediction $\widehat{\boldsymbol{\varphi}}$, let us denote

$$\mathbf{err}(\widehat{\boldsymbol{\varphi}}, \boldsymbol{\varphi}) = [\mathrm{err}_1(\widehat{\boldsymbol{\varphi}}, \boldsymbol{\varphi}), \ldots, \mathrm{err}_m(\widehat{\boldsymbol{\varphi}}, \boldsymbol{\varphi})]^\top := \left[\frac{|\widehat{\varphi}_1 - \varphi_1|}{\varphi}, \ldots, \frac{|\widehat{\varphi}_m - \varphi_m|}{\varphi} \right]^\top$$

as the vector of relative errors computed with respect to the true maxflow $\varphi = \sum_{j=1}^{m} \varphi_j$ (see (3.18)). Then, the performance measures MRE_{av} and MRE_{φ} are defined as

$$\mathrm{MRE}_{av}(\mathcal{P}) := \frac{1}{m} \sum_{j=1}^{m} \left(\frac{1}{|\mathcal{P}|} \sum_{(\boldsymbol{c},\boldsymbol{\varphi}) \in \mathcal{P}} \mathrm{err}_j(\widehat{\boldsymbol{\varphi}}, \boldsymbol{\varphi}) \right) \tag{3.19}$$

and

$$\mathrm{MRE}_{\varphi}(\mathcal{P}) := \frac{1}{|\mathcal{P}|} \sum_{(\boldsymbol{c},\boldsymbol{\varphi}) \in \mathcal{P}} \frac{|\widehat{\varphi} - \varphi|}{\varphi} \,, \tag{3.20}$$

respectively.

The smaller both the MRE_{av} and the MRE_{φ} values are on the test set, the better the performances of the NN, with respect to the maximum-flow regression task, are.

Remark 3.3.5 (Interpretation of MRE_{av} and MRE_{φ}). It is worth highlighting the different meanings of the errors (3.19) and (3.20): MRE_{av} describes the average quality of the NN in predicting the single elements $\varphi_1, \dots, \varphi_m$ of the target vector $\boldsymbol{\varphi}$, while MRE_{φ} describes the ability of the NN to predict a vector $\widehat{\boldsymbol{\varphi}}$, such that the corresponding maxflow $\widehat{\varphi} = \sum_{j=1}^{m} \widehat{\varphi}_j$ approximates the true maxflow $\varphi = \sum_{j=1}^{m} \varphi_j$. Therefore, a small MRE_{av} corresponds to a good approximation of the flow vectors (i.e., $\widehat{\boldsymbol{\varphi}} \approx \boldsymbol{\varphi}$) and a small MRE_{φ} corresponds to a good approximation of the maximum-flows (i.e., $\widehat{\varphi} \approx \varphi$). Nonetheless, it is important to point out that a small MRE_{av} does not necessarily imply a small MRE_{φ}, and vice-versa. For example, an NN with large MRE_{av}, characterized by the underestimation of the flows φ_{j_1} and by the overestimation of the flows φ_{j_2}, may return a small MRE_{φ} because the sum of the flows is not so far from the true maximum-flow; similarly, a large MRE_{φ} can be obtained from a sufficiently small MRE_{av} if, e.g., the NN underestimates or overestimates all the flows $\varphi_1, \dots, \varphi_m$ equivalently, such that $\widehat{\boldsymbol{\varphi}} \approx \boldsymbol{\varphi}$ but $\widehat{\varphi} \not\approx \varphi$.

Line Graphs for the Exploitation of GINN Models

Since the inputs of the target function $\boldsymbol{F}$ are the capacity vectors $\boldsymbol{c}$, which are defined on the edges of the graph G and not on the nodes, a preprocessing phase on the graph data is needed to compute the line graph L of G in order to exploit the GINN models for the maximum-flow regression problem. Here, for the ease of reading the definition of line graph is recalled (see [54, 84]).

Definition 3.3.6 (Line Graph). *Let $G = (V, E)$ be a graph (either directed or not). The line graph of G is a graph $L = (E, E')$, such that:*

- *The vertices of L are the edges of G;*
- *Two vertices in L are adjacent if the corresponding edges in G share at least one vertex.*

Given the line graph L of the graph G of a stochastic flow network $\mathcal{G}$, the adjacency matrix A_L of L is used to define NN models characterized by GI layers to perform the maximum-flow regression task. See the next section for more details about the GINN architectures that are built.

3.4 Experiments

In this section, experiments are conducted to show the performances of the proposed architecture named GINNs with respect to Multi-Layer Perceptrons (MLPs). The case studies analyzed are the regression on graph data. The first case study comes from the stochastic maximum-flow, as detailed in Section 3.3.2, and the results are reported in Section 3.4.1. The second case study comes from the flux regression on the DFNs, as presented in Section 2.2.3, and the results are reported

in Section 3.4.4. In particular, the results on the stochastic maximum-flow show the generalities of the proposed architecture, while the DFNs regression is aimed to show the potentialities of using GINNs in practical applications related to realistic underground flow simulations.

3.4.1 Maximum-Flow Numerical Experiments

The experiments related to the maximum-flow regression problem take into account two stochastic flow networks:

- $\mathcal{G}_1 = (G_{\text{BA}}, s, t, p)$. The graph G_{BA} of $\mathcal{G}_1$ characterizes a flow network built on an extended Barabási–Albert (BA) model graph [21, 8]. Put simply, an extended BA model graph is a random graph generated using a preferential attachment criterion. This family of graphs describes a very common behavior in many natural and human systems, where few nodes are characterized by a higher degree if they are compared to the other nodes of the network.

  In particular, the function *extended_barabasi_albert_graph* of the NetworkX Python module [94] generates an extended BA undirected graph , with input arguments $\texttt{n} = 50, \texttt{m} = 2, \texttt{p} = 0.15$, and $\texttt{q} = 0.35$); then, denote t (the sink of the network) as the node with the highest betweenness centrality [73] and add a new node s (the source of the network) connected to the 10 nodes with smallest closeness centrality [22, 164]. With these operations, it is possible to obtain a graph G_{BA} of 51 nodes and $n = |E| = 126$ edges, where the source s is connected to the 10 nodes and the sink t is connected to the $m = 15$ nodes (see Figure 3.3-left).

  In the end, since, in real-world applications, truncated normal distributions seem to be very common (see Remark 3.4.1), in order to simulate a rather general maximum-flow regression problem, the choice was a truncated normal distribution between 0 and 10, with a mean of 0, and a standard deviation of 5/3, as a probability distribution p for the edge capacities (see Section 3.3.2); i.e.,

$$c_i \sim p \equiv \mathcal{N}_{[0,10]}(5\,,\, 5/3)\,, \quad \forall\ i = 1, \ldots, n\,. \tag{3.21}$$

- $\mathcal{G}_2 = (G_{\text{ER}}, s, t, p)$. The graph G_{ER} of $\mathcal{G}_2$ characterizes a flow network built on an Erdős-Rényi (ER) model graph [67, 79]. Put simply, an ER model graph is a random graph generated with a fixed number of nodes, where the edge $e_{ij} = (v_i, v_j)$ has a fixed probability of being created. This family of graphs is typically used to prove and/or find new properties that hold for almost all the graphs; for this reason, our experiments consider a stochastic flow network based on an ER graph.

  In particular, the function `fast_gnp_random_graph` of the NetworkX Python module [94] generates an ER undirected graph, with input arguments $\texttt{n} =$

$200, \mathtt{p} = 0.01$). Proceeding further, its largest connected component G_0 is selected, in terms of the number of vertices. Then, to G_0 two new nodes are added: a node s (the source of the network) connected to all the nodes with degree equal to 1, and a node t (the sink of the network) connected to the 15 most distant nodes from s. These operations allow obtaining a graph G_{ER} of 171 nodes and $n = |E| = 269$ edges, where the source s is connected to 37 nodes and the sink t is connected to $m = 15$ nodes (see Figure 3.3-right).

In the end, the truncated normal distribution (3.21) is the probability distribution p for the edge capacities.

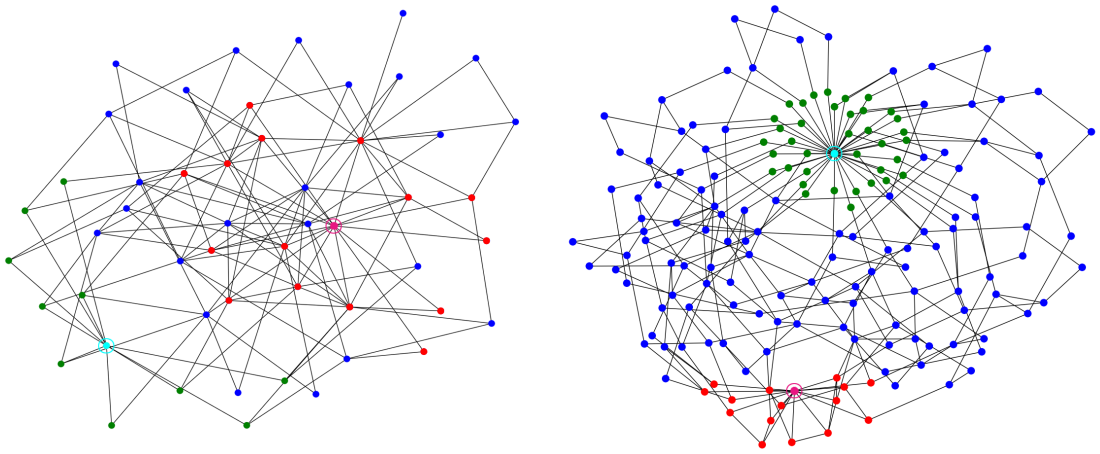

Figure 3.3: Graph G_{BA} of $\mathcal{G}_1$ (**left**) and graph G_{ER} of $\mathcal{G}_2$ (**right**). In cyan, and with a circle around the source s, in magenta and with a circle around the sink t, in green the nodes connected to s, and in red the nodes connected to t. All the other nodes are in blue.

Remark 3.4.1 (Regarding the truncated normal distribution for capacities). In a network describing a system of highroads, the capacity of a road is defined as $c = k\ell/S$ [157], where $k \in \mathbb{R}^+$ is a value depending on the type of the road, ℓ is the road length, and S is the average distance between two vehicles, typically chosen as a constant value. Then, assuming a network with all roads of the same type (i.e., k constant) and a truncated normal distribution for the length ℓ of the roads, the capacity can be modeled as a random variable with a truncated normal distribution. Therefore, generalizing the concept of the highroad capacity to other similar problems (e.g., a network of pipes, a communication network, etc.) the distribution (3.21) can be considered sufficiently generic for the numerical experiments of this section.

Given the stochastic flow networks $\mathcal{G}_1$ and $\mathcal{G}_2$, the corresponding maximum-flow regression problems consist of the approximation of the functions $\boldsymbol{F}_1 : \mathbb{R}^n \to \mathbb{R}^m$,

$n = 126$, $m = 15$, and $\boldsymbol{F}_2 : \mathbb{R}^n \to \mathbb{R}^m$, $n = 269$, $m = 15$, respectively, where $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$ are defined as in (3.17). For each $i = 1{,}2$, the dataset $\mathcal{D}_i$ of $\mathcal{G}_i$ is made of 10,000 pairs $(\boldsymbol{c}\,,\, \boldsymbol{\varphi} = \boldsymbol{F}_i(\boldsymbol{c})) \in \mathbb{R}^n \times \mathbb{R}^m$, where 3000 of them are used as the test set ($\mathcal{P}_i$) and the remaining 7000 pairs are used to generate the multi-set Θ_i sampling $\vartheta \in \{1, \ldots, 7000\}$ pairs. In particular, 80% of the pairs in Θ_i are used as the training set ($\mathcal{T}_i$) and the remaining 20% are used as the validation set ($\mathcal{V}_i$). An important aspect of our numerical experiments consists of analyzing the performance of the trained NNs, varying the quantity of available data for the training process (i.e., ϑ), and not only varying the hyper-parameters related to the architecture and optimization method; in particular, the study focuses on the NN performances when the number of training and validation data is $\vartheta = 7000, 1000, 500$. Indeed, in real-world problems, the amount of available data can be limited for many reasons (e.g., limited computational resources for simulations, limited time for measurements, etc.). Then, studying the performances of a regression model while decreasing ϑ is important to understand the sample efficiency of the model.

The dataset fluxes $\boldsymbol{F}_i(\boldsymbol{c}) = \boldsymbol{\varphi}$ are computed using the `maximum_flow` NetworkX function that, specifically, allows the computation of the flows for all the edges of the network (given the capacities $\boldsymbol{c}$). Then, considering all the 10 000 simulations executed to build the dataset $\mathcal{D}_i$, and denoting $\ell^{(i)}_{\max}(\boldsymbol{c})$ the length of the longest source-sink path in $\mathcal{G}_i(\boldsymbol{c})$, observe that:

1. $\ell_{\min}(\mathcal{G}_1) = 4$, $\ell_{\text{av}}(\mathcal{G}_1) \approx 5.5$, and $\ell_{\max}(\mathcal{G}_1) = 9$;
2. $\ell_{\min}(\mathcal{G}_2) = 7$, $\ell_{\text{av}}(\mathcal{G}_2) \approx 10.7$, and $\ell_{\max}(\mathcal{G}_2) = 17$;

where $\ell_{\min}(\mathcal{G}_i), \ell_{\text{av}}(\mathcal{G}_i)$, and $\ell_{\max}(\mathcal{G}_i)$ are the minimum, the average, and the maximum lengths, respectively, of the longest source-sink path of the flow for $\mathcal{G}_i$ with respect to $\mathcal{D}_i$, i.e.,

$$\ell_{\min}(\mathcal{G}_i) := \min_{(\boldsymbol{c},\boldsymbol{\varphi})\in\mathcal{D}_i} \ell^{(i)}_{\max}(\boldsymbol{c})\,, \tag{3.22}$$

$$\ell_{\text{av}}(\mathcal{G}_i) := \frac{1}{|\mathcal{D}_i|} \sum_{(\boldsymbol{c},\boldsymbol{\varphi})\in\mathcal{D}_i} \ell^{(i)}_{\max}(\boldsymbol{c})\,, \tag{3.23}$$

and

$$\ell_{\max}(\mathcal{G}_i) := \max_{(\boldsymbol{c},\boldsymbol{\varphi})\in\mathcal{D}_i} \ell^{(i)}_{\max}(\boldsymbol{c})\,. \tag{3.24}$$

The values reported in items 1 and 2 show that, on the average, in a radius of the length $\ell_{\text{av}}(\mathcal{G}_i)$ from the sink t, it is likely to find almost all the nodes characterizing the maximum-flow of the network $\mathcal{G}_i$. This information is then taken into account while choosing the depth values for the construction of the GINNs in the following Section 3.4.2. Indeed, recall that the number of consecutive GI layers in an NN tells if the input feature of node v_i contributes to the computation of the output feature of node v_j (see Proposition 3.2.8). Therefore, it is interesting to verify if the regression performance of a GINN improves or not, when the number of GI layers is related to one of the quantities (3.22)–(3.24).

3.4.2 NN Architectures, Hyper-Parameters, and Training

The numerical experiments of this section studies and compares the performances of MLPs and GINNs concerning the maximum-flow regression problems related to $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$. Then, the two archetypes of NN architectures are considered: an MLP archetype and a GINN archetype.

- **MLP Archetype:** The NN architecture is characterized by one input layer L_0, $H \in \mathbb{N}$, hidden layers $L_1, \ldots, L_H$ with a nonlinear activation function f, and one output layer L_{H+1} with a linear activation function. The output layer is characterized by m units, while all the other layers are characterized by n units. Finally, it is applied a batch normalization [108] before the activation function for each hidden layer $L_1, \ldots, L_H$. See Figure 3.4.

- **GINN Archetype:** The NN architecture is characterized by one input layer L_0 of n units, $H \in \mathbb{N}$ hidden GI layers $L_1^{GI}, \ldots, L_H^{GI}$ with a nonlinear activation function f, and one output layer L_{H+1}^{GI} with a linear activation function. All the GI layers are built with respect to the adjacency matrix $A_L \in \mathbb{R}^{n \times n}$ of the line graph of the network (see Section 3.3.3) and they are characterized by $F \in \mathbb{N}$ filters (i.e., output features). Then, the number of input features K of the GI layer L_h^{GI} is $K = F$, if $h > 1$, and $K = 1$, if $h = 1$. As for the MLP archetype, it is applied a batch normalization before the activation function of each hidden layer. Finally, the output layer is characterized by a pooling operation and by the application of a mask (see Section 3.2.4) to focus on the m units corresponding to the m target flows. See Figure 3.5.

Given the two NN archetypes above, the experiments start building a set of untrained NN models, varying the main hyper-parameters of the architectures. In particular, for the MLPs, the hyper-parameters H and f are varied (i.e., the depth and activation functions of the hidden layers), while for the GINNs, also F is varied (i.e., the number of filters of the GI layers) and the pooling operation. Specifically, the hyper-parameters vary among these values:

- **MLP archetype.** $f \in \{\text{relu, elu, swish, softplus}\}$ and $H \in \{2,3,4,5\}$. Deeper MLPs are not employed to avoid the so-called degradation problem [101], i.e., the problem in which increasing the number of hidden layers causes the performance of an NN to saturate and degrade rapidly.

- **GINN archetype.** $f \in \{\text{relu, elu, swish, softplus}\}$, $F \in \{1,5,10\}$, and pooling operations in $\{\max, \text{mean}\}$ (only if $F = 5,10$); $H \in \{3,5,7,9\}$ for $\mathcal{G}_1$ and $H \in \{4,9,14,19\}$ for $\mathcal{G}_2$. In particular, these values of H are selected because they are a discrete interval around the value $\ell_{\text{av}}(\mathcal{G}_i)$, also including cases near, or equal to, the minimum and maximum values $\ell_{\min}(\mathcal{G}_i)$ and $\ell_{\max}(\mathcal{G}_i)$, respectively.

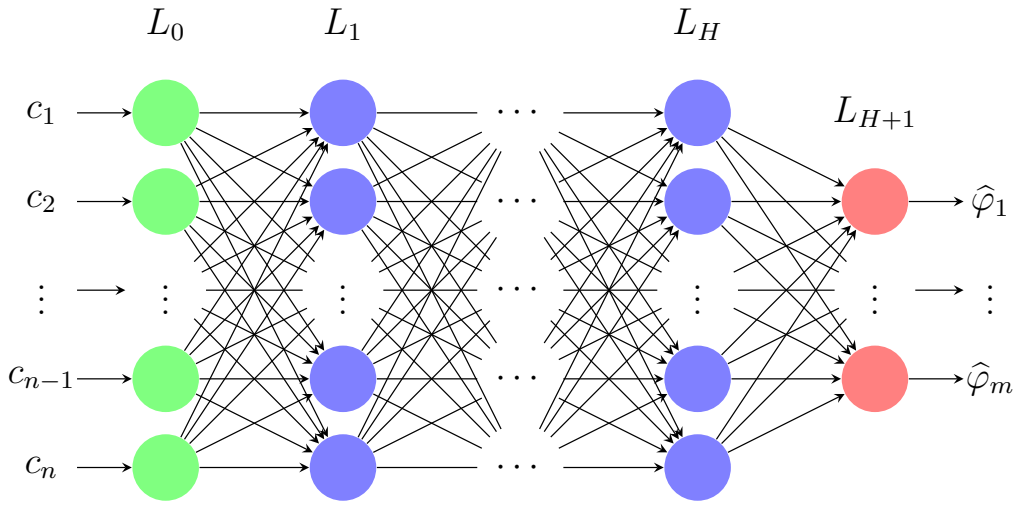

Figure 3.4: MLP archetype. The units of the input layer L_0 are in green, the units of the hidden layers $L_1, \ldots, L_H$ are in purple, and the units of the output layer L_{H+1} are in red.

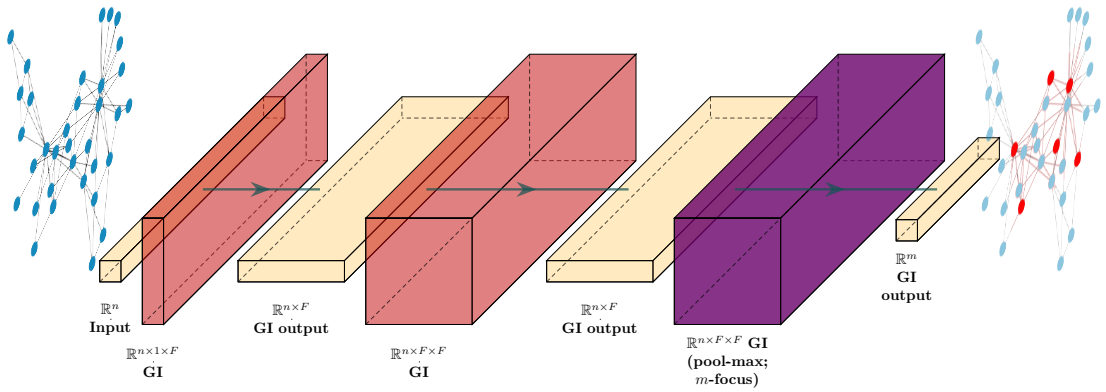

Figure 3.5: Example of a GINN archetype with depth $H = 2$ and max-pooling operation for the output layer. The output matrices Y of the NN layers are in orange, the weight tensors $\mathbf{W}$ of the hidden GI layers are in red (see Definition 3.2.4), and the weight tensor $\mathbf{W}$ of the output GI layer with max-pooling and masking operations (see Section 3.2.4) are in purple.

Then, these models are all trained on $\vartheta = 7000{,}1000{,}500$ input–output pairs sampled from $\mathcal{D}_i - \mathcal{P}_i$, using a mini-batch size $\beta = 128{,}64{,}32$; the weight initialization is a Glorot normal distribution [81] for the MLPs and it varies among a Glorot normal and a normal distribution $\mathcal{N}(0, 0.5)$ for the GINNs. All the biases are initialized as zeroes.

The remaining training options are fixed and shared by all the models during the training. In particular, these options are:

- Mean square error (MSE) loss, i.e.,

$$\text{loss}(\mathcal{B}) := \frac{1}{m}\sum_{j=1}^{m}\left(\frac{1}{|\mathcal{B}|}\sum_{(\boldsymbol{c},\boldsymbol{\varphi})\in\mathcal{B}}(\widehat{\varphi}_j - \varphi_j)^2\right), \tag{3.25}$$

  where $\mathcal{B}$ is any generic batch of input–output pairs;

- The Adam optimizer [114] (learning rate $\epsilon = 0.002$, moment decay rates $\rho_1 = 0.9$, $\rho_2 = 0.999$);

- Early stopping regularization [86, 128] (200 epochs of patience, restore best-weights), to avoid overfitting;

- Learning rate reduction on plateau [128] (reduction factor $\alpha = 0.5$, 100 epochs of patience, minimum learning rate $\epsilon = 10^{-6}$).

The training of all the NN models, with respect to all the different training configurations, returns 3168 trained NNs; in particular, there are 144 MLPs and 1140 GINNs, for each stochastic flow network $\mathcal{G}_1$ and $\mathcal{G}_2$.

3.4.3 Performance Analysis of Maximum-Flow Regression

To evaluate the performance of an NN trained with respect to the maximum-flow regression task, consider the errors MRE_{av} and MRE_{φ} (see Section 3.3.3, and Equations (3.19) and (3.20), respectively) measured on the test set. In particular, to better analyze the performances, the NNs are visualized as points in the $(\text{MRE}_{av}, \text{MRE}_{\varphi})$ plane (see Figure 3.6). Then, the nearer a point is to the origin (i.e., the ideal zero-error NN), the better the regression performances of the corresponding NNs are. This representation is motivated by the characteristics of the errors reported in Remark 3.3.5. Indeed, it is important to analyze the behavior of the NNs with respect to MRE_{av} and MRE_{φ} together.

Consider the first stochastic flow network $\mathcal{G}_1$. In general, looking at Figure 3.6, the GINNs have better regression performances than the MLPs. In particular:

1. The MRE_{φ} of the GINNs is generally smaller than the MLPs, and this effect increases with ϑ;

2. The MRE_{av} of the GINNs is almost always smaller than the MLPs, and this effect seems to be almost stable while varying ϑ;

3. Looking at the hyper-parameter F, observe that the cases with $F = 1{,}10$ generally perform better with fewer training samples (i.e., $\vartheta = 1000{,}500$) while the cases with $F = 5$ generally perform better with $\vartheta = 7000$. This phenomenon suggests that increasing the number of filters can improve the quality of the training, even if a clear rule for the best choice of F is not apparent.

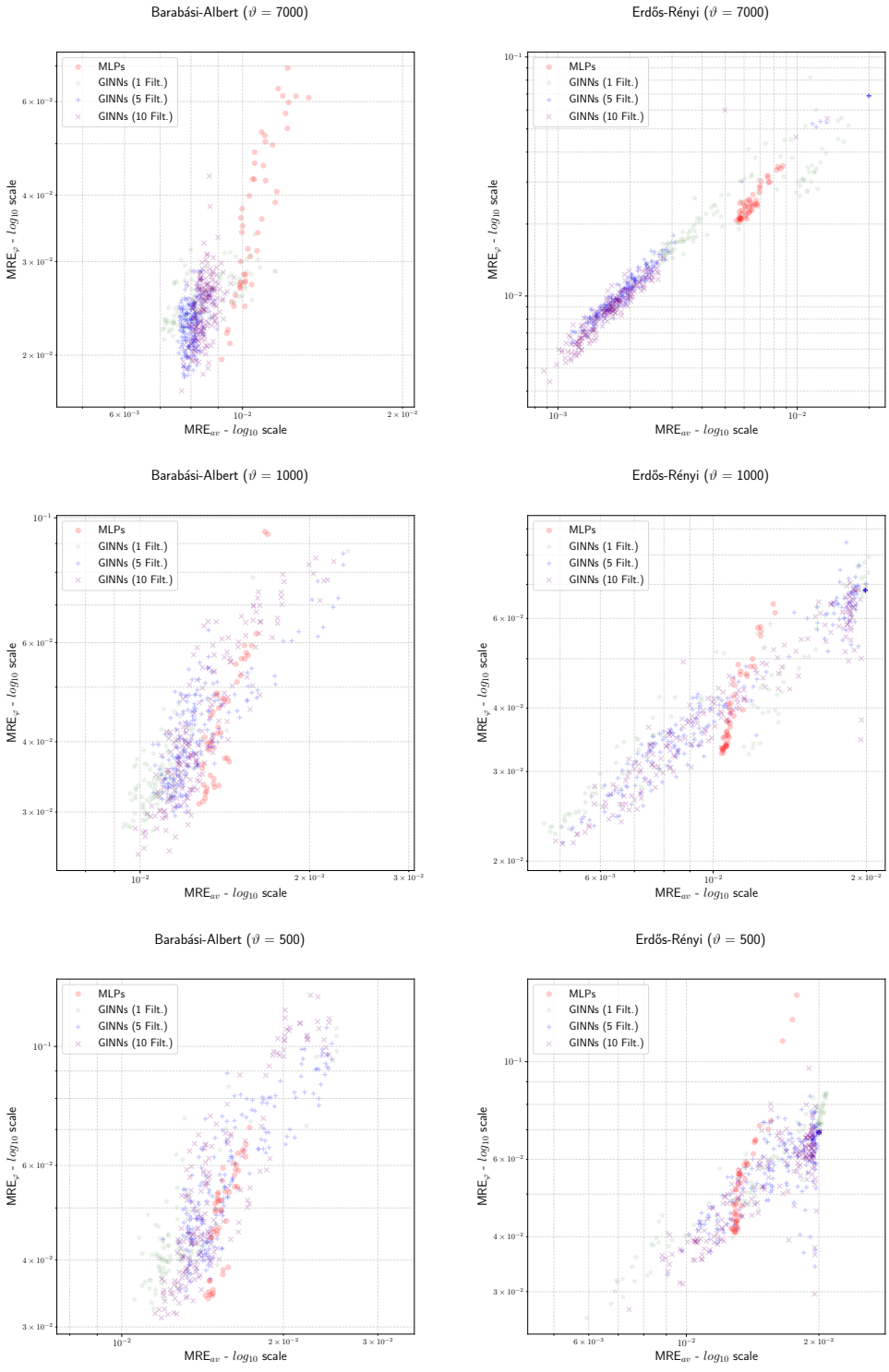

Figure 3.6: Network $\mathcal{G}_1$ **(left)** and $\mathcal{G}_2$ **(right)**. Scatter plots in the $(\text{MRE}_{av}, \text{MRE}_{\varphi})$ plane. Left to right: NNs trained with ϑ = 7000,1000,500 samples. Red circles: MLPs; green stars, blue crosses and purple "x": GINNs with F = 1,5,10, respectively.

The analysis continues with the second stochastic flow network $\mathcal{G}_2$, increasing the size and complexity of the flow network. Indeed, the graph G_{BA} of $\mathcal{G}_1$ is characterized by a reduced complexity of the maximum-flow problem, because the BA graphs are generated using a preferential attachment criterion that keeps the average length of the maximum source-sink path $\ell_{\text{av}}(\mathcal{G}_1)$ small, even when increasing the nodes of the graph (this phenomenon was observed during some preliminary experiments).

Figure 3.6 (right), highlights the same characteristics observed for $\mathcal{G}_1$, but much more emphasized. In particular, for each $\vartheta = 7000{,}1000{,}500$, the GINNs generally outperform the MLPs, especially with respect to the MRE_{av}. The reason for these similarities probably lies in the nature of the graphs G_{BA} and G_{ER} of $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively; indeed, the ER graphs are used to represent generic graphs and are typically used to show properties that hold for almost all the graphs. On the other hand, the graph G_{BA} is simpler than G_{ER}. Then, it is reasonable that the observations made for $\mathcal{G}_1$ are confirmed looking at $\mathcal{G}_2$ and it is reasonable that the performance differences observed in $\mathcal{G}_2$ are less emphasized in $\mathcal{G}_1$, since the maximum-flow problem on G_{BA} is less complex than on G_{ER}.

Remark 3.4.2 (GINNs, small MRE_{av}, and regressions on graphs). The above discussion shows that the GINNs generally perform better than MLPs for regression tasks on graphs but, focusing on the MRE_{av} values, the GINNs clearly show better performances (see Figure 3.6 and Tables 3.1 and 3.2). Specifically, Tables 3.1 and 3.2 show the three GINNs and MLPs with lowest MRE_{av} value on the test sets of $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively. The better performances of the GINNs, with respect to this error measure, are particularly important if extending the regression problem, e.g., if the task is translated to learning all the flow values $\varphi^{*(\boldsymbol{c})}(e_1), \ldots, \varphi^{*(\boldsymbol{c})}(e_n)$ on the edges of the graph and not only the ones characterizing the maximum-flow φ reaching the sink. Indeed, in this case, an NN with small errors on the single elements of the target vector is fundamental, while an NN with small errors on the sum of the elements of the target vector is useless; for this reason, in vector-valued regression tasks, the choice is of loss functions such as (3.25) (evidently similar to the performance measure MRE_{av}). In conclusion, the experiments provides clues about the great potential of GINNs in the field of regression graphs.

Below the performance analysis proceeds, and it is presented the analysis of the NN errors change when varying the hyperparameters.

The first observation is related to the activation functions and the mini-batch size. Observe that the trained NN models (both GINNs and MLPs) generally exhibit worse regression performances with a ReLU activation function and a mini-batch size equal to 128; Figure 3.7 report the same scatterplots of Figures 3.6, but without the points corresponding to the NNs with the ReLU activation function

or a mini-batch size equal to 128. It is worth noting that the general observations concerning the NN performances are even more evident when removing these models. Moreover, among the remaining models, there are no activation functions or mini-batch size values that are evidently better than the others; in general, as can be expected, there are only a few advantages of using a mini-batch size of 32 samples instead of a mini-batch size of 64 samples, while decreasing ϑ.

From now on, the analyses ignore models characterized by a ReLU activation function or mini-batch size equal to 128. Moreover, the focus is on the GINN models and, in particular, on their performances with respect to the hyperparameter H, characterizing the number of hidden layers. Indeed, the remaining hyperparameters (pooling operations and weight initializations) do not seem to have a particular impact on the results.

The study of the GINN performances with respect to H is particularly interesting, provided Proposition 3.2.8. In fact, from this proposition, the GINN models are expected to have a better performance if the depth H is such that $H+1 \gtrsim \ell_{\text{av}}(\mathcal{G}_i)$. This guess is indeed satisfied. Specifically, for $\mathcal{G}_1$, by increasing the depth H, observe that the GINNs improve their performances in general (see Figure 3.8). On the other hand, for $\mathcal{G}_2$, there is a slightly different behavior. The GINNs that are sufficiently deep (i.e., $H \geq 9$) show better performances than the GINNs with $H = 4$, but their errors tend to increase with a small ϑ; in particular, the more H is greater than $\ell_{\text{av}}(\mathcal{G}_2)$, the more the GINN performances seem to deteriorate (see Figure 3.9). To summarize, the depth in a GINN model is very important to obtaining good regression abilities, keeping in mind Proposition 3.2.8. Nonetheless, the practice of using as much of a GI layer as possible is not always the best choice, and this topic deserves attention in future work.

Table 3.1: Network $\mathcal{G}_1$. Top three GINNs and MLPs, for $\vartheta = 7000, 1000, 500$. Models are sorted with respect to the MRE_{av} error; the "rank" column describes their global position with respect to all the other models.

$\mathcal{G}_1$	GINNs								MLPs				
ϑ	**Rank**	MRE_{av}	H	F	f	β	**Pool.**	**Init.**	**Rank**	MRE_{av}	H	f	β
7000	1/528	0.00707	9	1	elu	64	-	G.Norm.	446/528	0.00914	3	swish	32
	2/528	0.00712	9	1	elu	128	-	Norm.	453/528	0.00934	3	swish	64
	3/528	0.00713	9	1	elu	32	-	Norm	455/528	0.00939	4	swish	32
1000	1/528	0.00933	9	1	softplus	32	-	G.Norm.	338/528	0.01272	5	softplus	32
	2/528	0.00940	7	1	elu	32	-	Norm.	353/528	0.01289	4	elu	32
	3/528	0.00952	7	1	elu	32	-	G.Norm.	357/528	0.01292	5	elu	32
500	1/528	0.01056	7	1	softplus	32	-	Norm.	263/528	0.01433	5	softplus	32
	2/528	0.01077	5	1	relu	32	-	G.Norm.	279/528	0.01448	5	softplus	64
	3/528	0.01092	7	1	softplus	32	-	G.Norm.	284/528	0.01452	4	softplus	32

Table 3.2: Network $\mathcal{G}_2$. Top three GINNs and MLPs, for ϑ = 7000,1000,500. Models are sorted with respect to the MRE_{av} error; the "rank" column describes their global position with respect to all the other models.

$\mathcal{G}_1$	GINNs								MLPs				
ϑ	**Rank**	MRE_{av}	H	F	f	β	**Pool.**	**Init.**	**Rank**	MRE_{av}	H	f	β
7000	1/528	0.00087	14	10	softplus	32	max	G.Norm.	442/528	0.00557	2	elu	32
	2/528	0.00092	14	10	softplus	32	mean	G.Norm.	445/528	0.00571	5	softplus	64
	3/528	0.00099	12	10	softplus	32	mean	G.Norm	446/528	0.00573	5	elu	32
1000	1/528	0.00465	9	1	elu	32	-	G.Norm.	263/528	0.01038	3	softplus	32
	2/528	0.00478	19	1	elu	32	-	G.Norm.	264/528	0.01038	2	swish	32
	3/528	0.00479	14	1	softplus	32	-	Norm.	267/528	0.01043	3	softplus	64
500	1/528	0.00593	14	1	elu	32	-	G.Norm.	114/528	0.01266	2	swish	32
	2/528	0.00674	14	1	softplus	32	-	G.Norm.	118/528	0.01272	2	swish	64
	3/528	0.00688	19	1	softplus	32	-	G.Norm.	119/528	0.01272	5	softplus	32

Remark 3.4.3 (Training time). Along the conducted experiments , the average training time for the GINN models is approximately 20 min in total, and one second per epoch; on the other hand, the average training time for the MLP models is approximately 10 min in total and half a second per epoch. Nonetheless, it can be pointed out that the difference in training times between GINNs and MLPs can be reduced with code optimization. Indeed, the GINN layers are a custom class of TensorFlow NN layers developed on purpose by the authors for numerical experiments. The code of TensorFlow's FC layers is extremely optimized. Therefore, at the present time, the GINNs and MLPs are not equally comparable as far as computational cost is concerned. More details about the average training times per epoch of the models are reported in Table 3.3, and this quantity is indicative of the training computational cost of the NNs. However, recall that the experiments take into account more than three thousand models, each one with a different training configuration that characterizes the training time per epoch. All the training was performed on a workstation with a CPU of 4 Core and 8 Threads, 32 GB of RAM, and a GPU Nvidia 1080 8 GB.

Table 3.3: Global statistics of the average training time per epoch for GINN and MLP models, expressed in seconds.

	Avg. Time per Epoch (s)	
	GINNs	**MLPs**
Mean	1.099	0.565
Std	1.359	0.292
25th perc.	0.318	0.380
50th perc.	0.567	0.431
75th perc.	1.296	0.632

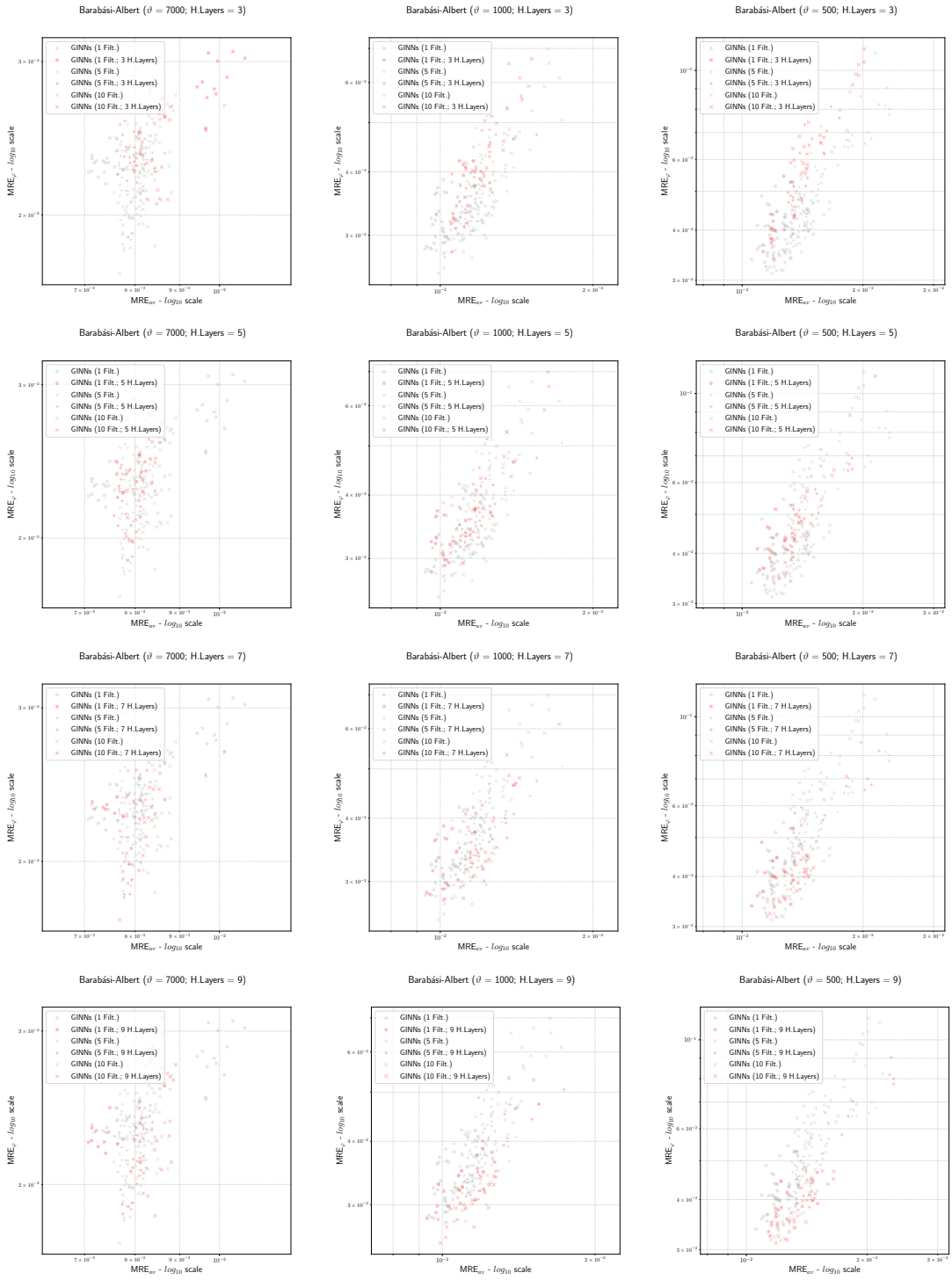

Figure 3.8: Scatter plots in the $(\text{MRE}_{av}, \text{MRE}_{\varphi})$ plane for GINNs trained with respect to $\mathcal{G}_1$. Left to right: GINNs trained with ϑ = 7000,1000,500 samples; top to bottom: red markers highlight GINNs with H = 3,5,7,9 (black markers for all the other models).

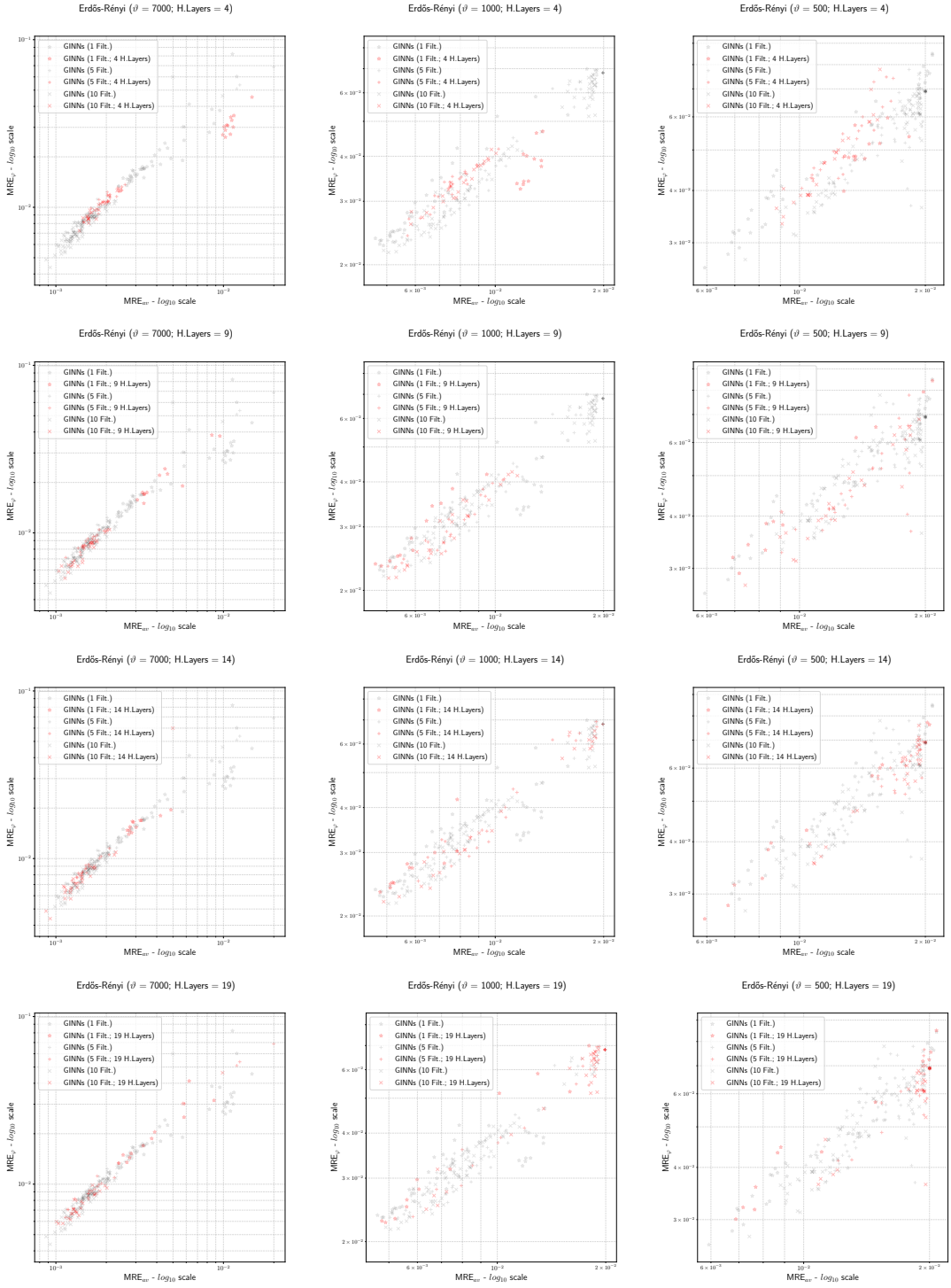

Figure 3.9: Scatter plots in the $(\mathrm{MRE}_{av}, \mathrm{MRE}_{\varphi})$ plane of the GINNs trained with respect to $\mathcal{G}_2$. From left to right, the GINNs are trained using $\vartheta = 7000,1000,500$ samples; from top to bottom, the red markers highlight the GINNs with hyper-parameters of $H = 4,9,14,19$ (black markers for all the other models).

3.4.4 Performance Analysis of DFN Flux Regression

Section 3.4.1 has showed the regression abilities and the potentialities of the GINN models for the maximum-flow regression problem, i.e., for a problem representative of generic real-world applications. In this section, the focus is on a specific real-world application where GINNs can be useful; in particular, it considers an uncertainty quantification (UQ) problem related to underground flows in fractured media. Here, the idea is to take advantage of the DFN's graph structure to build GINN models and to analyze the advantages of using such models for the DFN flux regression tasks instead of the classic NN architectures (e.g., MLPs or multitask NNs).

The numerical experiments and analyses presented in what follows are based on the example DFN158 described in Section 2.2.1. Then, the same experimental setting studies and compares the performances of the MLPs and GINNs concerning the flux regression problem related to the function $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^m$ described in Section 2.2.3. The two archetypes of MLP and GINN architectures are the same as in Section 3.4.2, as are most of the hyperparameter values and the training options used; the only differences are the following:

- $\vartheta = 1000{,}500$ (number of training and validation data);
- $\beta = 64{,}32$ (mini-batch size);
- The ReLU activation function is not considered in the experiments;
- For the GINN models, the depth parameter values are $H \in \{4{,}7{,}9{,}14{,}19\}$. The rationale behind this choice is that it is a set of values around 8, which is the number of deterministic fractures that, on average, represent an inlet-outlet flow path for DFN158 (in the absence of a value equivalent to ℓ_{av} that cannot be easily computed for DFN158);
- The GI layers are built with respect to the adjacency matrix A of DFN158; indeed, the line graph of the DFN is not needed since the features (i.e., the transmissivities) are assigned to the nodes of the graph and not to the edges.

As in Section 3.4.3, the performance evaluation of the NNs trained with respect to the DFN flux regression task takes into account the errors MRE_{av} and MRE_{φ} on the test set (also, in this case, $\mathcal{P}$ is made of 3000 samples). Then, the results of the NNs are visualized as points in the $(\text{MRE}_{av}, \text{MRE}_{\varphi})$ plane (see Figure 3.10). Analyzing the error values and looking at the scatter plots of Figure 3.10, it clearly appears that the GINN models outperform the MLPs, and that they are characterized by more regular error behaviors than the GINNs trained for the maximum-flow regression task, with respect to the filter hyperparameter (see Section 3.4.3). In particular:

1. Both the MRE_{φ} and the MRE_{av} of the GINNs are almost always smaller than the ones of the MLPs, independently of ϑ;

2. Looking at the filter hyper-parameter F, the GINN performances are better as F increases (from $F = 1$, to $F = 5$, to $F = 10$).

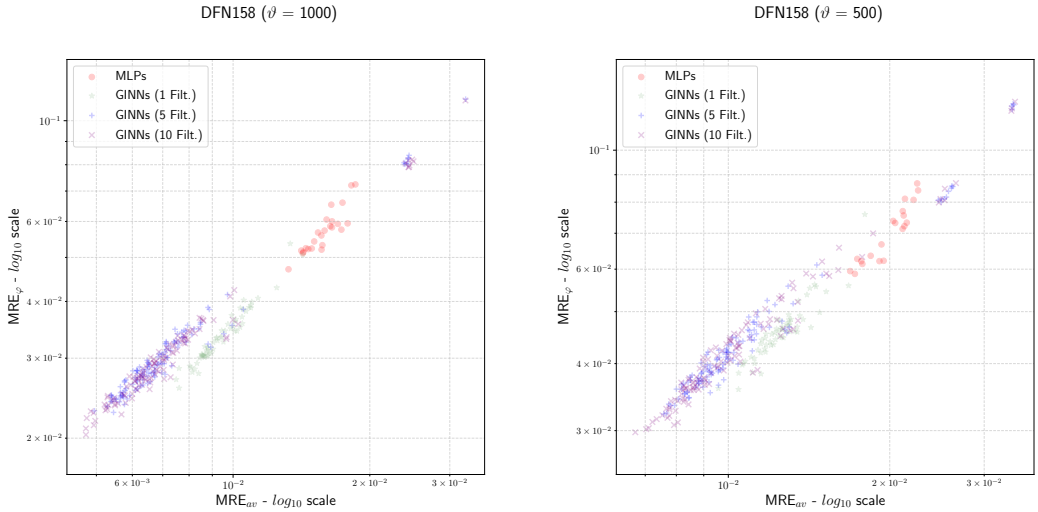

Figure 3.10: Scatter plots in the $(\text{MRE}_{av}, \text{MRE}_{\varphi})$ plane for NNs trained with respect to DFN158. NNs are trained using $\vartheta = 1000$ (**left**) and $\vartheta = 500$ (**right**) samples. Red circles: MLPs; green stars, blue crosses and purple "x": GINNs with $F =$ 1,5,10, respectively.

Furthermore, it is possible to study the relationships between the GINN errors and the other model hyperparameters. Considering the activation functions, the GINN models with the relu activation functions have, in general, slightly better performances than other models; on the other hand, all the GINN models with the worst performances (i.e., the points in the top-right corners of Figure 3.10) have softplus activation functions. Concerning the mini-batch size β, the GINNs with the best performances (corresponding to points in the bottom-left corners of Figure 3.10) are trained with $\beta = 32$. Similar results hold for the weights initialization, where the best performing GINNs are initialized with a Glorot normal distribution. About the pooling operations, specific differences in the error values of GINN models does not appear if using a max-pooling or a mean-pooling operation.

Analogous to Section 3.4.3, the study ends with a focus on the error behaviors with respect to the depth H of the GINN models. Moreover, for the DFN flux regression task, it can be observed that the depth H of the model can improve the regression quality. In particular, for each $\vartheta = 1\,000$,500, the best performances are obtained by the GINNs with a depth of $H = 7$,9,14, while both the shallowest

and the deepest GINNs ($H = 4{,}19$) have higher errors (see Figures 3.11 and 3.12). In accordance with Proposition 3.2.8 and the observations of Section 3.4.3, this characteristic make possible to deduce that, on average, the maximum inlet-outlet flux path in DFN158 is probably made of 8 to 15 fractures; i.e., a value not so far from the length of the inlet-outlet path defined by the fractures $\mathcal{F}_1, \ldots, \mathcal{F}_8$.

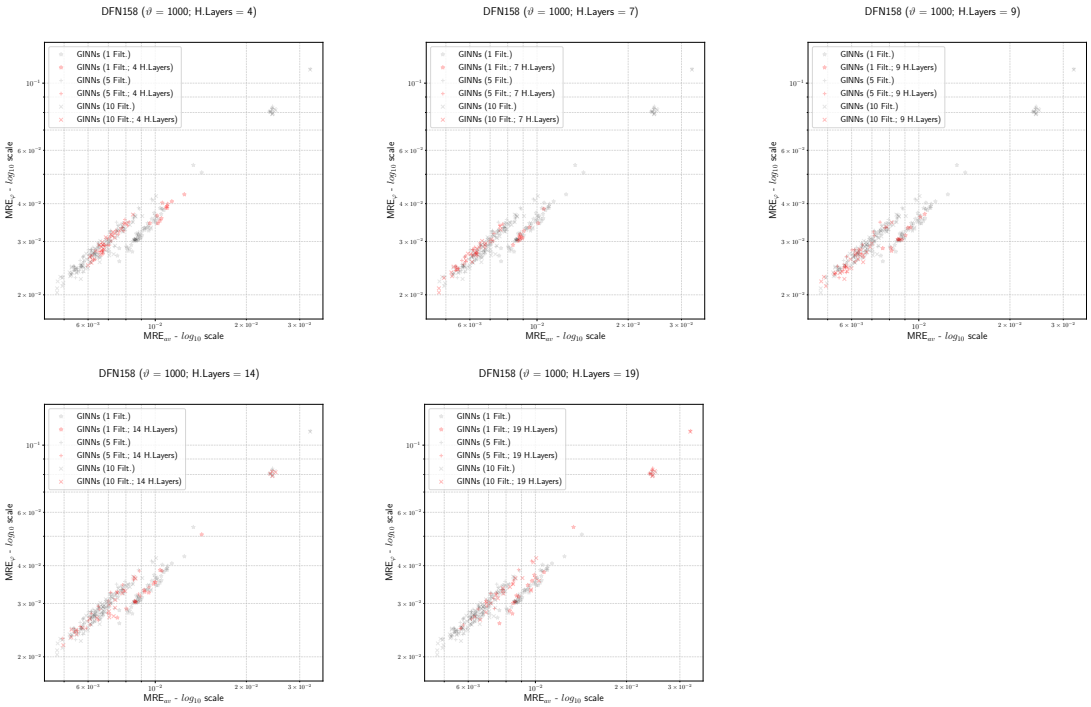

Figure 3.11: Scatter plots in the $(\mathrm{MRE}_{av}, \mathrm{MRE}_{\varphi})$ plane for GINNs trained with respect to DFN158, $\vartheta = 1000$. Left to right, top to bottom: red markers highlight GINNs trained with $H = 4{,}7{,}9{,}14{,}19$, respectively (black markers for all the other models).

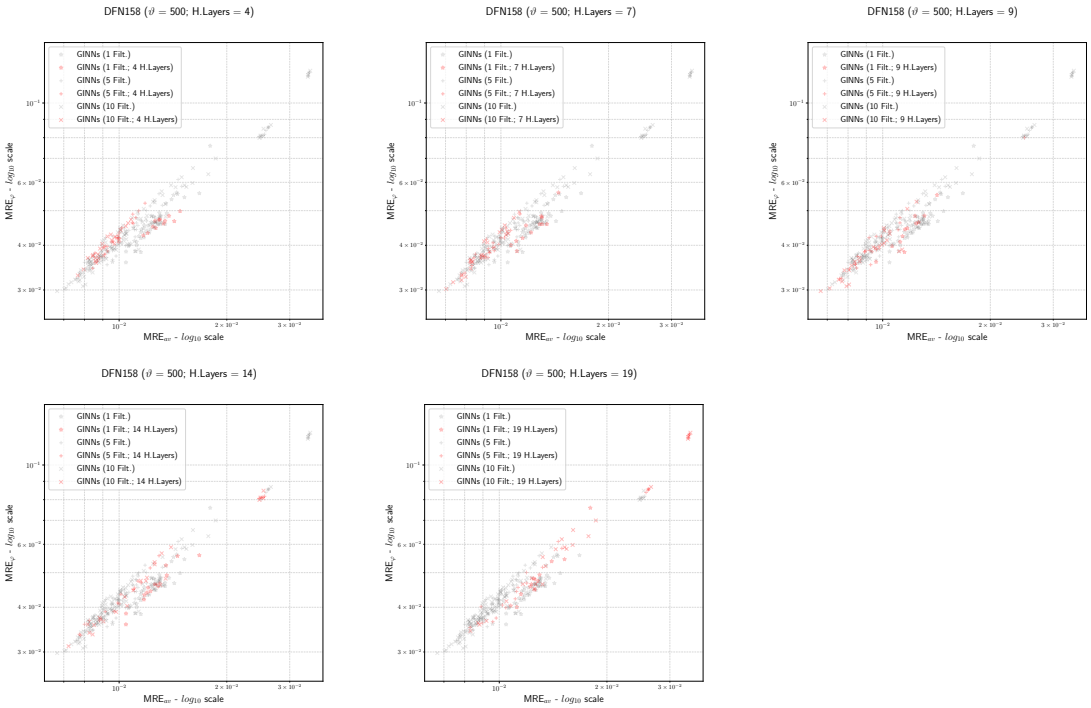

Figure 3.12: Scatter plots in the $(\mathrm{MRE}_{av}, \mathrm{MRE}_{\varphi})$ plane for GINNs trained with respect to DFN158, $\vartheta = 500$. Left to right, top to bottom: red markers highlight GINNs trained with $H = 4,7,9,14,19$, respectively (black markers for all the other models).

3.5 Conclusions and Future Directions

This chapter has presented and analyzed graph-informed (GI) layers, a new kind of spatial-based graph convolutional layer, designed for regression tasks on graph-structured data. The GI layers have been formally defined, from the simplest version to the most general and tensor version. Moreover, additional optional operations are introduced: the pooling operation and mask operations.

To study the regression capabilities of graph-informed NNs (GINNs), i.e., NNs made from GI layers, in the experiments thousands of NN models (both GINNs and MLPs) have been trained on two maximum-flow regression problems, with networks based on a Barabási–Albert graph ($\mathcal{G}_1$) and an Erdős–Rényi graph ($\mathcal{G}_2$). The maximum-flow regression problem is a representative case study, since it is a sufficiently general problem to demonstrate the applications in many topics of the network analysis. By analyzing the approximation errors of the NNs, overall, the GINNs emerges demonstrating better performances than the MLPs. In particular, for $\mathcal{G}_2$, the GINNs in almost all the cases outperform the MLPs. The study of the regression performances also showed an interesting relationship between small

errors and a depth greater than, or equal to, the average length of the maximum source-sink path in the stochastic network.

After the case study on the maximum-flow regression task, Section 3.4.4 has presented an example of a possible application of the GINNs to a real-world problem: a DFN flux regression problem, i.e., an uncertainty quantification problem for the characterization of the exiting flux distribution of an underground network of fractures. In this practical application, the GINN models completely outperform the MLPs; moreover, both the depth and the filter hyperparameters of the GINNs proved to be significant enough to improve the approximation quality of the target function.

From the application standpoint, the previous GNN architectures recalled in Section 3.2.3 are designed for different types of tasks, such as semi-supervised or supervised node classification or graph classification tasks [191], inferred from the known features of a subset of nodes. In particular, the comparison with NN4G [131] shows the novelty of the tensor formulation (see Equation (3.8)), that allows multiple output features. Generally speaking, the number of nodes and edges of the dataset faced in the experiments presented by the previous aforementioned works, like [15, 115, 190] are greater than the one faced in this work, therefore the need for faster convolutions. Nonetheless, the obtained experimental results suggest that on the examined test cases, the computational cost of the GINN implementation is comparable with MLP architectures. In addition, GINNs benefit from the depth hyperparameter much more than the previous graph convolutional NNs [191]. In the end, from a theoretical standpoint, nothing prevents from adding a softmax layer at the end of a GINN to extend the new model architecture to cover graph classification tasks with respect to vertex labels (like CNNs for image classification). A complete comparison, including the scalability of the GINN architecture on larger graph regression tasks, is out of the scope of this chapter, and it is left for future research.

With the presented applications, the idea of GINNs is clearly related to the framework of Theory-Guided Data Science (TGDS) [112], because the design of the architecture of the predictive model takes inside the knowledge of the phenomenon the data represents. In practice, not only the model makes use of the collected independent and dependent variables, but the relationship between the variables is inserted into the model by means of the graph, so that the NN is informed by the graph. Indeed, each model layer represents a building block for the task, mirroring the complete graph regression.

In conclusion, this work has introduced a new, useful, contribution to the family of spatial-based graph convolutional neural networks; indeed, the numerical experiments illustrated here show that the new GI layers and the GINNs have great potentialities in the framework of regression tasks on graph-structured data.

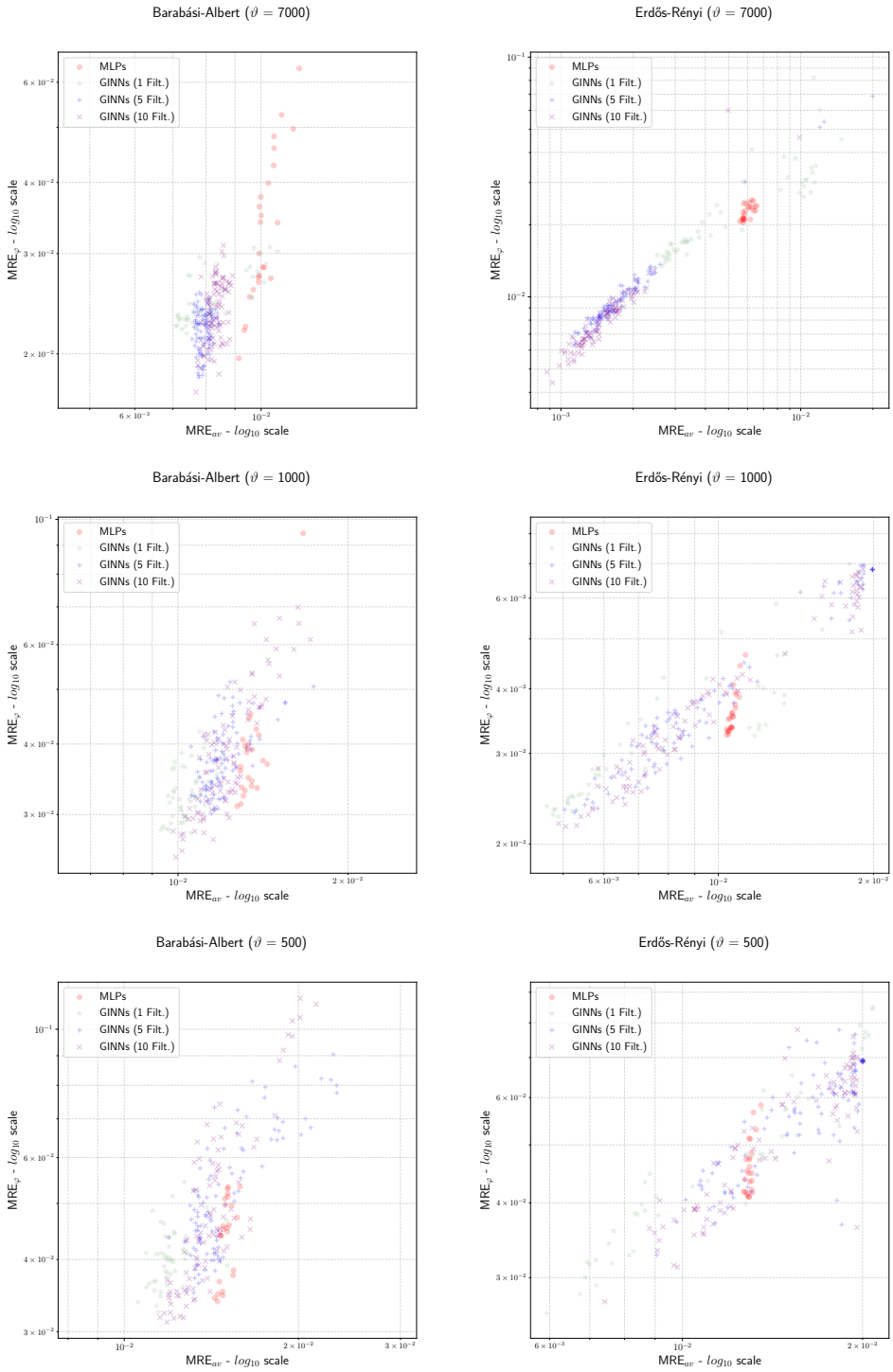

Figure 3.7: Scatter plots in the $(\text{MRE}_{av}, \text{MRE}_{\varphi})$ plane for $\mathcal{G}_1$ (**left**) and $\mathcal{G}_2$ (**right**). Top to bottom: ϑ = 7000,1000,500 samples. Red circles: MLPs; green stars, blue crosses and purple "x": GINNs with F = 1,5,10, respectively. NNs with ReLU activation function or mini-batch size 128 are omitted.

Chapter 4

Towards extending Shapley value to coalitions

The Shapley value is probably the most famous solution concept in cooperative game theory. Its name comes after the Nobel-winning Lloyd S. Shapley from his published work of 1953 [170]. The Shapley value has been applied to numerous fields, starting from economy [17], graph theory [1], statistics [104] or machine learning [126]. A classic essay [161] and a recent handbook [10] give a complete view of the Shapley value and its applications.

In the framework of cooperative game theory, a set of players or decision-makers should negotiate to agree how to split the gain of the cooperation, that is, how to allocate the worth gained by the coalition composed of all the players. A value is a solution concept that suggests the negotiation outcome among players. Many alternative solution concepts exist, and each solution fits for different modeling assumptions. For example, other solutions are the Banzhaf value [20], Deegan-Packel value [52], or the equal division solution [39]. The popularity of Shapley value derives from the property of being a "fair" allocation, where a set of desirable axioms defines the fairness principle. The axioms characterize the Shapley value because it is the unique value satisfying those properties; likewise, the axioms allow deriving a simple explicit combinatorial formula to compute the Shapley value.

This chapter suggests a new paradigm of allocating worth to all the coalitions of a cooperative game, similarly to what the Shapley value does for players. In other words, the purpose of the proposed allocation is to assign a scalar to each coalition, that is, every set of possible players; in addition, the assignment should follow a fairness principle inspired by the one holding for the Shapley value. The starting idea is extending the connection of [180] between cooperative game theory and graph theory. In particular, [180] proposes a representation of a transferable utility game as a graph, and it shows how to derive the Shapley value by the graph combinatorial Hodge decomposition. The development presented here follows the

most straightforward path: presenting a new graph representation of the game and a consequent different Hodge decomposition. This chapter defines a new differential, inspired by the one of [180], on the transitive closure of the Hasse diagram, which is a new oriented graph describing a cooperative game. Then, it shows the properties of the vector games derived from the combinatorial Hodge decomposition of the new oriented graph. Furthermore, the chapter illustrates how to compute the candidate analogous to the Shapley value for coalitions. The found simple analytic formula defines the newly denominated $\mathcal{X}$-Shapley allocations (also named coalitional Shapley). From $\mathcal{X}$-Shapley, it draws appealing theoretical results for cooperative game theory to show the strong parallelism between the new coalition allocation and the Shapley value. The main difference with the classical framework is that coalitions, instead of single players, are the main subjects of the cooperation. The presentation illustrates that $\mathcal{X}$-Shapley satisfies a principle of fairness through axioms defined for coalitions. In addition, it shows that the stability property of the Shapley value can be rephrased for $\mathcal{X}$-Shapley. Finally, it proves other significant properties of the newly defined coalitional allocation.

The precise motivation and scope of the investigations of this chapter derive from Explainable Artificial Intelligence (XAI) [62]. The Shapley value application as a feature importance method has been a recent breakthrough of XAI [126], opening a vast research horizon. The hope is that digging into cooperative game theory would make it possible designing advanced methods for XAI. In particular, the possibility of attributing a relevance score directly to a group of features and investigating their interaction is a current research topic [183]. Section 4.4.3 presents a deeper discussion about the research possibilities offered by the proposed $\mathcal{X}$-Shapley solution concept in XAI.

To the best of the author's knowledge, there are no other research papers that try to answer the same question. The most similar approaches are possibly the Owen value [149], interaction indices [90], or the value of a coalition to another coalition [100]. Yet, the question answered by those papers are clearly different from the one exposed, although they all take inspiration and generalize the Shapley value. In fact, the Owen value [149] proposes to generalize the Shapley value to allocate the worth of the grand coalition to single coalitions, assuming a fixed partition of players. This specific assumption is inspired by the dynamic usually emerging in a parliament, where players are representatives playing voting games to decide about each regulation. In this case, the set of coalitions to which assigning a value is not the complete power set of the participating players. Instead, the coalitions are the parties viewed as aggregations of players, where each party plays the role of one single player in the voting game. The Shapley value allows distributing the global power among the aggregations. Interaction indices [90] propose a generalization of a value accounting for the joint role of players in a cooperative game. The interactions inspect the game structure as well as measure the contribution of the joint presence of multiple players in a coalition. One specific interaction index is shown to be a

generalization of the Shapley value because, when analyzing the presence of a single player in a game, the interaction index becomes the Shapley value. The work of [100] proposes an abstract view on the evaluation process of coalition formation in a cooperative game. In particular, the authors say that the value of a coalition to another is a measure of the value of an already formed coalition when it decides to alter its members. Altering the members means modifying the coalition structure to form a different coalition by including or excluding some players. The authors suggest computing a matrix of values for each possible pair of coalitions. The generalization of Shapley value descends from the possibility of drawing the Shapley value as a summing of matrix rows or columns. The three discusses approaches do not view coalitions as single entities participating in the cooperative game; instead, the Owen value along with the interaction indices view them as a group of players, while the third looks at pairing of coalitions to evaluate the value of forming new coalitions starting from one pair.

The chapter is organized as follows. Section 4.1 recalls cooperative game theory to introduce the Shapley value and to describe three of its distinct characterizations. In particular, it focuses and comments on the recent one proposed in [180], based on the Hodge decomposition. Starting from this last characterization, Section 4.2 extends the parallelism between cooperative game theory and graph theory: it suggests the new allocation for coalitions called $\mathcal{X}$-Shapley and computes a simple analytic formula for its computation. Moreover, Section 4.3 proposes two game theoretic characterizations that are similar to the classic characterizations of the Shapley value already reviewed in Section 4.1. Then, Section 4.3.3 elaborates the properties of $\mathcal{X}$-Shapley from a game theoretic point of view. Finally, Section 4.4 concludes the presented study as well as it discusses the possible perspectives for future research.

4.1 Introduction to the Shapley value

Following the well known textbooks of game theory [148, 69, 137], this section starts with a brief comparison between non-cooperative and cooperative game theory (Section 4.1.1) to show the peculiarity of the cooperative framework. Then, Section 4.1.2 moves on by illustrating the class of cooperative games with transferable utility and their main related concepts. Section 4.1.3 defines the Shapley value, and it discusses different interpretations from the literature.

Then, three characterizations are presented. The first characterization is recalled in Section 4.1.4, which reviews the axiomatic characterization of the Shapley value. In particular, it defines the principle of fairness of a division of the worth grand coalition among players in a cooperative game. The property of fairness comes from the first work of Shapley [170], as reported by Theorem 4.1.43. Section 4.1.5 presents an alternative characterization of the same value by means of objections

and counter-objections. It recalls the stability property of the Shapley value solution for the cooperative game of interest. Theorem 4.1.47 reports the milestone result of this characterization. This result derives from the works of Myerson et al. [138, 136]. The presentation of the above two characterizations follows the book [148]. The third characterization of the Shapley value uses the combinatorial Hodge decomposition (Section 4.1.7), and it is presented in Section 4.1.8. This part revisits the results of a recent work [180], and in particular Theorem 4.1.80 and Proposition 4.1.84.

The three above characterizations are preparatory for the suggested $\mathcal{X}$-Shapley as allocation to coalitions. In particular, the Hodge theoretic characterization of $\mathcal{X}$-Shapley is the starting point for the next Section 4.2, while the other properties are discussed in Section 4.3 and 4.3.3.

4.1.1 Cooperative and Non-Cooperative Game Theory

A game expresses the interaction of multiple players, assuming they follow a rational behavior. A player is a decision-maker that can be, for example, an individual human being, a government, an institution, or a computer program. In a game, each player has a set of available actions to interact with the others. A player is also referred to as agent, because of being the active subject of the game. Each player's rationality generally translates into the goal of maximazing the utility obtained interacting, where the utility is specific for each player. In addition, the rationality assumption implies that the complexity of the computations or the sophistication of the strategies the player can make is unbounded. A game representation describes the strategic interaction among the players, including the constraint over the actions they can do [148]. A game solution is a systematic description of the results that could happen according to the rational interaction of the players. Different classes of games are present in the literature, but the main focus of this chapter is limited to cooperative games of transferable utility, as well as the deriving theory. Before delving into a precise description of cooperative game theory, below a brief introduction to its counterpart is furnished, named non-cooperative game theory.

Two game representations formalize and clarify these concepts: the strategic and characteristic form. The first and complete abstract representation of a game is the strategic form [148].

Definition 4.1.1 (Strategic form). *A strategic game consist of*

- $N \in \mathbb{N}$ *a finite set of players;*
- $\forall i \in N \in \mathbb{N}$ *a non-empty set* A_i *of the actions available to the agent; the product of their actions set is denoted by* $A = \times_{j \in N} A_j$*;*
- $\forall i \in N$*, a preference relation* $\lesssim_i$ *on* A*.*

This chapter refers to each player using the feminine pronouns "she/her", according to the convention established by Martin Osborne in [148]. Under mild assumptions, a function $u_i \colon A \longrightarrow \mathbb{R}$ can express the preference relation of a player i in a strategic form game. The function u_i is called utility function or payoff function for player i, and it is such that $u_i(a) \leq u_i(b) \iff a \lesssim_i b$. In this formulation, a player is rational because her objective is to maximize her payoff. Using the payoff function, the tuple $(N, (A_i)_{i \in N}, (u_i)_{i \in N})$ represents a game in strategic form.

The formulation above is general; it comprehends many possible modeling situations, referred to as "primitives" in the game theory literature. A simple primitive is a game where players simultaneously choose their actions, having information about the strategic behavior of the other players in the past. Each agent uses her information to find the best action, but she is interested only in her instantaneous payoff; that is, the agent assumes to play the game only once. The strategic form is the game representation mainly used in non-cooperative game theory. A solution consists of determining the strategy of each agent to produce an individual payoff for each one. The solution should be stable regarding the behavior of the players. Different definitions of stability lead to different resulted outcomes. An example of the solution concepts for non-cooperative game theory is the Nash equilibrium [140]. The pairing of stability and solution concepts also holds for cooperative games. Yet, the definition of stability comes at the level of agreement inside coalitions, not for players. In general, stability consists of the absence of a rational incentive to deviate from the solution outcome by groups of players.

The terms distinguishing a cooperative and a non-cooperative game could sometimes be confusing [50]. In fact, it does not hold true that non-cooperative games impede players' cooperation, neither that cooperative games permit cooperation only. Rather, the modelling process makes the difference: non-cooperative games include the possibilities for cooperation as actions in the game; cooperative games allow players to act outside the specified rules. In real situations, the complexity of the cooperation route does not enable to completely describe the players' negotiation by a formal mathematical model. Non-cooperative game theory handles strategies and payoffs: it assumes that players are choosing the actions to optimize their individual payoffs. Instead, cooperative game theory handles coalitions and allocations: it focuses on groups of players oriented to allocate the collective benefits coming from cooperation, however it appears. A non-cooperative game needs to define how coalitions form, as well as how their players select collective actions to model the possibility of coalition formation. A cooperative game could avoid these details; actually, the result of this kind of games does not show the set of actions chosen or how coalitions have been formed [148, 85], but only their outcome.

From a different angle, the distinction between a cooperative or non-cooperative game consists of effective negotiation. A cooperative setting assumes that players can negotiate effectively. More precisely, the assumption is that the negotiation happens at the same time within all the coalitions that include a player [137]. In

other words, it implicitly assumes that players would agree to modify their actions when there is a permitted change of players' strategies that would improve their utility. At the same time, the change does not come along with an explicitly specified strategy for the players. Instead, the players are assumed implementing the strategic change unless it contradicts the agreements of some members of the coalition. In fact, other players of the coalition could have treaties with different players outside this coalition, inside another equally effective coalition. A non-cooperative game does not allow the above because there are no implicit agreements: only individual actions of players could express treaties [137]. From the cooperative point of view, the actual strategy implemented to negotiate and form the coalition is not of interest; instead, only the coalition payoff is.

This chapter focuses on the class of cooperative game with transferable utility (TU-games). In other words, games where players should subdivide the payoff gained by cooperation without restrictions. In practice, it is common to assume that the transferable utility is a commodity - money - that can be split among the members of coalitions without limits, except that the sum of the amount received by each member should not exceed the worth gained by the grand coalition. This property differentiates with cooperative games with non-transferable utility (NTU-games), where there are constraints over the possible splitting of coalitions' gains. The game representation in characteristic form is the main subject for both classes, but here only the case of transferable payoff is presented. The reader can refer to [148, 96] for a complete view of the NTU-games class.

Notation 4.1.2. Consider a set of players N that are allowed to cooperate by forming coalitions. The coalition N is called the grand coalition. A coalition of the game is usually referred to as a subset $S \subseteq N$. In what follows, the set cardinalities are indicated by the corresponding lowercase characters (e.g. $n := |N|$, $s := |S|$). In addition, $\mathcal{P}_N$ indicates the powerset of N, that is, the set of all coalitions. The complement coalition of a given $S \subseteq N$ is denoted by $N \setminus S$ or S^c, if N is implied.

Definition 4.1.3 (Characteristic form of a TU-game). *A cooperative game with transferable utility (TU-game) is a pair (N, u) where*

1. *$N \in \mathbb{N}$ is the set of players;*
2. *$u \colon \mathcal{P}_N \to \mathbb{R}$ is the characteristic function of the game, requiring $u(\emptyset) = 0$.*

$u(S)$ represents the payoff or worth of the coalition $S \subseteq N$.

For each coalition $S \subseteq N$, its payoff $u(S)$ stands for the commodity that the members of S can split among themselves.

Notation 4.1.4. In what follows, $\mathcal{G}^N$ is the set of TU-games. Furthermore, each TU-Game (N, u) is identified with its characteristic function u.

When modeling a game allowing intrinsic agreements between multiple players, a critical challenge arises because of the assumption that all coalitions can negotiate

effectively. In particular, there can be a competition between overlapping coalitions, (see [137, Chapter 9.1]). Thus, the characteristic form allows representing the abstract dynamic of cooperation, while it voluntarily ignores the actual strategy chosen by each player to build agreements. For example, the rationality of players implies that in a two players game where the cooperation of the two gain more than the sum of singles, the grand coalition of the two players is formed. Yet, the actual chosen actions of the players are not explicitly expressed.

The analysis of the outcome represented in characteristic form derives from the hypothesis that the grand coalition forms because of the player's rationality; then, the players divide the grand coalition payoff $u(N)$ among themselves after some negotiation or bargaining. The final allocation to each player is the amount received at the end of the negotiation. The allocation derived from the negotiation should depend on the payoff structure of the coalitions, that is, their relative negotiation power, rather than the bargaining process itself. In these terms, a characteristic function shows a summary of the power structure of the game [138].

Finally, note that it is possible to translate a strategic form 4.1.1 to the characteristic form 4.1.3. This derivation is not unique; many proposals have appeared in the literature, depending on the actual modelled primitive. It is worth mentioning the rational-threats translation proposed by Harsanyi [97], based on the study of Nash in the bargaining two-person games [139].

Remark 4.1.5 (Harsanyi's rational-threats translation). Fixed a coalition S, in the view of rational threats both the players in S and $N \setminus S$ choose a joint action $a_S \in \times_{j \in S} A_j$ or $a_{N \setminus S} \in \times_{j \in N \setminus S} A_j$, respectively, such that both coalitions S or $N \setminus S$ are committed to carry out the action if the agreement would not be reached. Then, a_S and $a_{N \setminus S}$ are the threats of the two confronting parties, and the strategy $(a_S, a_{N \setminus S})$ is the disagreement point from which the negotiation power of subsets S and $N \setminus S$ descends. The interested reader could deepen the subject in [137, Chapter 9] or [69, Part IV, Chapter 1].

The above Remark is interesting for the formulation of $\mathcal{X}$-Shapley is Section 4.2 because of the direct opposition between the complementary coalitions of S and $N \setminus S$ of $\mathcal{X}$-Shapley (Definition 4.3.9). The following section recalls the main definitions of TU-games at the level of players, coalitions and games.

4.1.2 Transferable Utility Games

This section recalls the main definitions concerning cooperative game theory, focusing on the properties of cooperative games with transferable utility.

Notation 4.1.6. Hereafter, a game u denotes a TU-game $u \in \mathcal{G}^N$, according to the characteristic form of Definition 4.1.3. To simplify the notation, $u(i) := u(\{i\})$ denotes the payoff computed on a coalition of one player. Analogously to subsets terminology, a coalition S such that $S \in \mathcal{P}_N \setminus \{\emptyset, N\}$ is said proper coalition.

Denote by $\mathcal{S}_N$ the group of permutations of the players in N. Let $k \leq n$, denote by $S_1 \sqcup \cdots \sqcup S_k$ a partition of N in k subset, that is, $\sqcup_{i=1}^{n} S_i = N$ and $S_i \cap S_j = \emptyset$ for all $i \neq j$.

Definition 4.1.7 (Predecessors coalition). *Given $\sigma \in \mathcal{S}_N$, and $i \in N$, define the predecessors of i in σ as the coalition*

$$S_\sigma(i) := \{r \in N \mid \sigma^{-1}(r) < \sigma^{-1}(i)\} \tag{4.1}$$

Definition 4.1.8 (Permuted game). *Given a game u and $\sigma \in \mathcal{S}_N$, define the permuted game $\sigma_0^*(u)$ as:*

$$\sigma_0^*(u)(S) := u(\sigma(S)) \quad \forall S \subseteq N$$

Definition 4.1.9 (Payoff vector). *Given a game u, a payoff vector or payoff allocation is a vector $\boldsymbol{x} \in \mathbb{R}^n$, where each component x_i represents the allocation of the grand coalition payoff $u(N)$ to the single player $i \in N$. Given a coalition S, a payoff vector $\boldsymbol{x}$ is said S-feasible if $\sum_{i \in S} x_i = u(S)$. A payoff vector is called feasible if it is N-feasible*

Definition 4.1.10 (Imputation). *An imputation is a feasible payoff allocation $\boldsymbol{x} \in \mathbb{R}^n$ such that $x_i \geq u(i)$*

Definition 4.1.11 (Value). *Given a game u, a value is a function $\psi \colon \mathcal{G}^N \to \mathbb{R}^n$ that assigns a unique payoff vector to each game.*

Notation 4.1.12. A value depends on the number of players and the game over which it is evaluated. In the most general notation, the value $\psi(N, u)$ is expressed in terms of the number of players and the game, and the allocation to a player is $\psi_i(N, u)$. If not required, u and/or N are omitted, that is $\psi_i := \psi_i(u) = \psi_i(N, u)$.

Remark 4.1.13. A payoff vector represents a candidate solution of a game, that is, the assignment of an allocation to each player. The property of a payoff vector being feasible means that (i)the players receives all the worth, and (ii) the payoff vector does not exceed the available worth for the grand coalition. The imputation translates the players' rationality: each player is willing to cooperate because she will receive a greater allocated amount after the negotiation than by playing the game alone. A value is a solution concept because it is a rule to solve each game by assigning a payoff allocation.

Definition 4.1.14 (Inessential and essential game). *A game u is inessential if*

$$\forall S \subseteq N, \quad u(S) = \sum_{i \in S} u(i).$$

Equivalently, a game u is inessential if

$$\forall S \subseteq N \text{ and } T \subseteq N \setminus S, \quad u(S \cup T) = u(S) + u(T)$$

A game is essential if it is not inessential.

Remark 4.1.15. An inessential game has an additive payoff structure for each coalition, that can be derived from the payoff given to single player coalitions. Thus, it depicts the simpler possible relationship of cooperation between players in a game.

Definition 4.1.16 (Subgame). *Given a game u and a coalition S, define the subgame $u^S \colon \mathcal{P}_S \to \mathbb{R}$ as:*

$$\forall T \subseteq S \quad u^S(T) = u(T).$$

Note that $u^S(\emptyset) = 0$. Then every subgame is a game, and $u^N = u$.

Definition 4.1.17 (Superadditive and cohesive game). *A game u is said*

1. *superadditive:*

$$\forall S, T \subseteq N,\, S \cap T = \emptyset, \quad u(S \cup T) \geq u(S) + u(T); \tag{4.2}$$

2. *cohesive:*

$$\forall k \leq n, \quad \sqcup_{j=1}^{k} S_j = N, \quad u(N) \geq \sum_{j=1}^{k} u(S_j). \tag{4.3}$$

Remark 4.1.18. The superadditive property is a classical assumption, which justifies the formation of the grand coalition and all its subcoaltions in the game [135]. Indeed, when the game is superadditive, the rationality of the players implies that two coalitions would merge because, when the players of two disjoint coalitions S, T join, the resulting payoff to allocate is larger. A weaker assumption is the game cohesiveness. It again guarantees the grand coalition to form by players' rationality, but it does not justify all the sub-coalitions.

Definition 4.1.19 (Constant sum game). *A game u is constant sum (CS_S) if*

$$\forall S \subseteq N, \quad u(S) + u(N \setminus S) = u(N) \tag{4.4}$$

Remark 4.1.20. A constant sum game depicts a two-player competitive game between each pair of complementary coalitions. Indeed, each gain of one coalition is the loss of the other in the negotiation process.

Definition 4.1.21 (Dual game). *The dual game of u is the function $\bar{u} \colon \mathcal{P}_N \to \mathbb{R}$ defined as:*

$$\bar{u}(S) := u(N) - u(N \setminus S) \tag{4.5}$$

Note that the dual game is itself a game because $\bar{u}(\emptyset) = u(N) - u(N \setminus \emptyset) = 0$

Definition 4.1.22 (Quotient game). *Let $u \in \mathcal{G}^N$ with $n \geq 2$, and let $K = \{1, \ldots, k\}$ be such that $k \leq n$. Let $\mathfrak{P} = \{S_1, \ldots, S_k\}$ be a partition of N. Define $u^{\mathfrak{P}} \in \mathcal{G}^K$ as the game where the coalitions $S_1, \ldots, S_k$ of the partition are the players, that is:*

$$\forall \{i_1, \ldots, i_l\} \subseteq K, \quad u^{\mathfrak{P}}(i_1 \cup \cdots \cup i_k) = u(S_{i_1} \sqcup \cdots \sqcup S_{i_k}).$$

In particular, $u^{\mathfrak{P}}(\emptyset) = u(\emptyset) = 0$, and $u^{\mathfrak{P}}(K) = u(N)$.

Remark 4.1.23. In particular, if $k = 2$, $\mathfrak{P} = \{S, N \setminus S\}$, and the set of players is $K = \{1, 2\}$, where $S =$ player 1, $N \setminus S =$ player 2. The associated quotient game $u^{\mathfrak{P}}$ becomes:

$$\begin{aligned} u^{\mathfrak{P}}(\emptyset) &= 0 \\ u^{\mathfrak{P}}(\{1\}) &= u(S) \\ u^{\mathfrak{P}}(\{2\}) &= u(N \setminus S) \\ u^{\mathfrak{P}}(K) &= u(N). \end{aligned}$$

Definition 4.1.24 (Bilateral coalition). *A coalition S is said bilateral in the game u if*

$$u(S) = u(N \setminus S)$$

Definition 4.1.25 (Bilateral game). *A game u is said bilateral if all proper coalitions S are bilateral in u, that is:*

$$\forall S \subseteq N \text{ such that } S \neq \emptyset, N, \quad u(S) = u(N \setminus S).$$

Remark 4.1.26. The dual payoff $\bar{u}$ represents the amount the other agents in $N \setminus S$ cannot avoid S from obtaining in u, as reported by [147]. Notably, any constant sum game is self-dual, that is, $\bar{u} = u$. For bilateral games, individual players are relatively important because any coalition and its complement receive the same benefits. Indeed, when one player transfers from one coalition to the other, all other agents change their gain [132]. Bilateral coalitions make a bilateral structure at a coalitional level emerge.

Definition 4.1.27 (Marginal contribution of a player to a coalition). *Given a game u and a player $i \in N$, define the marginal contribution of the player i as:*

$$\forall S \subseteq N, \quad \Delta_i(S) := u(S \cup \{i\}) - u(S). \tag{4.6}$$

Definition 4.1.28 (Carrier coalition). *A coalition $R \subseteq N$ is a carrier in u if*

$$v(S \cap R) = v(S), \quad \forall S \subseteq N$$

Definition 4.1.29 (Dummy, Null, Nullifying player). *Given a game u, a player $i \in N$ is a :*

- *dummy player if $\forall S \subseteq N$ s.t. $i \notin S$, $\Delta_i(S) = u(i)$.*
- *null player if $\forall S \subseteq N$ s.t. $i \notin S$, $\Delta_i(S) = 0$.*
- *nullifying player if $\forall S \subseteq N$ s.t. $i \notin S$, $u(S \cup i) = 0$.*

Definition 4.1.30 (Symmetric players). *Given $i, j \in N$, they are said symmetric players if $\forall S \subseteq N$ s.t. $i, j \notin S$, $u(S \cup i) = u(S \cup j)$.*

Definition 4.1.31 (Equal division solution). *For each game $u \in \mathcal{G}^N$, the equal division solution ψ^{EQ} is the value*

$$\psi_i^{\mathrm{EQ}}(u) = \frac{u(N)}{N}$$

Remark 4.1.32. The marginal contribution of a player i expresses how much gain a player provides to a coalition S already formed. The concept of dummy player, null player, and carrier coalition are strictly linked. In particular, it is easy to notice that if R is a carrier, then every $i \notin R$ is a null player. In addition, if i is a null player, then $u(i) = 0$ [137, Chapter 9.4]. A nullifying player makes 0 the value of the coalition she joins. The nullifying player is involved in an axiomatic characterization of the equal division solution ψ^{EQ}, which distinguishes the Shapley value from the equal division [39]. Two symmetric players are indistinguishable from the point of view of the characteristic function. Notice that two symmetric players i, j have an equal marginal contribution to all the coalitions not containing them.

Definition 4.1.33 (Unanimity games and Dirac games). *Given a coalition $S \subseteq N$, $S \neq \emptyset$, for each $T \subseteq N$, define the unanimity game θ_S and the Dirac game τ_S as:*

$$\theta_S(T) = \begin{cases} 1 & \text{if } T \supseteq S \\ 0 & \text{otherwise} \end{cases}$$

$$\tau_S(T) = \begin{cases} 1 & \text{if } T = S \\ 0 & \text{otherwise} \end{cases}$$

Proposition 4.1.34 (Basis of $\mathcal{G}^N$). *The set of unanimity games $\{\theta_S\}_{S\subseteq N,\, S\neq\emptyset}$ and the set of Dirac games $\{\tau_S\}_{S\subseteq N,\, S\neq\emptyset}$ are linear algebraic basis for the space of games $\mathcal{G}^N$.*

Proof. The first part has already been shown by Lloyd S. Shapley in [170]. See also [148, Proposition 293.1] or [89, Theorem 2.56]. □

4.1.3 The Shapley value

This section defines the Shapley value and provides different interpretations to its combinatorial formulation.

Definition 4.1.35 (Shapley value). *Given $i \in N$ and $u \in \mathcal{G}^N$, the Shapley value ϕ_i is defined by the formula*

$$\phi_i(N, u) := \frac{1}{n!} \sum_{\sigma \in \mathcal{S}_N} \Delta_i(S_\sigma(i)), \tag{4.7}$$

where $S_\sigma(i)$ is the predecessor coalition of i in σ (Definition 4.1.7).

Remark 4.1.36 (Random ordering interpretation of the Shapley value). The interpretation of the Shapley value defined above is straightforward. Concretely, the idea is to suppose that the players could enter a room one by one according to a random order, and each time they are assigned the marginal contribution to the players already inside the room. More formally, let the players be ordered according to σ, and assume all orders being equally likely. Then, ϕ_i is the expected marginal contribution of the player i to the set of her preceding players over all ordering. [148, 96].

Remark 4.1.37 (Combinatorial interpretation of the Shapley value). By simple combinatorial manipulation, the formula 4.7 can be rewritten as

$$\phi_i(u) = \sum_{S \subseteq N \setminus \{i\}} \frac{s!(n-s-1)!}{n!} \Delta_i(S). \tag{4.8}$$

The formula 4.8 is the most used for computations. It allows proposing a second probabilistic interpretation of the Shapley value as observed in [96]: The contribution of player i is evaluated assuming that the cardinality of the coalitions she can join are equally probable. That is, first draw at random a number s from $\{1, \ldots, n-1\}$, representing cardinalities in $N \setminus \{i\}$. Then draw a coalition S of the drawn size s . Note also that the coefficients

$$\frac{s!(n-s-1)!}{n!} = \frac{1}{n}\binom{n-1}{s}^{-1} \in [0,1] \quad \text{and} \quad \sum_{S \subseteq N \setminus \{i\}} \frac{1}{n}\binom{n-1}{s}^{-1} = 1.$$

The above confirms that ϕ_i is a weighted average of the marginal contribution, where the weights depend on the coalition size.

Proposition 4.1.38 (About values depending on marginal contributions). *If $\psi \colon \mathcal{G}^N \to \mathbb{R}^N$ is a value defined by a linear combination of marginal contributions, that is, if $\forall i \in N$ exists coefficients $\{a_S\}_{S \subseteq N \setminus i} \in \mathbb{R}$ such that*

$$\forall u \in \mathcal{G}^N, \quad \psi_i(u) = \sum_{S \subseteq N \setminus \{i\}} a_S \Big(u(S \cup i) - u(S) \Big)$$

then

$$\forall i \in N, \quad \psi_i(u) = \sum_{S \subseteq N \setminus \{i\}} a_S \Big(u(N \setminus S) - u(S) \Big) \tag{4.9}$$

Proof. Fix $i \in N$. Observe that, by the change of summands $R := N \setminus (S \cup i)$, ψ is rewritten as:

$$\psi_i(u) = \sum_{R \subseteq N \setminus i} a_S \Big(u(N \setminus R) - u(N \setminus (R \cup i)) \Big). \tag{4.10}$$

Therefore:

$$\begin{aligned}
2\psi_i(u) &= \sum_{S\subseteq N\setminus\{i\}} a_S\Big(u(S\cup i)-u(S)\Big) + \sum_{S\subseteq N\setminus i} a_S\Big(u(N\setminus S)-u(N\setminus(S\cup i)\Big)\\
&= \sum_{S\subseteq N\setminus\{i\}} a_S\Big[u(S\cup i)-u(S)+u(N\setminus S)-u(N\setminus(S\cup i)\Big]\\
&= \sum_{S\subseteq N\setminus\{i\}} a_S\Big[u(N\setminus S)-u(S)+u(S\cup i)-u(N\setminus(S\cup i)\Big]\\
&\underset{(4.10)}{=} \sum_{S\subseteq N\setminus\{i\}} a_S\Big(u(N\setminus S)-u(S)\Big) + \sum_{S\subseteq N\setminus\{i\}} a_S\Big(u(N\setminus S)-u(N\setminus(S\cup i)\Big)\\
&= 2\sum_{S\subseteq N\setminus\{i\}} a_S\Big(u(N\setminus S)-u(S)\Big)
\end{aligned}$$

This proves the statement. □

Remark 4.1.39. The authors of [137, Chapter 9] describes a formula for the Shapley value similar to Equation (4.9). Proposition 4.1.38 shows in a general statement that every value expressed by a linear combination of marginal contributions of i to all the coalitions S non containing i can be rewritten in terms of differences between $u(N\setminus S)$ and $u(S)$. Examples are the Shapley value and the Banzhaf value [20]. As noted by [137, 97], the above proposition may suggest that the Harsanyi's rational-threats translation of Remark 4.1.5 is the best way to derive a game in characteristic form from the strategic form, when the Shapley value is being used as solution concept.

4.1.4 Axiomatic Characterization

The first characterization of the Shapley value comes from the concept of fair allocation of the payoff.

Definition 4.1.40 (Axioms for a fair value). *The value ψ is said to satisfy the axioms of:*

EFF$_i$ Efficiency if $\sum_{i\in N}\psi_i(u)=u(N)$;

NLL$_i$ Null player if, given a null player $i\in N$, then $\psi_i(u)=0$;

SYM$_i$ Symmetry if, given two symmetric players $i,j\in N$, then $\psi_i(u)=\psi_j(u)$;

LIN$_i$ Linearity if, given $a,a'\in\mathbb{R}$, and games u,u', then

$$\forall i\in N \quad \psi_i(a\,u+\,a'u')=a\,\psi_i(u)\,+\,a'\,\psi_i(u');$$

Remark 4.1.41 (About EFF$_i$, NLL$_i$, and SYM$_i$ axioms). The efficiency axiom requires that the payoff allocation provided by the value ψ should be feasible.

This assumption motivates by the need for a complete and non-exceeding allotment of the grand coalition payoff. In some texts, EFF_i is part of the value definition, for example [148, 137], while others keep it separated [85]. This section follows keeps it as a separate axiom to provide a better parallel when deriving a concept of value for coalitions in Section 4.2. NLL_i can be stated equivalently using carrier coalition (Definition 4.1.28) or dummy player (Definition 4.1.29). SYM_i states that only the role of an agent in the game should matter, not her label or names ("i") [137]. The NLL_i and SYM_i axioms express the fairness principle because one excludes players who do not contribute to the game looking at marginal contributions, while the other guarantees that equal marginal contribution to coalitions is given equal worth.

Remark 4.1.42 (About the LIN_i axiom). LIN_i is the most controversial from an interpretative point of view, although mathematically convenient. It has been originally stated as *additive axiom* by Shapley [170], that is, $\psi_i(u + u') = \psi_i(u) + \psi(u')$; the motivation is for the value to be additive in a system of interdependent games. Then, some authors proposed the value to be linear for convex combinations of games, that is $\psi_i(p\,u + (1-p)u') = p\,\psi_i(u) + (1-p)\,\psi_i(u')$, $p \in [0,1]$. In this case, p can be interpreted as the probability of playing u or u' according to some random event [137]. Some texts report it as additivity [148, 96, 69], others extends to linearity for convex combination of games [85, 137]. In this section, the axiom is stated in the most general linear form, as useful for some applications, for example in Machine Learning [126]. The problem of the linearity property outside convex combinations for game theory lies in the following: the loss of a connection between the games $a\,u + a'u'$, u and u' if $a, a' \in \mathbb{R}$, especially with a of negative sign. For example, a superadditive game u, which implies rationality, could become non superadditive in $a\,u$. Finally, note that linearity implies convex combination linearity, which again implies additivity.

Theorem 4.1.43 (Axiomatic uniqueness of the Shapley value). *The Shapley value ϕ is the only value that satisfies the axioms of EFF_i, NLL_i, SYM_i and LIN_i.*

Proof. The original proof is due to Lloyd Shapley [170, 169]. Similar proof of the statement is reported in textbooks [148, 85, 137, 96, 69]. □

Remark 4.1.44. Theorem 4.1.43 states that the Shapley value is the only one satisfying the fairness property as defined by the four above axioms. This result is fundamental for applications because it provides a formalized version of the fair allocation according to the power structure of the characteristic form of the game.

4.1.5 Counter-objections Characterization

This section recalls the characterization presented in [148, Section 14.4.1], although a more formal discussion should refer to [136, 98]. This second characterization of the Shapley value is codified by means of objection and counter-objection. The described property results in a stability concept of the allocation given by the Shapley value ϕ.

During the effective negotiation in the grand coalition, a request of deviation of one player i to a candidate allocation ψ may force other players to respond. Thus, the request causes a chain reaction of deviations that resolves in an outcome ψ'. If $\psi = \psi'$, the allocation is deemed stable, otherwise it is unstable. There are different possible stability concepts, that may characterize other solution of games with respect to ϕ, for example the Stable Sets [142], the Bargaining Set [16], the Kernel [51] or the Nucleolus [167]. The idea behind the concept of stability marking the Shapley value is that the chain reaction is cut short after two steps: the condition is that for every objection of player i motivating the deviation from the allocation, there is a player j who has a balancing counter-objection. The different notions of objection and counter-objection characterize different solution concepts. In the following, the focus is on the stability of ϕ.

Consider a game u. Recall Definition 4.1.16 of subgame. The following discussion defines the possible objections for a candidate allocation $\psi(N, u) \in \mathbb{R}^n$ of a player i against $j \neq i$. Assume that i wants a higher allocation $\psi_i(N, u)$ from j and she asks j to give more to i with one of the following arguments, expresses by the respective inequality:

1. i threatens to leave the game, so that j could obtain a smaller amount :
$$\psi_j(N \setminus i, u^{N\setminus i}) \leq \psi_j(N, u); \tag{4.11}$$
2. i threatens to persuade the other players $N \setminus \{i, j\}$ to exclude j from the game, so that i can receive more
$$\psi_i(N \setminus j, u^{N\setminus j}) \geq \psi_i(N, u). \tag{4.12}$$

Then, the player j could counter-object by saying one of the following in response, respectively:

1. j says that if i leaves then j would lose, but if j leaves then i would lose at least as much:
$$\psi_i(N, u) - \psi_i(N \setminus j, u^{N\setminus j}) \geq \psi_j(N, u) - \psi_j(N \setminus i, u^{N\setminus i}); \tag{4.13}$$
2. j says that if she is excluded, i would gain, but if j persuade the other players $N \setminus \{i, j\}$ to exclude i, then j would gain at least as much:
$$\psi_j(N \setminus i, u^{N\setminus i}) - \psi_j(N, u) \geq \psi_i(N \setminus j, u^{N\setminus j}) - \psi_i(N, u). \tag{4.14}$$

The following definition expresses the balancing between the statements of Equation (4.11) and (4.13), or Equation (4.12) and (4.14).

Definition 4.1.45 (Balanced contribution for players - BCP$_i$). *A value ψ satisfies the balanced contributions property for players (BCP_i) if, for every game u:*

$$\psi_i(N,u) - \psi_i(N \setminus j, u^{N\setminus j}) = \psi_j(N,u) - \psi_j(N \setminus i, u^{N\setminus i})$$

Remark 4.1.46. The property BCP_i for a value means that the outcome of a game using the reasoning of the value should derive from the same reasoning that describes the outcome of its subgames. The property translates into the stability of the outcome, or equilibrium of the allocation. Indeed, every objection of a player i towards the player j balances with a counter-objection of j towards i.

Theorem 4.1.47 (Shapley value and balanced contributions). *The Shapley value ϕ is the unique value satisfying EFF_i and BCP_i.*

Proof. See [148, Proposition 291.3] or [136] and [98]. □

4.1.6 Shapley value for XAI: SHAP

This section briefly recalls the application of the Shapley value solution concept of cooperative game theory to the field of Explainable Artificial Intelligence (XAI). The XAI class of algorithms have been already thoroughly described and summarized in Section 1.1.3. In particular, recall the assumptions for attribution methods (Notation 1.1.1), the definitions of Explanation Model (Definition 1.1.5) and Additive Feature Attribution Methods (Definition 1.1.6).

The seminal works of [182] and [181], proposing the algorithm named Shapley Sampling, are the starting papers for the application of the Shapley value in Machine Learning (ML). In particular, they propose a general method for explaining individual predictions of a classification ML model. [104] suggests a second line of research, where the goodness of fit of linear models is decomposed into relevance scores allocated to single features, again through the Shapley value. Yet, the breakthrough work of [126], proposing the SHAP framework, have laid the foundations for several studies about the Shapley value for XAI. Among the others, it is worth mentioning the paper of [49], which extends [126] to a global feature attribution method for model interpretation, and the paper of [2], that provides an algorithm to explain individual prediction when there are dependent features.

In general, all these methods provide additive feature explanation model $\boldsymbol{G}_{\widehat{\boldsymbol{F}}}$ assigning a relevance score ϕ_i to each of the N input features. The relevance score is computed through the application of the Shapley value to a cooperative game built on the ML predictive model $\widehat{\boldsymbol{F}}$ to be explained. There are many possibilities to derive a game from an ML model, so some authors have defined many explanation games [130]. This section recalls just the most used explanation game of [126],

though many limitations are known and improvements have been already suggested, for example [75] and again [2].

Notation 4.1.48. Fix a sample $\boldsymbol{x}^* \in \mathbb{R}^n$ to be explained. Consider the outcome explanation setting, and for the ease of presentation, assume that the simplified input features are the original features, that is $Z = X$. Therefore, each player $i \in N$ is a feature and a coalition $S \subseteq N$ is a feature subset; the grand coalition N is the set of all the features. The goal is to allocate the contribution of the actual prediction $\widehat{\boldsymbol{F}}(\boldsymbol{x}^*)$ to the single features x_i. In XAI, the allocation of the Shapley value becomes the attribution of the Explanation Model, as in Definition 1.1.5.

Definition 4.1.49 (SHAP Explanation game). *For each coalition S and ML model $\widehat{\boldsymbol{F}}$, the game characteristic function $u^{\widehat{\boldsymbol{F}}}(S)$ is defined as the expected value of the prediction, assuming that only the features in S are known.*

$$u^{\widehat{\boldsymbol{F}}}(S) = \mathbb{E}\left[\widehat{\boldsymbol{F}}(\boldsymbol{x}) \mid \boldsymbol{x}_S = \boldsymbol{x}_S^*\right]$$

Remark 4.1.50. In other words, it is the expected output of the ML model $\widehat{\boldsymbol{F}}$, conditional on the values of the features on the selected coalition $\boldsymbol{x}_S = \boldsymbol{x}_S^*$: The work of [126] calls the Shapley value for the features of the explanation game as SHAP. In addition, it shows that there is an axiomatic characterization for attribution, parallel to the one recalled for the Shapley value in Section 4.1.4. In particular, first, the authors restate the axioms of EFF_i, NLL_i, SYM_i, and LIN_i for the game $u^{\widehat{\boldsymbol{F}}}$ in terms of explanation model; for an additive feature attribution method, they become local accuracy, missingness, and consistency.

Theorem 4.1.51 (SHAP axiomatic uniqueness [126]). *There is only one possible explanation model $\boldsymbol{G}_{\widehat{\boldsymbol{F}}}$ satisfying the properties of local accuracy, missingness and consistency, defined by:*

$$\varphi_j^*(\widehat{\boldsymbol{F}}, \boldsymbol{x}^*) = \sum_{S \subseteq N \setminus \{i\}} \frac{s!(n-s-1)!}{n!} \left(u^{\widehat{\boldsymbol{F}}}(S \cup i) - u^{\widehat{\boldsymbol{F}}}(S)\right) \tag{4.15}$$

Proof. The proof is derived from [192]. □

Remark 4.1.52. In many practical situations, the SHAP value of Equation (4.15) is only approximated because of the exponential computation in n needed for the allocation to input features. Thus, the literature proposes many algorithms to compute the approximations, for example the already mentioned Shapley Samplings [181, 182], KernelSHAP, DeepSHAP [126], or DASP [12].

Example 4.1.53 (SHAP for a linear model and independent features). To conclude the presentation, just recall an example coming from [2, Appendix B], of the exact computation of SHAP in a simple scenario. Let $\widehat{\boldsymbol{F}}$ be a linear regression

model on a vector of n independent features $\boldsymbol{x} \in \mathbb{R}^n$:

$$\widehat{\boldsymbol{F}}(\boldsymbol{x}) = \sum_{j=1}^{n} \theta_j \, x_j.$$

The explanation game becomes

$$u^{\widehat{\boldsymbol{F}}}(S) = \sum_{j \in N \setminus S} \theta_j \mathbb{E}\left[x_j\right] + \sum_{j \in S} \theta_j x_j^*$$

Then, the SHAP value for each feature $\forall i = 1, \ldots, N$ can be expressed as: [2, Appendix B.1]:

$$\varphi_i^*(u^{\widehat{\boldsymbol{F}}}) = \theta_i \Big(x_i^* - \mathbb{E}\left[x_i\right] \Big).$$

In particular, if the features are zero-mean, that is $\forall i \in N, \mathbb{E}\left[x_i\right] = 0$, then

$$\varphi_i^*(u^{\widehat{\boldsymbol{F}}}) = \theta_i x_i^*$$

4.1.7 Interlude: Hodge Decomposition of a graph

This section presents the general mathematical framework of the combinatorial Hodge decomposition from graph theory, used in a recent characterization of the Shapley value [180].

Notation 4.1.54. Let V be a set of vertices and $E \subseteq V \times V$ be a set of edges connecting a pair of nodes in V.

Definition 4.1.55 (Oriented graph). *$G = (V, E)$ is called an oriented graph if given two vertices $a, b \in V$ connected by an edge, only one between (a, b) and (b, a) is in E.*

Definition 4.1.56 (Incidence matrix of oriented graph). *Given a graph G, the incidence matrix $M_G \in \mathbb{R}^{|V| \times |E|}$:*

$$\forall i \in V, \forall (j,k) \in E, \quad M_G[i,(j,k)] = \begin{cases} 1 & \text{if } j = i \text{ and } (i,k) \in E \\ -1 & \text{if } k = i \text{ and } (j,i) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Definition 4.1.57 (Degree of nodes in an oriented graph). *Given an oriented graph $G = (V, E)$, for each node $i \in V$ define the indegree $\deg^+$, the outdegree $\deg^-$, and the total degree $\deg$ as the cardinalities:*

$$\begin{aligned} \deg^+ i &= |\{j \in V \mid (j,i) \in E| \\ \deg^- i &= |\{j \in V \mid (i,j) \in E| \\ \deg i &= \deg^+ i + \deg^- i \end{aligned}$$

To explicit the graph over which the degree is computed, write $\deg_G^+$, $\deg_G^-$, $\deg_G$

Definition 4.1.58 (Adjacency matrix and degree matrix). *Given an oriented graph $G = (V, E)$, the unsigned and signed adjacency matrix, respectively $A_G, A'_G \in \mathbb{R}^{|V|\times|V|}$, and the degree matrix $D_G \in \mathbb{R}^{|V|\times|V|}$, are defined as:*

$$\forall i, j \in V, \quad A_G[i,j] = \begin{cases} 1 & \text{if } (i,j) \in E \text{ or } (j,i) \in E \\ 0 & \text{otherwise.} \end{cases}$$

$$\forall i, j \in V, \quad A'_G[i,j] = \begin{cases} 1 & \text{if } (i,j) \in E \\ -1 & \text{if } (j,i) \in E \\ 0 & \text{otherwise.} \end{cases}$$

$$\forall i, j \in V, \quad D_G[i,j] = \begin{cases} \deg_G i & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Definition 4.1.59 ($l^2(V)$ and $l^2(E)$). *Denote by $l^2(V)$ the space of functions with domain V and range $\mathbb{R}$, $u\colon V \to \mathbb{R}$, equipped with the inner product*

$$\langle u, v\rangle_{l^2(V)} := \sum_{a\in V} u(a)v(a). \tag{4.16}$$

Analogously, denote by $l^2(E)$ the space of functions with domain E and range $\mathbb{R}$, $f\colon E \to \mathbb{R}$, equipped with the inner product

$$\langle f, g\rangle_{l^2(\overline{E})} := \sum_{(a,b)\in E} f(a,b)g(a,b). \tag{4.17}$$

For $f\colon E \to \mathbb{R}$, set $f(b,a) := -f(a,b)$. In addition, note that the inner-product spaces $l^2(V)$ and $l^2(E)$ can be identified with the spaces $\mathbb{R}^{|V|}$, $\mathbb{R}^{|E|}$, respectively, each equipped with its Euclidean dot product. A canonical basis for the space $l^2(V)$ is the set of function $f_a \in l^2(V)$ such that $\forall a \in V$, $f_a(b) = 1$ if $a = b$, otherwise $f_a(b) = 0$.

Definition 4.1.60 (d and d^*). *The linear operator $\mathrm{d}_G\colon l^2(V) \to l^2(E)$ is defined as*

$$\mathrm{d}_G\, u(a,b) := u(b) - u(a). \tag{4.18}$$

Its adjoint $\mathrm{d}^*_G\colon l^2(E) \to l^2(V)$ is defined by

$$\langle u, \mathrm{d}^*_G\, f\rangle_{l^2(V)} = \langle \mathrm{d}_G\, u, f\rangle_{l^2(E)}$$

and it is explicitly provided by the formula:

$$(\mathrm{d}^*_G\, f)(a) := \sum_{(b\sim a)} f(b,a), \tag{4.19}$$

where $(b \sim a)$ denotes the set $\{b \in E \mid (a,b) \in E \text{ or } (b,a) \in E\}$.

Notation 4.1.61. Whenever the graph G is clear from the context, the notation omits the subscript G from the differential or its adjoint, that is, $\mathrm{d} := \mathrm{d}_G$ or $\mathrm{d}^* := \mathrm{d}_G^*$.

Remark 4.1.62. The operator d is the discrete analogue of the gradient in graph theory, while d^* is the discrete analogue of the negative divergence. Assuming canonical bases on $l^2(V)$ and $l^2(E)$, the matrix of d^*, $M(\mathrm{d}^*) \in \mathbb{R}^{|2^N|\times|E|}$, is the incidence matrix of the graph G, as in Definition 4.1.56; instead, the matrix of d is the transpose: $M(\mathrm{d}) = M(\mathrm{d}^*)^\top$

Definition 4.1.63 (Graph Laplacian). *Given a graph G, the graph Laplacian matrix $L_G \in \mathbb{R}^{|V|\times|V|}$ is defined as $L_G = \mathrm{d}_G^*\,\mathrm{d}_G$. If clear from the context, the subscript G is dropped for the graph Laplacian matrix $L = L_G$.*

Remark 4.1.64. The graph Laplacian defined above is the same as the usual graph Laplacian of spectral graph theory $L_G = D_G - A_G$ ([180]). Indeed, given a graph $G = (V, E)$, then $\forall u \in l^2(V)$, $a \in V$:

$$\begin{aligned} L_G\, u(a) = (\mathrm{d}_G^*\mathrm{d}_G\, u)(a) &= \sum_{(b\sim a)} \mathrm{d}u(b,a) = \sum_{(b\sim a)\in E} \Big(u(a) - u(b)\Big) \\ &= \deg(a)\, u(a) - \sum_{(b\sim a)} u(b) \\ &= \Big((D_G - A_G)\, u\Big)(a) \end{aligned}$$

where D_G and A_G are the degree matrix and the adjacency matrix of the graph G, defined in Definition 4.1.58. Recall the following properties of the graph Laplacian L:

(i) The diagonal entries of L equal to the vector of the node degrees;

(ii) The sum of each row and column of L is equal to 0;

(iii) $\mathrm{Ker}(L) = \mathrm{Ker}(\mathrm{d}^*\mathrm{d}) = \mathrm{Ker}(\mathrm{d})$, and $\mathrm{Im}(L) = \mathrm{Im}(\mathrm{d}^*)$.

(iv) The (algebraic and) geometric multiplicity of the eigenvalue 0 of L equals to the number of connected components of the graph G. In particular, if G is connected, then the algebraic multiplicity of 0 is 1;

Proposition 4.1.65 (Combinatorial Hodge decomposition). *the inner-product spaces $l^2(V)$ and $l^2(E)$ are decomposed as*

$$l^2(V) = \mathrm{Im}(\mathrm{d}^*) \oplus \mathrm{Ker}(\mathrm{d}) \quad \textit{and} \quad l^2(E) = \mathrm{Im}(\mathrm{d}) \oplus \mathrm{Ker}(\mathrm{d}^*), \tag{4.20}$$

where $\mathrm{Im}(\cdot)$ *and* $\mathrm{Ker}(\cdot)$ *denote the image space and the kernel space of a linear operator, respectively. The decomposition obtained in* (4.20) *is referred to as the combinatorial Hodge decomposition of $l^2(V)$ and $l^2(E)$.*

Proof. The result follows from the application of the fundamental theorem of linear algebra applied to d and d*, respectively. □

Definition 4.1.66 (Orthogonal projection on Im(d)**).** *Denote by* $P\colon l^2(E) \to \mathrm{Im(d)}$ *the orthogonal projection onto* Im(d)*.*

Remark 4.1.67. By the properties of the projector $P^2 = P$. In addition, given $\psi \in l^2(E)$, it holds the decomposition

$$\begin{aligned} \psi &= P\psi + (1 - P)\psi \\ &= \mathrm{d}\, u_\psi + r_\psi, \end{aligned} \tag{4.21}$$

with $u_\psi \in l^2(V)$ and $r_\psi \in \mathrm{Ker(d^*)}$. In addition, note that if Equation (4.21) holds, then applying d* on both sides gives:

$$\begin{aligned} \mathrm{d^*d}\psi &= \mathrm{d^*}u_\psi + \mathrm{d^*}r_\psi \\ L\,\psi &= \mathrm{d^*}u_\psi \qquad\qquad \text{by } r_\psi \in \mathrm{Ker(d^*)} \text{ and } L = \mathrm{d^*d} \end{aligned} \tag{4.22}$$

The focus of the study is on the component u_ψ.

4.1.8 Shapley value Characterization via Hodge Decomposition

The third characterization resumed here is based on the recent work of [180], starting from the framework introduced in Section 4.1.7. The authors introduce a decomposition of a game into a sum of inessential games (Definition 4.1.14), one for each player. This decomposition derives by formulating the game as a function on the vertices of a graph and computing the Hodge decomposition of the defined graph. The formulation allows getting the Shapley values for each player as the value assigned to the grand coalition of the inessential games in which the original game is decomposed. The goal is to highlight a connection between the Shapley values for players and the Hodge decomposition of a graph. This characterization will be the starting point for the suggested definition of a value for coalitions in the following Section 4.2.1.

Definition 4.1.68 (Game graph for players). *Let N be the set of n players in a game. Define the game graph for players as the oriented graph $\overline{G} = (V, \overline{E})$, where:*

1. $V := \mathcal{P}_N$*, i.e. the power set of the set of players;*
2. $\overline{E} := \{(S, S \cup \{i\}) \in V \times V \mid S \subseteq N \backslash \{i\},\, i \in N\}$.

Remark 4.1.69. This graph corresponds to a n-dimensional hypercube graph. Each vertex of the graph corresponds to a coalition S, and each edge to a player i

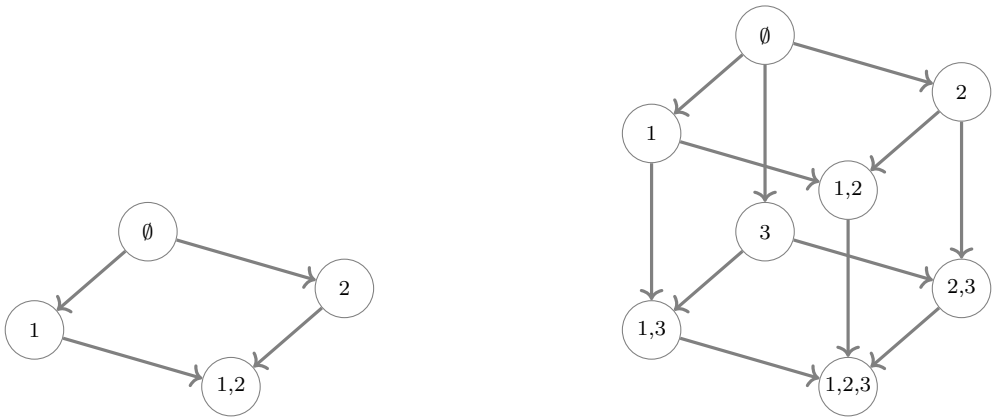

Figure 4.1: The game graph for players for a game with $n = 2$ (left) or $n = 3$ (right).

joining one coalition S not containing i. An edge is oriented in the direction of the inclusion $(S) \to (S \cup \{i\})$. See Figure 4.1 for an illustration of the graph $\overline{G}$.

In this framework, the defined graph is the Hasse diagram of the partial ordered set $(2^N, \overline{G})$. The Hasse diagram derives from the relation

$$\forall a, b \in V,\ a \preceq b \Leftrightarrow (a \leq b \wedge a \leq c \leq b \Longrightarrow c = a \vee c = b)$$

In order theory, a Hasse diagram represents a finite partially ordered set, also called poset, in the form of a drawing of its transitive reduction. In practice, for a poset $(A, \preceq)$, each element of A is a vertex in the plane and a curve goes downward from x to y whenever y covers x, that is, whenever $x \preceq y$ and there is no z such that $x \preceq z \preceq y$. These curves may cross each other, but must not touch any vertices other than their endpoints. Such a diagram, with labeled vertices, uniquely determines its partial order.

Remark 4.1.70 (Equivalent notation for a game). By the definition of game graph $\overline{G}$, the characteristic function u of a TU-game is as a function, element of $l^2(V)$. In addition, the same u can be viewed as a vector. Denote by $\boldsymbol{u}$ the vector in $\mathbb{R}^{2^n}$ such that $\boldsymbol{u} = \Big(\boldsymbol{u}[S]\Big)_{S \in \mathcal{P}_N}$, whose S-entry is equal to the corresponding function evaluated in the coalition S, that is $\boldsymbol{u}[S] = u(S)$. Note that $\boldsymbol{u}[\emptyset] = u(\emptyset) = 0$. Therefore, $\mathcal{G}^N$ is identified with the subspace of $l^2(V)$ made by the vectors $\boldsymbol{u}$ such that $\boldsymbol{u}[\emptyset] = 0$.

Definition 4.1.71 (Differential operator $\bar{\mathrm{d}}$ for game graph of players). *Denote with $\bar{\mathrm{d}} \colon l^2(V) \to l^2(\overline{E})$ the linear operator for the game graph $\overline{G}$ deriving from Equation* (4.18)*, such that :*

$$\bar{\mathrm{d}}u(S, S \cup \{i\}) = u(S \cup \{i\}) - u(S) \tag{4.23}$$

Remark 4.1.72. Equation (4.23) is a formal algebraic rewriting of the marginal contribution of the player i to the coalition $S \subseteq N \setminus i$ for u, as expressed in Definition 4.1.27. From the viewpoint of graph theory, $\bar{\mathrm{d}}$ is the 0-coboundary operator of $\overline{G}$, and its adjoint $\bar{\mathrm{d}}^*$ is the corresponding 0-boundary operator expressed as in Equation 4.19. Thus, because $\overline{G}$ is connected, the dimension of the kernel space of $\bar{\mathrm{d}}$ is $\dim \mathrm{Ker}(\bar{\mathrm{d}}) = 1$ [65, 124].

Definition 4.1.73 (i-player differential d_i). *For each $i \in N$, let $\bar{\mathrm{d}}_i \colon l^2(V) \to l^2(\overline{E})$ be the operator such that for all $S \subseteq N \setminus i$, and for all $u \in l^2(V)$:*

$$\bar{\mathrm{d}}_i\, u(S, S \cup \{j\}) = \begin{cases} \bar{\mathrm{d}} u(S, S \cup \{i\}) & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{4.24}$$

Remark 4.1.74. The i-player differential $\bar{\mathrm{d}}_i$ is nothing else than the zeroing of the operator $\bar{\mathrm{d}}$ on the edges of $\overline{G}$ not involving the marginal contribution of i.

Definition 4.1.75 (i-player Laplacian L_i). *Given $i \in N$, denote by $L_i \colon l^2(V) \to l^2(V)$ the i-player Laplacian defined by $L_i = \bar{\mathrm{d}}^* \bar{\mathrm{d}}_i$.*

Remark 4.1.76. L_i describes a weighted graph Laplacian of $\overline{G}$, where the edge $(S, S \cup j)$ have weight 1 if $i = j$ and 0 otherwise. In addition, note that L_i is represented by a symmetric matrix, like L.

Lemma 4.1.77 (Unique game decomposition for d_i). *Given $u \in \mathcal{G}^N$, the vector $\bar{\mathrm{d}}_i u \in l^2(\overline{E})$ can be uniquely decomposed as:*

$$\bar{\mathrm{d}}_i\, u = \bar{\mathrm{d}}\, u_i + r_i(u) \qquad s.t.\ \langle \bar{\mathrm{d}}\, u_i, r_i(u) \rangle_{l^2(\overline{E})} = 0, \tag{4.25}$$

where $u_i \in l^2(V)$, and $r_i(u) \in l^2(\overline{E})$. Furthermore, there exists a unique $u_i \in \mathcal{G}^N$ allowing the above decomposition.

Proof. For the ease of readability, the following proof report what observed in [180]. Using the Hodge decomposition of $l^2(\overline{E})$ of Equation (4.20), for a $u \in \mathcal{G}^N$ and a $i \in N$, there exists an element $u_i \in l^2(V)$ such that Equation (4.25) holds, where $r_i(u) \in \mathrm{Ker}(\bar{\mathrm{d}}^*)$. In addition, $\dim \mathrm{Ker}(\bar{\mathrm{d}}) = 1$ from what observed in Remark 4.1.72. Therefore, u_i is uniquely determined by imposing the condition $u_i(\emptyset) = 0$, that is, by the condition that u_i is a game. □

Remark 4.1.78 (Orthogonal projection on $\mathrm{Im}(\bar{\mathrm{d}})$). Recall the definition of Orthogonal projection of Definition 4.1.66, here denoted by $\overline{P}$. For example, $\overline{P}\, \bar{\mathrm{d}}_i\, u$ is the $\mathrm{Im}(\bar{\mathrm{d}})$ component of $\bar{\mathrm{d}}_i\, u$ in the Hodge decomposition.

Definition 4.1.79 (i-differential game). *Following Lemma 4.1.77, for each $i \in N$, let u_i be the unique game such that $\bar{\mathrm{d}} u_i = \overline{P} \bar{\mathrm{d}}_i\, u$. Call u_i the i-differential game of u.*

Theorem 4.1.80 (Decomposition by i-differential games [180]). *Given a game u, let u_i be the corresponding i-differential game for each $i \in N$, as in Definition 4.1.79. Then, the games u_i satisfy the following:*

(a) $\sum_{i \in N} u_i = u$;

(b) If $u(S \cup \{i\}) - u(S) = 0$ *for all* $S \subseteq N \backslash \{i\}$, *then* $u_i = 0$;

(c) If $\sigma \in \mathcal{S}_N$ *is a permutation of* N *and* $\sigma^* u$ *is the permuted game as in Definition 4.1.8, then* $(\sigma^* u)_i = \sigma^*(u_{\sigma(i)})$. *In particular, if* σ *swaps* i *and* j, *and if* $\sigma^* u = u$, *then* $u_i = \sigma^*(u_j)$.

(d) For any two games u, u' *and* $\alpha, \alpha' \in \mathbb{R}$, *then* $(\alpha u + \alpha' u')_i = \alpha u_i + \alpha' u'_i$

Proof. See Theorem 3.4 of [180]. □

Remark 4.1.81 (Inessential game and i-differential game). Note that by [180, Proposition 3.3], u is an inessential game (Definition 4.1.14) if and only if each i-differential game u_i is given by

$$u_i(S) = \begin{cases} u(\{i\}) & \text{if } i \in S \\ 0 & \text{if } i \notin S \end{cases}. \tag{4.26}$$

Thus, u can be decomposed as $u = \sum_{i \in N} u_i$. Theorem 4.1.80 generalizes the above decomposition to a generic game.

Corollary 4.1.82 (New characterization of ϕ_i). $u_i(N)$ *is the Shapley value* $\phi_i(u)$ *for each player* $i \in N$.

Proof. By the uniqueness of the Shapley value shown by Theorem 4.1.43. □

Remark 4.1.83. Observe that the properties [(a)-(d)] stated in Theorem 4.1.80 involve all the entries of the vectors $\boldsymbol{u}_i$, $i = 1, \ldots, n$, while the axiomatic characterization of the Shapley value involve just the N-entry of the vector $\boldsymbol{u}_i(N)$. Nonetheless, there is a clear correspondence between each of the axioms of Definition 4.1.4 and the properties [(a)-(d)].

Proposition 4.1.84 (Solution of Laplacian equation for players [180]). *Given* $u \in \mathcal{G}^N$, *for each* $i \in N$, *the i-differential game u_i is the unique solution* $x \in \mathcal{G}^N$ *to the equation:*

$$L\, x = L_i\, u \tag{4.27}$$

Proof. See [180, Proposition 3.9]. □

Remark 4.1.85. The work of [180] provides an alternative path to explicitly find the formula of the Shapley value. In particular, it manages to find the expression of each entry of the vectors $\boldsymbol{u}_i$ by computing the solution to Equation (4.27). The solution is found by inverting the graph Laplacian of the game graph $\overline{G}$ using the results of [46].

4.2 A Shapley value for Coalitions: $\mathcal{X}$-Shapley

This section proposes a new extension of the Shapley value solution concept. In particular, the aim is to derive a solution for a fair allocation to coalitions instead of players.

The extension starts from Section 4.2.1, where a Hodge theoretic argument similar to the one recalled in Section 4.1.8 is described. In particular, the interaction between players is viewed through the lens of a new oriented game graph, where the edges correspond to the marginal contribution of coalitions instead of players. The new point of view provides a decomposition similar to that of i-differential game (Definition 4.1.79). In this case, the elements of the decomposition are named S-differential games (Definition 4.2.14). Theorem 4.2.15 shows that S-differential games satisfy similar properties to those shown in Theorem 4.1.80 for i-differential games, where the subjects are the coalitions of a game instead of the players. Furthermore, Proposition 4.2.18 shows that S-differential games are solutions to Laplacian equations analogous to that analyzed in Proposition 4.1.84. Yet, in the case of coalitions, the computation of the solution of the Laplacian equation is not straightforward. Nonetheless, Theorem 4.2.34 shows that it is possible to compute the N-entry of the S-differential games. Then, the resulting solution concept for coalitions is called $\mathcal{X}$-Shapley.

4.2.1 Towards a characterization via the Hodge decomposition

To focus on coalitions of a game instead of players, a new version of the game graph is defined, where the edge set is modified with respect to Definition 4.1.68 to account for a concept of marginal contribution of coalition. To simplify the notation, G and E denote the new graph and the new edge set, respectively, differentiating with respect to the graph and edge set of the player version, $\overline{G}$ and $\overline{E}$. From now on, the graph G and the edge set E refers to this new game graph of coalitions.

Definition 4.2.1 (Game graph of coalitions). *Let N be the set of n players in a game. Define the oriented graph $G = (V, E)$ as:*

1. *$V := \mathcal{P}_N$, i.e. the powerset of the set of players;*
2. *$E := \{(S, T) \mid S, T \in \mathcal{P}_N,\, S \subsetneq T\} = \{(A,\, A \cup B) \mid A, B \in \mathcal{P}_N,\, A \cap B = \emptyset\}$.*

A game subgraph H denotes a subgraph of a game graph G. See Figure 4.2 for examples.

Remark 4.2.2. The game graph of coalitions G derives by the strict transitive closure of the game graph of players $\overline{G}$. The transitive closure of an oriented graph

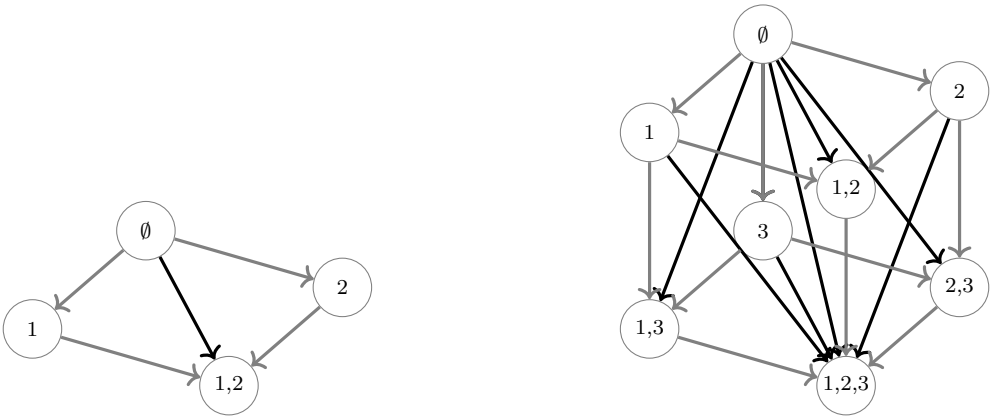

Figure 4.2: A representation of the graph G for a game with $n = 2$ (left) or $n = 3$ (right). Highlighted, the "diagonal" edges added in game graph for coalition with respect to game graph for players.

G is a graph having an edge (a, b) whenever G has a directed path from a to b. In practice, the difference between $\overline{G}$ and G is that the last comprehends the edges $(A,\ A \cup B)$ with $|B| > 1$.

Lemma 4.2.3 (Node degree in a game graph for coalitions). *Given a game graph $G = (V, E)$ over N players as in Definition 4.2.1,*

$$\forall S \in V, \quad \begin{cases} \deg_G^+ S = 2^s - 1 \\ \deg_G^- S = 2^{n-s} - 1 \end{cases} \qquad \text{where } s = |S|. \tag{4.28}$$

In particular the total degree of the nodes N and $\emptyset$ are:

$$\deg_G S = 2^n - 1. \tag{4.29}$$

Proof. For a given node $S \in V$, the total degree of S depends only on $s = |S|$. Indeed, for each node $S \in V$ there is:

- an inbound game subgraph $H^i = (V_{H^i}, E_{H^i})$ isomorphic to the game graph with $s = V_{H^i}$ players, where for each node $R \in V_{H^i}$ there is an edge $(R, S) \in E_{H^i}$, according to the definition of the edge set of G. In this case, the isomorphism sends the node $S \in V$ to the grand coalition N_S of H^i.
- an outbound game subgraph $H^o = (V_{H^o}, E_{H^o})$ isomorphic to the game graph with $n - s = |V_{H^o}|$ players, where for each node $T \in V_{H^o}$ there is an edge $(S, T) \in E_{H^o}$, according to the definition of the edge set of G. In this case, the isomorphism sends the node $S \in V$ to the empty set $\emptyset_S$ of H^o.

Then the $\deg_G^+$ of node $S \in V$ is the $\deg_{H^i}^+$ of the grand coalition with cardinality s. Because the grand coalition is connected with every other node in H^i different

from itself:

$$\deg_G^+ S = \deg_{H^i}^+ N_S = 2^s - 1.$$

Analogously, the $\deg_G^-$ of node $S \in V$ is the $\deg_{H^o}^-$ of the $\emptyset \in V_{H^o}$ with $n-s = |V_{H^o}|$. Because $\emptyset_S$ is connected with every other node in H^i different from itself:

$$\deg_G^- S = \deg_{H^i}^- \emptyset_S = 2^{n-s} - 1.$$

In particular, if $s = n$, $\deg_G S = \deg_G^+ S = 2^n - 1$ and if $s = 0$, $\deg_G S = \deg_G^- S = 2^n - 1$. □

Similar to the players setting, let us denote with $l^2(V)$ the space of the functions with domain in V and range $\mathbb{R}$. Likewise, let $l^2(E)$ be the space of functions with domain in E and range $\mathbb{R}$, both equipped with their inner product (Definition 4.1.59).

Definition 4.2.4 (Differential d operator for G). *Denote with* $\mathrm{d}\colon l^2(V) \to l^2(E)$ *the linear operator for the game graph G deriving from Equation* (4.18)*, such that :*

$$\forall S \subsetneq T \in \mathcal{P}_N, \quad \mathrm{d}\, u(S,T) := u(T) - u(S) \tag{4.30}$$

Definition 4.2.5 (S-coalition differential d_S). *For each $S \subseteq N$, let* $\mathrm{d}_S\colon l^2(V) \to l^2(E)$ *be the operator such that for all $A, B \in \mathcal{P}_N$, $A \cap B = \emptyset$, and for all $u \in l^2(V)$:*

$$\mathrm{d}_S\, u(A, A \cup B) = \begin{cases} \mathrm{d}\, u(A, A \cup S) & \text{if } B = S \\ 0 & \text{otherwise} \end{cases} \tag{4.31}$$

Note that $\mathrm{d}_\emptyset\, u = 0 \in l^2(E)$.

Lemma 4.2.6 (d sum of d_S). *The differential* d *can be written as a sum of S-coalition differentials* d_S*:*

$$\mathrm{d} = \sum_{S \subseteq N} \mathrm{d}_S.$$

Proof. Take $u \in \mathcal{G}^N$ and take $(A, A \cup B) \in E$, $A \cap B = \emptyset$.

$$\begin{aligned} \sum_{S \subseteq N} \mathrm{d}_S\, u(A, A \cup B) &= \mathrm{d}_B\, u(A, A \cup B) \quad \text{by } \mathrm{d}_S \text{ non-zero only on } S = B \text{ increments} \\ &= \mathrm{d}\, u(A, A \cup B) \end{aligned}$$

□

Lemma 4.2.7 (Computing d_N). *Given $u \in l^2(V)$,*

$$\forall A, B \in \mathcal{P}_N :\ A \cap B = \emptyset, \quad \mathrm{d}_N\, u(A, A \cup B) = \begin{cases} u(N) - u(\emptyset) & \text{if } A = \emptyset,\ B = N \\ 0 & \text{otherwise.} \end{cases} \tag{4.32}$$

In particular, if $u \in \mathcal{G}^N$, $\mathrm{d}_N u(\emptyset, N) = u(N)$.

Proof. For each $A, B \in \mathcal{P}_N$ such that $A \cap B = \emptyset$, if $A \neq \emptyset$ or $B \neq N$, then $\mathrm{d}_N u(A, A \cup B) = 0$. Only the edge $(\emptyset, N) \in E$ is such that $\mathrm{d}_N u\,(\emptyset, N) = \mathrm{d}u\,(\emptyset, N) = u(N) - u(\emptyset)$. □

Proposition 4.2.8 (Describing the kernel space of d). *The kernel space of the differential* d *can be described by:*

$$\mathrm{Ker}(\mathrm{d}) = \{[\lambda, \ldots, \lambda]^\top \in \mathbb{R}^{2^n} \mid \lambda \in \mathbb{R}\} \tag{4.33}$$

Proof. Denote by $\mathbf{1} \in l^2(V)$ the vector with entries equal to 1. Consider $u \in \mathrm{Ker}(\mathrm{d})$:

$$\mathrm{d}u = 0 \iff \forall A, B : A \subsetneq B \subseteq N,\ u(A) = u(B) \iff \boldsymbol{u} = u(\emptyset)\mathbf{1}.$$

Therefore, Equation (4.33) results calling $\lambda = u(\emptyset)$. □

Corollary 4.2.9 (d injective on $\mathcal{G}^N$). $\mathrm{Ker}(\mathrm{d}) \cap \mathcal{G}^N = \{\mathbf{0}\}$. *In particular,* d *restricted to* $\mathcal{G}^N$ *is injective.*

Proof. If $u \in \mathcal{G}^N \implies u(\emptyset) = 0$, then $u \in \mathrm{Ker}(\mathrm{d}) \cap \mathcal{G}^N \implies \lambda = 0$ in Equation (4.33). □

Definition 4.2.10 (Permutation operators σ_0^* and σ_1^*). *For all permutations of the* N *players* $\sigma \in \mathcal{S}_N$*, define:*

(i) $\sigma_0^* \colon l^2(V) \to l^2(V)$ *such that* $\forall u \in l^2(V)$, $\sigma_0^* u(S) := u\big(\sigma(S)\big)$
(analogous to Definition 4.1.8 of permuted game);

(ii) $\sigma_1^* \colon l^2(V) \to l^2(V)$ *such that* $\forall \psi \in l^2(E)$, $\sigma_1^* \psi(A, B) := \psi(\sigma(A), \sigma(B))$
(a permutation of the vertices σ *induces a permutation of the edges).*

Remark 4.2.11. $\forall \sigma \in \mathcal{S}_N, \quad A \subseteq B \subseteq N \implies \sigma(A) \subseteq \sigma(B)$

Lemma 4.2.12 (Properties of σ_0^* and σ_1^*). *Given* $S \subseteq N$*, consider the differential* d*, the associated orthogonal projection* P*, the differential* d_S*, and* $\sigma \in \mathcal{S}_N$*. The following holds:*

(i) $\mathrm{d}\sigma_0^* = \sigma_1^* \mathrm{d}$
(flipping d *and* σ_0^* *changes* σ_0^* *into* σ_1^**)*

(ii) $P\sigma_1^* = \sigma_1^* P$
(P *commutes with* σ_1^**)*

(iii) $\mathrm{d}_S \sigma_0^* = \sigma_1^* \mathrm{d}_{\sigma(S)}$
(a permutation of the argument of $u \in \mathcal{G}^N$ *followed by an increment by* S *is the same as an increment by* S *followed by a permutation)*

Proof. We prove the points in sequence.

Proof of (i). Given $u \in \mathcal{G}^N$, $\forall (A,B) \in E$,

$$\begin{aligned}
\mathrm{d}\,(\sigma_0^* u)(A,B) &= \sigma_0^* u(B) - \sigma_0^* u(A) && \text{definition of d}\\
&= u\Big(\sigma(B)\Big) - u\Big(\sigma(A)\Big) && \text{definition of } \sigma_0^*\\
&= \mathrm{d}\, u\Big(\sigma(A),\sigma(B)\Big) && \text{definition of d}\\
&= \sigma_1^*\, \mathrm{d}u(A,B) && \text{definition of } \sigma_1^*.
\end{aligned}$$

Thus (i) is proven.

Proof of (ii). Given $\psi \in l^2(E)$, remember from Remark 4.1.67 that $P\,\psi = \mathrm{d}u_\psi$, with $u_\psi \in l^2(V)$. Then, $\forall (A,B) \in E$

$$\begin{aligned}
P\,\sigma_1^*\psi(A,B) &= P\,\psi(\sigma(A),\sigma(B)) && \text{definition of } \sigma_1^*\\
&= \mathrm{d}\, u_\psi(\sigma(A),\sigma(B)) && \text{definition of } u_\psi\\
&= u_\psi\Big(\sigma(B)\Big) - u_\psi\Big(\sigma(A)\Big) && \text{definition of d}\\
&= \sigma_0^* u_\psi(B) - \sigma_0^* u_\psi(A) && \text{definition of } \sigma_0^*\\
&= \mathrm{d}\,\sigma_0^* u_\psi(A,B) && \text{definition of d}\\
&= \sigma_1^* \mathrm{d}\, u_\psi(A,B) && \text{(i) shown before}\\
&= \sigma_1^* P\,\psi(A,B) && \text{definition of } u_\psi.
\end{aligned}$$

Thus (ii) is proven.

Proof of (iii). Given $u \in \mathcal{G}^N$, by definition of d_S, definition of σ_0^*, using (i) and definition of σ_1^*, it holds $\forall (A,B) \in E$

$$\mathrm{d}_S\,\sigma_0^* u(A,B) = \begin{cases} \mathrm{d}\,\sigma_0^* u(A,B) = \sigma_1^* \mathrm{d}\, u(A,B) = \mathrm{d}\, u(\sigma(A),\sigma(B)) & \text{if } B = S\\ 0 & \text{otherwise}\end{cases} \tag{4.34}$$

By the same arguments, $\forall (A,B) \in E$:

$$\begin{aligned}
\sigma_1^* \mathrm{d}_{\sigma(S)}\, u(A,B) &= \mathrm{d}_{\sigma(S)}\, u(\sigma(A),\sigma(B)) =\\
&= \begin{cases} \mathrm{d}\, u(\sigma(A),\sigma(B))) & \text{if } \sigma(B) = \sigma(S) \iff B = S\\ 0 & \text{otherwise}\end{cases}
\end{aligned} \tag{4.35}$$

By comparing expressions of (4.34) and (4.35), (iii) is proven. □

Lemma 4.2.13 (Unique game decomposition for d_S). *Given $u \in \mathcal{G}^N$, the vector $\mathrm{d}_S u \in l^2(E)$ can be uniquely decomposed as:*

$$\mathrm{d}_S\, u = \mathrm{d}\, u_S + r_S(u) \quad \text{ s.t. } \langle \mathrm{d}\, u_S, r_S(u)\rangle_{l^2(E)} = 0, \tag{4.36}$$

where $u_S \in l^2(V)$, and $r_S(u) \in l^2(E)$. Furthermore, there exists a unique $u_S \in \mathcal{G}^N$ allowing the above decomposition.

Proof. The proof proceeds similarly to Lemma 4.1.77, using the injectivity of d on $\mathcal{G}^N$ from Corollary 4.2.9. □

Definition 4.2.14 (S-differential game). *Following Lemma 4.2.13, for each $S \in N$, let $u_S \in l^2(V)$ with $u_S(\emptyset) = 0$ be the unique game such that $\mathrm{d}u_S = P\mathrm{d}_S\, u$, where $P\mathrm{d}_S u$ is the $\mathrm{Im}(\mathrm{d})$ component of $\mathrm{d}_S\, u$ in the Hodge decomposition. Call u_S the S-differential game of u.*

The following Theorem is inspired by the Theorem 4.1.80, but it focuses on coalitions instead of single players.

Theorem 4.2.15 (Decomposition by S-differential games). *Given a game u, let u_S be the corresponding S-differential game for each $S \in \mathcal{P}_N$. Then, the games u_S satisfy the following:*

(a) $\sum_{S\subseteq N} u_S = u$;

(b) if $u(S \cup T) - u(T) = 0$ *for all* $T \subseteq N \setminus S$, *then* $u_S = 0$;

(c) if $\sigma \in \mathcal{S}_N$ *and* $\sigma^* u$ *is the permuted game, then* $(\sigma^* u)_S = \sigma^* (u_{\sigma(S)})$. *In particular, if* σ *is a permutation that swaps* S *and* T, *with* $S \cap T = \emptyset$, *and if* $\sigma^* u = u$, *then* $u_S = u_T$.

(d) For any two games u, u' *and* $\alpha, \alpha' \in \mathbb{R}$, *then* $(\alpha\, u + \alpha'\, u')_S = \alpha\, u_S + \alpha'\, u'_S$

Proof. Let the games u_S be as Definition 4.2.14. Prove the properties (a)-(d).

Proof of (a). Let us prove that u_S satisfy (a). Take $u \in \mathcal{G}^N$ and apply both sides of Lemma 4.2.6 to u.

$$
\begin{aligned}
\mathrm{d}\, u &= \sum_{S\subseteq N} \mathrm{d}_S\, u && \\
P\,\mathrm{d}\, u &= P \sum_{S\subseteq N} \mathrm{d}_S\, u && \text{apply } P \text{ to both sides} \\
\mathrm{d}\, u &= \sum_{S\subseteq N} P\,\mathrm{d}_S\, u && \text{linearity of } P \text{ and } P\mathrm{d}\, u = \mathrm{d}\, u \\
\mathrm{d}\, u &= \sum_{S\subseteq N} \mathrm{d}\, u_S && \text{definition of } u_S \\
\mathrm{d}\, u &= \mathrm{d} \sum_{S\subseteq N} u_S && \text{linearity of } d \\
u &= \sum_{S\subseteq N} u_S && \text{Corollary 4.2.9: d injective on } \mathcal{G}^N
\end{aligned}
$$

Proof of (b).

$$\begin{aligned}
&\forall T \subseteq N \setminus S,\ u(T \cup S) - u(T) = 0 && \text{hyphotesis}\\
\Longrightarrow &\forall T \subseteq N \setminus S,\ \mathrm{d}_S\, u(T, T \cup S) = 0 && \text{Definition 4.2.5 of } \mathrm{d}_S\\
\Longrightarrow &\mathrm{d}_S u = \mathbf{0} && \mathbf{0} \in l^2(E)\\
\Longrightarrow &P\,\mathrm{d}_S\, u = \mathbf{0} = \mathrm{d}\, u_S && \text{definition of } u_S\\
\Longrightarrow &u_S = 0 && \text{Corollary 4.2.9: d injective on } \mathcal{G}^N
\end{aligned}$$

Proof of (c).

$$\begin{aligned}
\mathrm{d}\Big((\sigma_0^* u)_S\Big) &= P\mathrm{d}_S(\sigma_0^* u) && \text{definition of } u_S \text{ applied to } (\sigma_0^* u)_S\\
&= P\sigma_1^* \mathrm{d}_{\sigma(S)}(u) && \text{Lemma 4.2.12.(iii)}\\
&= \sigma_1^* P\mathrm{d}_{\sigma(S)}(u) && \text{Lemma 4.2.12.(ii)}\\
&= \sigma_1^* \mathrm{d} u_{\sigma(S)} && \text{definition of } u_{\sigma(S)}\\
&= \mathrm{d}\sigma_0^* u_{\sigma(S)} && \text{Lemma 4.2.12.(i)}
\end{aligned}$$

Therefore, $\mathrm{d}\Big((\sigma_0^* u)_S\Big) = \mathrm{d}\sigma_0^* u_{\sigma(S)}$, and by Corollary 4.2.9:

$$(\sigma_0^* u)_S = \sigma_0^*(u_{\sigma(S)}) \tag{4.37}$$

Consider now $S, T \subseteq N$ s.t. $S \cap T = \emptyset$ and $\sigma \in \mathcal{S}_N$ s.t. $\sigma(S) = T$ and $\sigma(T) = S$. In other words, σ swaps S and T. In addition, assume as before that $\sigma_0^* u = u$.

$$\begin{aligned}
u_S &= (\sigma_0^* u)_S && \text{assumption } \sigma_0^* u = u\\
&= \sigma_0^*(u_{\sigma(S)}) && \text{Equation (4.37)}\\
&= \sigma_0^* u_T && \sigma(S) = T\\
&= u_T && \text{assumption } \sigma_0^* u = u
\end{aligned}$$

and this concludes the proof of (c).

Proof of (d). Let us prove now linearity. Let $\alpha, \alpha' \in \mathbb{R}$ and u, u' be two games. Then, since $\mathrm{d}\, u_S = P\mathrm{d}_S\, u$,

$$\begin{aligned}
\mathrm{d}\,(\alpha\, u + \alpha'\, u')_S &= P\,\mathrm{d}_S\,(\alpha\, u + \alpha'\, u') && \text{definition of } u_S\\
&= \alpha\, P\,\mathrm{d}_S\, u + \alpha'\, P\,\mathrm{d}_S\, u' && \text{linearity of } P \text{ and } \mathrm{d}_S\\
&= \alpha\,\mathrm{d}\, u_S + \alpha'\,\mathrm{d}\, u'_S && \text{definition of } u_S
\end{aligned}$$

□

Remark 4.2.16. Following what done for players in Section 4.1.8, the question now is if $u_S(N)$ represents a concept of value for the coalition S, similarly to the

Shapley value as pointed out in by Corollary 4.1.82 and Remark 4.1.83. From the discussion until now, no answer is possible because there is not a uniqueness result like Theorem 4.1.43 that allows deriving a similar conclusion. The following sections try to analyze the properties of $u_S(N)$ and derive characterizations similar to those of the Shapley value for players.

Definition 4.2.17 (Coalitional Laplacian L_S). *Given $S \subseteq N$, $S \neq \emptyset$, denote by $L_S \colon l^2(V) \to l^2(V)$ the coalitional Laplacian defined by $L_S = \mathrm{d}^* \mathrm{d}_S$.*

Proposition 4.2.18 (Solution of Laplacian equation for coalitions). *Given $u \in \mathcal{G}^N$, $\forall S \subseteq N$, $S \neq \emptyset$, the S-differential game u_S represents the unique solution $x \in \mathcal{G}^N$ to the equation:*

$$L\, x = L_S\, u \tag{4.38}$$

Proof. Proceeding like Proposition 4.1.84 proven in [180]: first, show that u_S is a solution of (4.38), second show uniqueness.

Proof of S-differential game is solution. From Lemma 4.2.13:

$$r_S = \mathrm{d}_S\, u - \mathrm{d}\, u_S \tag{4.39}$$

with $u_S \in \mathcal{G}^N$ and $r_S \in \mathrm{Ker}(\mathrm{d}^*)$. Then, applying d^* to both sides of Equation (4.39)

$$0 = \mathrm{d}^*(\mathrm{d}_S\, u - \mathrm{d}\, u_S) = \mathrm{d}^*\mathrm{d}_S\, u - \mathrm{d}^*\mathrm{d}\, u_S = L_S\, u - L\, u_S$$

So u_S is a solution of Equation (4.38).

Proof of uniqueness. Observe that $\mathrm{Ker}(L) = \mathrm{Ker}(\mathrm{d})$ by Remark 4.1.64.(iii), so L is injective on $\mathcal{G}^N$, and u_S is the unique solution. □

Remark 4.2.19. The last Proposition 4.2.18 is the natural translation of Proposition 4.1.84 from [180] to the new game graph G of Definition 4.2.1.

4.2.2 Interlude: Preliminary Results and Notation

Before going forward to derive a candidate Shapley value for coalitions, this section shows preliminary results about possible partitions of $\mathcal{P}_N$, and it fixes a notation useful for the computation of the analytic formula of $\mathcal{X}$-Shapley of the following Section 4.2.3.

Lemma 4.2.20 ($\mathcal{A}$-Partition of $\mathcal{P}_N$). *If $S \subseteq N$ and $S \neq \emptyset$, then the power set of N can be partitioned into:*

$$\mathcal{P}_N = \mathcal{A}_1^S \sqcup \mathcal{A}_2^S \sqcup \mathcal{A}_3^S \tag{4.40}$$

where

$$\begin{aligned}\mathcal{A}_1^S &:= \{T \subseteq N \setminus S\} \\ \mathcal{A}_2^S &:= \{T \cup S \mid T \in \mathcal{A}_1\} \\ \mathcal{A}_3^S &:= \{T \subseteq N \mid S \neq T \cap S \neq \emptyset\}.\end{aligned}$$

Proof. Let $T \subseteq N$. Three alternative cases emerge:

(i) $T \cap S = \emptyset \implies T \in \mathcal{A}_1^S$;

(ii) $T \cap S = S \implies T \in \mathcal{A}_2^S$;

(iii) $T \cap S \neq \emptyset$ and $T \cap S \neq S \implies T \in \mathcal{A}_3^S$.

□

Notation 4.2.21. In the following, omit the dependency over S for the $\mathcal{A}$-sets to simplify the notation, that is, $\mathcal{A}_j = \mathcal{A}_j^S$.

Remark 4.2.22. Focus on the cases $s = n$ and $s = 0$. The case $s = 0$ is not considered in Lemma 4.40, indeed the $\mathcal{A}$-Partition degenerates:

(i) if $s = 0$, then the $\mathcal{A}^\emptyset$-sets do not form a partition because $\mathcal{A}_1 = \mathcal{A}_2 = \mathcal{P}_N$ and $\mathcal{A}_3 = \emptyset$;

(ii) if $s = n$, then the $\mathcal{A}^N$-sets become $\mathcal{A}_1 = \{\emptyset\}$, $\mathcal{A}_2 = \{N\}$ and $\mathcal{A}_3 = \mathcal{P}_N \setminus \{\emptyset, N\}$.

Corollary 4.2.23 (Emptiness of $\mathcal{A}_3$ for singletons). *Under the same hypotheses of Lemma 4.2.20,*

$$|S| = 1 \implies \mathcal{A}_3 = \emptyset. \tag{4.41}$$

Proof. Consider $S = \{i\}$. Use a proof by contradiction. Take $T \in N$ and suppose that both $T \cap S \neq S$ and $T \cap S \neq \emptyset$. Obtain $T \cap S = T \cap \{i\} \subseteq \{i\}$. Then $|T \cap \{i\}| \leq |\{i\}| = 1$. This means $|T \cap \{i\}|$ can only be 0 or 1. If $|T \cap \{i\}| = 0$, then $T \cap S = \emptyset$, that is a contradiction. If $|T \cap \{i\}| = 1$, then $T \cap S = \{i\} = S$, that is a contradiction. □

Remark 4.2.24. Remember that Lemma 4.2.20 holds for all proper subsets S of N. So, assume $n > 1$; $\forall S \subseteq N$ such that $1 \leq s \leq n-1$, then $\mathcal{A}_1 \neq \emptyset$ because $\mathcal{A}_1 \supseteq \{\emptyset\}$. In addition, under the same assumption, it follows that $\mathcal{A}_2$ is a nonempty set because $\mathcal{A}_2 = \{T \cup S \mid T \in \mathcal{A}_1\} \supseteq \{S\}$.

Corollary 4.2.25 (Complementary relations of the $\mathcal{A}$-Partition). *Under the same hypotheses of Lemma 4.2.20, it holds that*

$$\forall T \in \mathcal{A}_1,\ N \setminus T \in \mathcal{A}_2 \text{ and viceversa, } \forall T \in \mathcal{A}_2,\ N \setminus T \in \mathcal{A}_1 \tag{4.42}$$

$$\forall T \in \mathcal{A}_3,\ N \setminus T \in \mathcal{A}_3 \tag{4.43}$$

Proof. First, show claim 4.42. Given a set $T \subseteq N$, denote $T^c = N \setminus T$. If $T \in \mathcal{A}_1$, then $T \subseteq S^c \Longrightarrow T^c \supseteq S$. Therefore, denoting by $R = T^c \setminus S$, then $R \subseteq N \setminus S$, that is, $R \in \mathcal{A}_1$, and $T^c = R \cup S$. The above have just shown that T^c is a union of an element of $\mathcal{A}_1$ and S so $T^c \in \mathcal{A}_2$ and (4.42) is proven.

Second, show claim (4.43). If $T \in \mathcal{A}_3$ then $T \cap S \neq S$ and $T \cap S \neq \emptyset$. Consider T^c. Find a contradiction on two cases:

1. suppose $T^c \cap S = S$. Then $T^c \supseteq S \Longrightarrow T \subseteq S^c \Longrightarrow T \in \mathcal{A}_1$. But this is a contradiction because $T \in \mathcal{A}_3$ by hypothesis and $\mathcal{A}_1, \mathcal{A}_3$ are disjoint sets.

2. suppose $T^c \cap S = \emptyset$. Then $T^c \subseteq S^c \Longrightarrow T^c \in \mathcal{A}_1 \Longrightarrow T \in \mathcal{A}_2$ by (4.42). But this is again a contradiction because $T \in \mathcal{A}_3$ by hypothesis and $\mathcal{A}_2, \mathcal{A}_3$ are disjoint sets.

So (4.43) is proven. □

Corollary 4.2.26 ($\mathcal{B}$-Partition of $\mathcal{P}_N$). *If S is a proper subset of N, the power set of N, $\mathcal{P}_N$, can be decomposed into the partition:*

$$\mathcal{P}_N = \mathcal{B}_1 \sqcup \mathcal{B}_2 \sqcup \mathcal{B}_3 \sqcup \mathcal{B}_4 \sqcup \mathcal{B}_5 \tag{4.44}$$

where $\sqcup$ indicates disjoint union and

$$\begin{aligned}
\mathcal{B}_1 &:= \{T \subsetneq N \setminus S \mid T \neq \emptyset\} = \text{“proper subsets of } N \setminus S\text{”} \\
\mathcal{B}_2 &:= \{T \cup S \mid T \in \mathcal{A}_1\} \\
\mathcal{B}_3 &:= \{T \subseteq N \mid S \neq T \cap S \neq \emptyset\} \\
\mathcal{B}_4 &:= \{S, N \setminus S\} \\
\mathcal{B}_5 &:= \{\emptyset, N\}.
\end{aligned}$$

Proof. From $\mathcal{A}$-Partition given by Lemma 4.2.20, observe that $S \in \mathcal{A}_2$, $N \setminus S \in \mathcal{A}_1$, $\emptyset \in \mathcal{A}_1$, and $N \in \mathcal{A}_2$. Therefore, define:

(i) $\mathcal{B}_1 = \mathcal{A}_1 \setminus \{N \setminus S, \emptyset\}$

(ii) $\mathcal{B}_2 = \mathcal{A}_2 \setminus \{S, N\}$

(iii) $\mathcal{B}_3 = \mathcal{A}_3$

(iv) $\mathcal{B}_4 = \{S, N \setminus S\}$

(v) $\mathcal{B}_5 = \{\emptyset, N\}$

and $\mathcal{B}_{i=1,\ldots,5}$ is a new partition of $\mathcal{P}_N$. □

Notation 4.2.27. The notation of the following sections assumes that $G = (V, E)$ is a game graph having coalition as edges (Definition 4.2.1), and $e = |E|$. It fixes a total ordering of the subset elements in $\mathcal{P}_N$, which are the nodes of V. The ordering is induced (i) by the size of the set elements and (ii) by the natural ordering of $(\{1, \ldots, N\}, \leq)$. For example, the subsets of $\{1, 2, 3\}$ are sorted in this order (without parenthesis for the ease of notation): $\emptyset < 1 < 2 < 3 < 12 < 13 < 23 < 123$. Note that the fixed ordering is compatible with the partial ordering of inclusion on $\mathcal{P}_N$, that is, if $A \subseteq B \in \mathcal{P}_N$, then $A \leq B$. Consequently, there is an induced lexicographic ordering on E. For example, the edges of $\{1, 2\}$ are sorted in this order: $\{(\emptyset, 1) < (\emptyset, 2) < (\emptyset, 12) < (1, 12) < (2, 12)$. In addition, the following sections assume the canonical basis of $l^2(V)$ and $l^2(E)$ according to fixed orderings. Fixing a coalition $S \subseteq N$, $S \neq \emptyset$, denote the matrices associated with the linear operators d, d*, d$_S$ named respectively M_{d}, M_{d^*} and M_{d_S}. Denote with $M^\top$ the transpose of matrix M. Recall that $L = M_{\mathrm{d}^*} M_{\mathrm{d}}$, $L_S = M_{\mathrm{d}^*} M_{\mathrm{d}_S}$ and $M_{\mathrm{d}^*} = M_{\mathrm{d}}^\top$. Moreover, observe that the rows of $M_{\mathrm{d}}, M_{\mathrm{d}_S}$ are indexed by edges, their columns are indexed by nodes, that is, by coalitions; instead, both the rows and columns of the Laplacian matrices L and L_S are indexed by nodes, that is $L, L_S \in \mathbb{R}^{n \times n}$. In particular, let $R, C \subseteq N$ and let $A \subsetneq B \subseteq N$, that is, $(A, B) \in E$.

(i) If $M \in \mathbb{R}^{e \times n}$, $M[(A, B), :]$ denotes the row corresponding to the (A, B)-th basis element on $l^2(E)$, and $M[:, C]$ denotes the column corresponding to the C-th basis element on $l^2(V)$;

(ii) If $L \in \mathbb{R}^{n \times n}$, $L[R, :]$ denotes the row corresponding R-th basis element of $l^2(V)$, while $L[:, C]$ denotes the column of L corresponding to the C-th basis element of $l^2(V)$;

(iii) If $\boldsymbol{u} \in \mathbb{R}^n$ is a column vector, $\boldsymbol{u}[R]$ denotes the component of $\boldsymbol{u}$ corresponding to the R-th basis element.

4.2.3 Computing the Analytic Formula of $\mathcal{X}$-Shapley

The goal of this section is to derive a value for coalitions as expressed in Remark 4.2.16. The procedure is again inspired by [180], that is, the computations try to solve the normal equation of Proposition 4.2.18. Nonetheless, while [180] uses the inverse matrix of the graph Laplacian of $\overline{G}$ as recalled in Remark 4.1.85, the analogous computation of L^{-1} of the game graph of coalitions G is not straightforward, and it is avoided here. Instead, linear algebra computations of the N-entry of the solution $\boldsymbol{x} \in \mathcal{G}^N$ of Equation (4.38) allow computing the N-entry of the S-differential game u_S.

This section is organized as follows: Remark 4.2.28, Proposition 4.2.29 and Remark 4.2.30 provide a characterization of L_S. Then, Proposition 4.2.31 provides a characterization of L. Furthermore, Lemma 4.2.32 and Lemma 4.2.33 provides

an explicit expression for a specific linear combination of the rows of L_S and L, called $\boldsymbol{q}_S$ and $\boldsymbol{q}$, respectively. Finally, using the computed linear combinations, Theorem 4.2.34 solves Equation (4.38) for the N-entry of the solution, that is, the S-differential game u_S. Equation (4.78) shown by Theorem 4.2.34 is the starting point for a discussion about the suggested Shapley value for coalitions. To conclude, Example 4.2.35 provides computations in the case $n = 3$.

Remark 4.2.28 (Computation of L_N and $L_\emptyset$). Recall Definition 4.2.17 of L_S. Observe that $L_\emptyset = \mathrm{d}^*\mathrm{d}_\emptyset$ is the zero matrix of $\mathbb{R}^{n\times n}$ because $\mathrm{d}_\emptyset = 0$ (Definition 4.2.5 of S-differential). Furthermore, note that L_N has a particular expression. Indeed $L_N = \mathrm{d}^*\mathrm{d}_N$, and Lemma 4.2.7 allows expressing d_N. Now, compute a generic matrix entry $L_N[R, C]$. Fix $R, C \subseteq N$ and consider the element $f_R \in l^2(V)$ of the canonical basis on $l^2(V)$: f_R is such that, for each $T \in V$, $f_R(T) = 1$ if $T = R$ and $f_R(T) = 0$ otherwise. Using Equation (4.19), L_N can be expressed as:

$$L_N[R, C] = \mathrm{d}^*\mathrm{d}_N\, f_R(C) = \sum_{(T,C)\in E} \mathrm{d}_N f_R(T, C) - \sum_{(C,T)\in E} \mathrm{d}_N f_R(C, T)$$

Computing each summand results in:

$$\mathrm{d}_N f_R(T, C) = \begin{cases} f_R(C) - f_R(T) & \text{if } C = N,\ T = \emptyset \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } R = C = N,\ T = \emptyset \\ -1 & \text{if } R = T = \emptyset,\ C = N \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{d}_N f_R(C, T) = \begin{cases} f_R(T) - f_R(C) & \text{if } T = N,\ C = \emptyset \\ 0 & \text{otherwise} \end{cases} = \begin{cases} 1 & \text{if } R = T = N,\ C = \emptyset \\ -1 & \text{if } R = C = \emptyset,\ C = N \\ 0 & \text{otherwise} \end{cases}$$

Therefore, the expression for L_N is:

$$L_N[R, C] = \begin{cases} 1 & \text{if } C = R = N \text{ or } C = R = \emptyset \\ -1 & \text{if } C = \emptyset,\ R = N \text{ or } C = N,\ R = \emptyset \\ 0 & \text{otherwise.} \end{cases} \tag{4.45}$$

Note that L_N is a symmetric matrix, as expected.

Proposition 4.2.29 (Characterization of L_S). *Let S be a proper subset of N, and let $R \subseteq N$.*

(i) The S-column of L_S is such that:

$$L_S[R,\, S] = \begin{cases} -1 & \textit{if } R = \emptyset \\ 1 & \textit{if } R = S \\ 0 & \textit{otherwise.} \end{cases}$$

(ii) The N-column of L_S is such that:

$$L_S[R,\, N] = \begin{cases} -1 & \text{if } R = N \setminus S \\ 1 & \text{if } R = N \\ 0 & \text{otherwise.} \end{cases}$$

(iii)

$$L_S[:,\, C] = -L_S[:, C \cup S] \qquad \forall C \subseteq N \setminus S \tag{4.46}$$

$$L_S[:,\, C'] = \mathbf{0} \in \mathbb{R}^n \qquad \forall C' \subseteq N \text{ such that } S \neq C' \cap S \neq \emptyset \tag{4.47}$$

Proof. Before proving the claims one by one, make some preliminary observations.

Preliminary: properties of M_{d_S}. Consider M_{d_S} as in Notation 4.2.27. Observe that for a given $C \subseteq N \setminus S$:

$$M_{\mathrm{d}_S}[:,\, C] = \Big[0, \dots, 0, \underset{(C, C\cup S)}{-1}, 0, \dots, 0\Big]^\top \tag{4.48}$$

$$M_{\mathrm{d}_S}[:,\, C \cup S] = \Big[0, \dots, 0, \underset{(C, C\cup S)}{1}, 0, \dots, 0\Big]^\top \tag{4.49}$$

because the edge $(C, C \cup S) \in E$ is the unique edge starting from C whose $\mathrm{d}_S u$ is not identically zero, and $\mathrm{d}_S u\,(C, C \cup S) = \mathrm{d}u\,(C, C \cup S) = u(C \cup S) - u(C)$.

In particular, if $C = \emptyset$:

$$M_{\mathrm{d}_S}[:,\, \emptyset] = \Big[0, \dots, 0, \underset{(\emptyset, S)}{-1}, 0, \dots, 0\Big]^\top \tag{4.50}$$

$$M_{\mathrm{d}_S}[:,\, S] = \Big[0, \dots, 0, \underset{(\emptyset, S)}{1}, 0, \dots, 0\Big]^\top \tag{4.51}$$

while if $C = N \setminus S$:

$$M_{\mathrm{d}_S}[:,\, N \setminus S] = \Big[0, \dots, 0, \underset{(N\setminus S, N)}{-1}, 0, \dots, 0\Big]^\top \tag{4.52}$$

$$M_{\mathrm{d}_S}[:,\, N] = \Big[0, \dots, 0, \underset{(N\setminus S, N)}{1}, 0, \dots, 0\Big]^\top \tag{4.53}$$

Instead, given $C' \subseteq N$ such that $C' \cap S \neq \emptyset$ and $C' \cap S \neq S$:

$$M_{\mathrm{d}_S}[:,\, C'] = \mathbf{0} \in \mathbb{R}^e. \tag{4.54}$$

Indeed, if $C' \cap S \neq \emptyset$, then C' is not a subset of $N \setminus S$, so there is no edge in E starting from C'. In addition, if $C \cap S \neq S$, then C' contains S, so there is no edge in E ending in C'.

Preliminary: properties of M_{d^*}. Consider now $M_{\mathrm{d}^*} = M_{\mathrm{d}}^\top$ as in Notation 4.2.27. Observe that:

$$M_{\mathrm{d}}^\top[\emptyset, :] = \Big[*, \dots, *, \underset{(\emptyset,S)}{-1}, *, \dots, *\Big] \tag{4.55}$$

$$M_{\mathrm{d}}^\top[S, :] = \Big[*, \dots, *, \underset{(\emptyset,S)}{1}, *, \dots, *\Big] \tag{4.56}$$

$$M_{\mathrm{d}}^\top[N \setminus S, :] = \Big[*, \dots, *, \underset{(N\setminus S,N)}{-1}, *, \dots, *\Big] \tag{4.57}$$

$$M_{\mathrm{d}}^\top[N, :] = \Big[*, \dots, *, \underset{(N\setminus S,N)}{1}, *, \dots, *\Big] \tag{4.58}$$

where $*$ means a generic real value.

Instead, if $R \subseteq N$ s.t. $R \notin \{N \setminus S, N\}$ and if $R' \subseteq N$ s.t. $R \notin \{\emptyset, S\}$

$$M_{\mathrm{d}}^\top[R, :] = \Big[*, \dots, *, \underset{(N\setminus S,N)}{0}, *, \dots, *\Big] \tag{4.59}$$

$$M_{\mathrm{d}}^\top[R', :] = \Big[*, \dots, *, \underset{(\emptyset,S)}{0}, *, \dots, *\Big] \tag{4.60}$$

because the edge $(N\setminus S, N) \in E$ does not start nor end in R, and the edge $(\emptyset, S) \in E$ does not start nor end in R'.

Proof of (i). Now consider the matrix L_S and its column $L_S[:, S]$.

$$\begin{aligned}
L_S[S, S] = M_{\mathrm{d}}^\top[S, :]\, M_{\mathrm{d}_S}[:, S] = 1 \qquad & \text{by (4.51) and (4.56)} \\
L_S[\emptyset, S] = M_{\mathrm{d}}^\top[\emptyset, :]\, M_{\mathrm{d}_S}[:, S] = -1 \qquad & \text{by (4.51) and (4.55)} \\
L_S[R', S] = M_{\mathrm{d}}^\top[R', :]\, M_{\mathrm{d}_S}[:, S] = 0 \qquad & \text{by (4.51), (4.60).}
\end{aligned}$$

Therefore:

$$L_S[:, S] = \Big[\underset{\emptyset}{-1}, 0, \dots, 0, \underset{S}{1}, 0, \dots, 0\Big]^\top, \tag{4.61}$$

so the first assertion of the lemma is proved.

Proof of (ii). Analogously, consider the column $L_S[:, N]$.

$$\begin{aligned}
L_S[N, N] = M_{\mathrm{d}}^\top[N, :]\, M_{\mathrm{d}_S}[:, N] = 1 \qquad & \text{by (4.53) and (4.58)} \\
L_S[N \setminus S, N] = M_{\mathrm{d}}^\top[N \setminus S, :]\, M_{\mathrm{d}_S}[:, N] = -1 \qquad & \text{by (4.52) and (4.58)} \\
L_S[R, N] = M_{\mathrm{d}}^\top[R, :]\, M_{\mathrm{d}_S}[:, N] = 0 \qquad & \text{by (4.53), (4.59).}
\end{aligned}$$

Therefore:

$$L_S[:, N] = \Big[0, \dots, 0, \underset{N\setminus S}{-1}, 0, \dots, 0, \underset{N}{1}, 0, \dots, 0\Big]. \tag{4.62}$$

□

Proof of (iii). By Equation (4.48) and (4.49), it holds that

$$\forall C \subseteq N \setminus S \qquad M_{\mathrm{d}_S}[:, C] = -M_{\mathrm{d}_S}[:, C \cup S]. \tag{4.63}$$

Hence,

$$\begin{aligned} L_S[:, C] &= M_{\mathrm{d}}^\top \, M_{\mathrm{d}_S}[:, C] && \text{by (4.63)} \\ &= -M_{\mathrm{d}}^\top \, M_{\mathrm{d}_S}[:, C \cup S] \\ &= -L_S[:, C \cup S]. \end{aligned}$$

In addition, by Equation (4.54), it holds that $\forall C' \subseteq N$ such that $\emptyset \neq C' \cap S \neq S$:

$$L_S[:, C'] = M_{\mathrm{d}}^\top \, M_{\mathrm{d}_S}[:, C'] = \mathbf{0} \in \mathbb{R}^n.$$

Remark 4.2.30. L_S is a symmetric matrix because it is the graph Laplacian of the subgraph given by the edges $(T, T \cup S) \in E$, for each $T \subseteq N \setminus S$. Therefore, from Equation (4.61) and (4.62), it holds the same for the rows

$$L_S[S, :] = \Big[0, \dots, 0, \underset{\emptyset}{-1}, 0, \dots, 0, \underset{S}{1}, 0, \dots, 0\Big]. \tag{4.64}$$

$$L_S[N, :] = \Big[0, \dots, 0, \underset{N \setminus S}{-1}, 0, \dots, 0, \underset{N}{1}, 0, \dots, 0\Big]. \tag{4.65}$$

In particular, given a game $\boldsymbol{u} \in \mathcal{G}^N$:

$$L_S[S, :] \, \boldsymbol{u} = \boldsymbol{u}[S] - \boldsymbol{u}[\emptyset], \tag{4.66}$$

$$L_S[N, :] \, \boldsymbol{u} = \boldsymbol{u}[N] - \boldsymbol{u}[N \setminus S]. \tag{4.67}$$

From Equation (4.46) and (4.47), it holds also that:

$$L_S[R, :] = -L_S[R \cup S, :] \qquad \forall R \subseteq N \setminus S, \tag{4.68}$$

$$L_S[R', :] = \mathbf{0} \in \mathbb{R}^n \qquad \forall R' \subseteq N \text{ such that } \emptyset \neq R' \cap S \neq S. \tag{4.69}$$

Proposition 4.2.31 (Characterization of L).

$$\forall R, C \subseteq N, \quad L[R, C] = \begin{cases} \deg R & \textit{if } R = C \\ -1 & \textit{if } R \subsetneq C \textit{ or } C \subsetneq R \\ 0 & \textit{otherwise.} \end{cases} \tag{4.70}$$

Proof. Fix a row R and a column C corresponding to subsets of N:

$$L[R, C] = \left(M_{\mathrm{d}}^\top M_{\mathrm{d}}\right)[R, C] = M_{\mathrm{d}}^\top[R, :] \, M_{\mathrm{d}}[:, C].$$

Given a generic $(A, B) \in E$:

$$M_{\mathrm{d}}^{\top}[R, (A,B)] = \begin{cases} 1 & \text{if } B = R, \text{ i.e. } A \subsetneq B = R \\ -1 & \text{if } A = R, \text{ i.e. } A = R \subsetneq B. \\ 0 & \text{otherwise} \end{cases} \tag{4.71}$$

$$M_{\mathrm{d}}[(A,B), C] = \begin{cases} 1 & \text{if } B = C, \text{ i.e. } A \subsetneq B = C \\ -1 & \text{if } A = C, \text{ i.e. } A = C \subsetneq B \\ 0 & \text{otherwise.} \end{cases} \tag{4.72}$$

If $R = C$, then:

$$\begin{aligned} L[R,R] &= M_{\mathrm{d}}^{\top}[R,:]\, M_{\mathrm{d}}[:,R] = \\ &= \sum_{T \subseteq N \,|\, (T,R) \in E \text{ or } (R,T) \in E} 1. \end{aligned}$$

In other words, $L[R,R]$ represents the number of edges connected to R in the game graph of coalitions G, so

$$L[R,R] = \deg R.$$

If $R \subsetneq C$, then the components of $M_{\mathrm{d}}^{\top}[R,:]$ and $M_{\mathrm{d}}[:,C]$ are both nonzero only for the entry $(A,B) = (R,C)$. Thus, by Equation (4.71) and (4.72):

$$L[R,C] = M_{\mathrm{d}}^{\top}[R,(R,C)]\, M_{\mathrm{d}}[(R,C),C] = -1.$$

By the same argument, if $C \subsetneq R$:

$$L[R,C] = M_{\mathrm{d}}^{\top}[R,(C,R)]\, M_{\mathrm{d}}[(C,R),C] = -1.$$

Otherwise, there are no edges of $(A,B) \in E$ such that both $M_{\mathrm{d}}^{\top}[R,(A,B)]$ and $M_{\mathrm{d}}[(A,B),C]$ are nonzero. Thus, in this case:

$$L[R,C] = 0.$$

□

Lemma 4.2.32 (Computing $\boldsymbol{q}_S$). *Given a coalition $S \subseteq N$, define the column vector $\boldsymbol{q}_S \in \mathbb{R}^n$ as:*

$$\boldsymbol{q}_S := \sum_{T \in \mathcal{P}_N \setminus \{\emptyset, N\}} L_S[T,:]^{\top} + 2L_S[N,:]^{\top}. \tag{4.73}$$

Then,

$$\boldsymbol{q}_S^{\top}\, \boldsymbol{u} = \boldsymbol{u}[N] - \boldsymbol{u}[N \setminus S] + \boldsymbol{u}[S] - \boldsymbol{u}[\emptyset] \tag{4.74}$$

Proof. Recall Remark 4.2.28 expressing L_N and $L_\emptyset$. Consider the case $S = \emptyset$. Then, check the identity (4.74):

$$\boldsymbol{q}_S = \mathbf{0} = \boldsymbol{u}[N] - \boldsymbol{u}[N] + \boldsymbol{u}[\emptyset] - \boldsymbol{u}[\emptyset].$$

Furthermore, consider the case $S = N$. Then, check again the identity (4.74):

$$\sum_{T \in \mathcal{P}_N \setminus \{\emptyset, N\}} L_S[T, :]^\top + 2L_S[N, :]^\top = \left[\underset{\emptyset}{0}, 0, \ldots, 0, \underset{N}{2}\right]^\top$$

Therefore,

$$\boldsymbol{q}_S^\top \boldsymbol{u} = 2\boldsymbol{u}[N] = \boldsymbol{u}[N] - \boldsymbol{u}[\emptyset] + \boldsymbol{u}[N] - \boldsymbol{u}[\emptyset].$$

Now, consider a proper coalition $S \notin \{\emptyset, N\}$ and recall the $\mathcal{B}$-Partition given by Lemma 4.2.26. Observe that $\mathcal{P}_N \setminus \{\emptyset, N\} = \bigsqcup_{j=1}^4 \mathcal{B}_j$. Focus on the sets $\mathcal{B}_1$ and $\mathcal{B}_2$. By Equation (4.68), it holds that:

$$\sum_{\mathcal{B}_1 \sqcup \mathcal{B}_2} L_S[T, :] = \sum_{T \subsetneq N \setminus S, T \neq \emptyset} L_S[T, :] + \sum_{T \subsetneq N \setminus S, T \neq \emptyset} L_S[T \cup S, :] = \mathbf{0} \in \mathbb{R}^n$$

Consider the set $\mathcal{B}_3$. By Equation (4.69),

$$\sum_{\mathcal{B}_3} L_S[T, :] = \sum_{T \subseteq N, S \neq T \cap S \neq \emptyset} L_S[T, :] = \mathbf{0}.$$

Then,

$$\sum_{\mathcal{B}_1 \sqcup \mathcal{B}_2 \sqcup \mathcal{B}_3 \sqcup \mathcal{B}_4} L_S[T, :] = \sum_{\mathcal{B}_4} L_S[T, :] = L_S[S, :] + L_S[N \setminus S, :]$$

So, $\boldsymbol{q}_S$ becomes:

$$\begin{aligned} \boldsymbol{q}_S &= L_S[S, :]^\top + L_S[N \setminus S, :]^\top + 2L_S[N, :]^\top = \\ &= L_S[S, :]^\top + L_S[N, :]^\top \quad \text{because } (N \setminus S) \sqcup S = N, \text{ using (4.68).} \end{aligned}$$

Finally, by Equation (4.66) and (4.67),

$$\boldsymbol{q}_S^\top \boldsymbol{u} = \boldsymbol{u}[N] - \boldsymbol{u}[N \setminus S] + \boldsymbol{u}[S] - \boldsymbol{u}[\emptyset]$$

□

Lemma 4.2.33 (Computing q). *Define the column vector $\boldsymbol{q} \in \mathbb{R}^n$ as*

$$\boldsymbol{q} := \sum_{T \in \mathcal{P}_N \setminus \{\emptyset, N\}} L[T, :]^\top + 2L[N, :]^\top. \tag{4.75}$$

Then

$$\boldsymbol{q}[T] := \begin{cases} -1 - \deg N & \text{if } T = \emptyset \\ 1 + \deg N & \text{if } T = N \\ 0 & \text{otherwise.} \end{cases} \tag{4.76}$$

Proof. Fix a column $C \in N$ and recall as in Remark 4.1.64 that the sum of the rows (or columns) of a graph Laplacian is equal to zero, so

$$\mathbf{0} = \sum_{T \in \mathcal{P}_N} L[T, C] = \sum_{T \in \mathcal{P}_N \setminus \{\emptyset, N\}} L[T,\, C] + L[N, C] + L[\emptyset, C].$$

Then, it follows:

$$\sum_{T \in \mathcal{P}_N \setminus \{\emptyset, N\}} L[T,\, C] = -L[N, C] - L[\emptyset, C] = \qquad \text{using (4.70)}$$

$$= \begin{cases} 1 - \deg N & \text{if } C = N \text{ or } C = \emptyset \\ +2 & \text{otherwise.} \end{cases} \tag{4.77}$$

Thus, again by the characterization of L of (4.70), Equation (4.75) becomes:

$$\begin{aligned} \boldsymbol{q} = \Big[& \underset{\emptyset}{1 - \deg N}, & +2, & & +2, & & \ldots, & & +2, & & \underset{N}{1 - \deg N} \Big]^\top + \\ + 2 \Big[& \underset{\emptyset}{-1}, & -1, & & -1, & & \ldots, & & -1, & & \underset{N}{\deg N} \Big]^\top = \\ = \Big[& \underset{\emptyset}{-1 - \deg N}, & 0, & & 0, & & \ldots, & & 0, & & \underset{N}{1 + \deg N} \Big]^\top \end{aligned}$$

Therefore, Equation (4.76) is proven. □

Theorem 4.2.34 (Solving $L\,\boldsymbol{x} = L_S\,\boldsymbol{u}$ for the N-entry of $\boldsymbol{x}$). *If $\boldsymbol{x} \in \mathcal{G}^N$ is the unique solution of $L\,\boldsymbol{x} = L_S\,\boldsymbol{u}$ of Equation* (4.38)*, then the N-entry of $\boldsymbol{x}$ is:*

$$\boldsymbol{x}[N] = \frac{1}{2^n}\Big(\boldsymbol{u}[N] - \boldsymbol{u}[N \setminus S] + \boldsymbol{u}[S]\Big). \tag{4.78}$$

Proof. Observe that the column vectors $\boldsymbol{q}$, defined by Equation (4.75), and $\boldsymbol{q}_S$, defined by Equation (4.73), are the same linear combination of the rows of the matrices L and L_S, respectively. Then, they represent elementary row operations on the system of linear equations (4.38). Therefore, the solution of (4.38) should satisfy the following:

$$\boldsymbol{q}^\top \boldsymbol{x} = \boldsymbol{q}_S^\top \boldsymbol{u}. \tag{4.79}$$

Using (4.76) and (4.74), Equation (4.79) becomes:

$$\Big(-\deg N - 1\Big)\,\boldsymbol{x}[\emptyset] + \Big(\deg N + 1\Big)\,\boldsymbol{x}[N] = \boldsymbol{u}[N] - \boldsymbol{u}[N \setminus S] + \boldsymbol{u}[S] - \boldsymbol{u}[\emptyset]. \tag{4.80}$$

Now, impose $\boldsymbol{x}[\emptyset] = 0$ because $\boldsymbol{x} \in \mathcal{G}^N$. Similarly, $\boldsymbol{u}[\emptyset] = 0$. From Lemma 4.2.3, remember $\deg N = 2^n - 1$. Therefore, Equation (4.80) becomes:

$$\Big(2^n - 1 + 1\Big)\boldsymbol{x}[N] = \boldsymbol{u}[N] - \boldsymbol{u}[N \setminus S] + \boldsymbol{u}[S]$$

$$\boldsymbol{x}[N] = \frac{1}{2^n}\Big(\boldsymbol{u}[N] - \boldsymbol{u}[N \setminus S] + \boldsymbol{u}[S]\Big).$$

□

Example 4.2.35 (Solution of Equation (4.38) for $n = 3$). Suppose $n = 3$. Consider the basis of games τ_S of Definition 4.1.33.
We can express $\boldsymbol{u} = \left[u_\emptyset, u_{\{1\}}, u_{\{2\}}, u_{\{3\}}, u_{\{12\}}, u_{\{13\}}, u_{\{23\}}, u_{\{123\}}\right]^\top$.
By Proposition 4.2.31, we can compute L:

$$L = \begin{bmatrix} 7 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 4 & 0 & 0 & -1 & -1 & 0 & -1 \\ -1 & 0 & 4 & 0 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & 4 & 0 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & 4 & 0 & 0 & -1 \\ -1 & -1 & 0 & -1 & 0 & 4 & 0 & -1 \\ -1 & 0 & -1 & -1 & 0 & 0 & 4 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & 7 \end{bmatrix}$$

The computations of Remark 4.2.28 provides $L_\emptyset$ and $L_{\{123\}}$. To compute L_S for $S = \{1\}$ and $S = \{12\}$, we use Proposition 4.2.29.

$$L_{\{1\}}\boldsymbol{u} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_\emptyset \\ u_{\{1\}} \\ u_{\{2\}} \\ u_{\{3\}} \\ u_{\{12\}} \\ u_{\{13\}} \\ u_{\{23\}} \\ u_{\{123\}} \end{bmatrix} = \begin{bmatrix} -u_{\{1\}} + u_\emptyset \\ u_{\{1\}} - u_\emptyset \\ -u_{\{12\}} + u_{\{2\}} \\ -u_{\{13\}} + u_{\{3\}} \\ u_{\{12\}} - u_{\{2\}} \\ u_{\{13\}} - u_{\{3\}} \\ -u_{\{123\}} + u_{\{23\}} \\ u_{\{123\}} - u_{\{23\}} \end{bmatrix}$$

$$L_{\{12\}}\boldsymbol{u} = \begin{bmatrix} 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{\emptyset} \\ u_{\{1\}} \\ u_{\{2\}} \\ u_{\{3\}} \\ u_{\{12\}} \\ u_{\{13\}} \\ u_{\{23\}} \\ u_{\{123\}} \end{bmatrix} = \begin{bmatrix} -u_{\{12\}} + u_{\emptyset} \\ 0 \\ 0 \\ -u_{\{123\}} + u_{\{3\}} \\ u_{\{12\}} - u_{\emptyset} \\ 0 \\ 0 \\ u_{\{123\}} - u_{\{3\}} \end{bmatrix}$$

For the remaining proper subsets of N, that is $S \in \{2,3,13,23\}$, L_S can be derived by an index permutation of the results obtained for $L_{\{1\}}$ or $L_{\{12\}}$, according to the corresponding subset cardinality. Then, the following are example computations of $\boldsymbol{q}$ and $\boldsymbol{q}_S\boldsymbol{u}$ that allow computing $\boldsymbol{x}[N]$ by Equation (4.78), after imposing $\boldsymbol{x}[\emptyset] = 0$.

$$\begin{aligned} \boldsymbol{q} &= \sum_{T \in \mathcal{P}_N \setminus \{\emptyset, N\}} L[T, :]^\top + 2L[N, :]^\top \\ &= \big[-8, 0, 0, 0, 0, 0, 0, 8 \big]^\top. \end{aligned}$$

$$\begin{aligned} \boldsymbol{q}_{\{1\}}^\top \boldsymbol{u} &= \Big(\sum_{T \in \mathcal{P}_N \setminus \{\emptyset, N\}} L_{\{1\}}[T, :] + 2L_{\{1\}}[N, :] \Big)^\top \boldsymbol{u} \\ &= u_{\{123\}} - u_{\{23\}} + u_{\{1\}} - u_{\{\emptyset\}}. \end{aligned}$$

$$\begin{aligned} \boldsymbol{q}_{\{12\}}^\top \boldsymbol{u} &= \Big(\sum_{T \in \mathcal{P}_N \setminus \{\emptyset, N\}} L_{\{12\}}[T, :] + 2L_{\{12\}}[N, :] \Big)^\top \boldsymbol{u} \\ &= u_{\{123\}} - u_{\{3\}} + u_{\{12\}} - u_{\{\emptyset\}}. \end{aligned}$$

4.3 Alternative Characterizations of $\mathcal{X}$-Shapley

This section looks at $\mathcal{X}$-Shapley from a cooperative game theory viewpoint. In particular, Section 4.3.1 suggests a formal definition of the game map, that is, the concept similar to the value allocation (Definition 4.1.11) when focusing on coalition allocations. The section culminates in the proof of Theorem 4.3.14, which provides an axiomatic characterization of $\mathcal{X}$-Shapley that makes the pair with the classical Theorem 4.1.43. Section 4.3.2 suggests a counter-objections characterization of $\mathcal{X}$-Shapley: inspired by the balanced contribution property for players (Definition 4.1.45), Theorem 4.3.18 derives a uniqueness result for the balanced contribution

property for coalitions (Definition 4.3.16), holding for $\mathcal{X}$-Shapley. Finally, Section 4.3.3 discusses other properties of the $\mathcal{X}$-Shapley, focusing on its scaling factor and the connections with the Shapley value ϕ.

4.3.1 Axiomatic Characterization

This section provides a characterization of the coalitional value expressed by the formula of Equation (4.78) that is inspired by the one recalled in Section 4.1.4 for the Shapley value. First, a new definition of allocation to coalitions is given, called game map. Second, it recalls definitions about properties of coalitions in a game. Third, it defines the axioms for a fairness concept about coalition allocation. Finally, Theorem 4.3.14 shows a uniqueness result for game maps satisfying an axiomatic fairness property that pairs with Theorem 4.1.43 for the Shapley value.

Definition 4.3.1 (Game map). *A game map is a function* $\gamma : \mathcal{G}^N \to \mathcal{G}^N$.

Remark 4.3.2. To the best of the author knowledge, the definition of game map provided here is a novel definition. For coalition allocation, the game map have the same role of the concept of value for players (Definition 4.1.11). An alternative choice for the codomain of the game map would be $\Xi^N = \{\xi \colon \mathcal{P}_N \to \mathbb{R}\}$, that is, the set of set functions [89]. Compared to Ξ^N, the choice of codomain $\mathcal{G}^N$ assumes implicitly that the image of a game map $v = \gamma(u)$ evaluated on the empty set should be zero, that is, $v(\emptyset) = 0$. This assumption is straightforward, because if γ is interpreted as an allocation to coalitions, then the worth allotted to the empty coalition should be assumed zero.

Notation 4.3.3. If γ is a game map and $u \in \mathcal{G}^N$, denote by $\gamma_S(u) := \Big(\gamma(u)\Big)(S)$. In what follows, $\gamma_S := \gamma_S(u)$, that is, the dependency over u is omitted to simplify the notation.

Definition 4.3.4 (Null coalition). *Given* $u \in \mathcal{G}^N$, $S \subseteq N$ *is said a null coalition if,* $\forall R \subseteq (N \setminus S),\ u(R \cup S) = u(R)$

Definition 4.3.5 (Dummy coalition). *Given* $u \in \mathcal{G}^N$, $S \subseteq N$ *is said a dummy coalition if,* $\forall R \subseteq (N \setminus S),\ u(R \cup S) - u(R) = u(S)$

Definition 4.3.6 (Symmetric coalitions). *Consider a game* u, *and* $\sigma \in \mathcal{S}_N$ *such that the permuted game* $\sigma^*(u) = u$. *Two coalitions* $S, T \subseteq N$ *are said symmetric if* $\sigma(S) = T$, *and* $\sigma(T) = S$.

Definition 4.3.7 (Axioms for game maps). *The game map* $\gamma \colon \mathcal{G}^N \to \mathcal{G}^N$ *is said to satisfy one of the following axioms if the corresponding statement holds:*

EFF_S *Efficiency:* $\forall u \in \mathcal{G}^N$, $\sum_{S \subseteq N} \gamma_S(u) = u(N)$.

NLL_S *Null Coalition:* $\forall u \in \mathcal{G}^N$, *if* $S \subseteq N$ *is a null coalition for* u, *then* $\gamma_S(u) = 0$.

SYM_S *Symmetry:* $\forall u \in \mathcal{G}^N$, *given* $S, T \subseteq N$ *symmetric coalitions for* u, $\gamma_S(u) = \gamma_T(u)$.

LIN_S *Linearity:* $\forall u, u' \in \mathcal{G}^N,\ \forall a, a' \in \mathbb{R}, \quad \gamma(au + a'u) = a\gamma(u) + a'\gamma(u')$.

CS_S *Constant sum:* $\forall u \in \mathcal{G}^N \quad \gamma_S(u) + \gamma_{N\setminus S}(u) = \gamma_N(u)$

BLT_S *Bilaterality:* $\forall u \in \mathcal{G}^N$, *given a bilateral coalition* $S \subseteq N$ *for* u *(Definition 4.1.24), then*

$$\gamma_S(u) = \gamma_{N\setminus S}(u)$$

Remark 4.3.8. An equivalent formulation for the null coalition axiom is the following: $\forall T \subseteq N$, $u(T \cup S) = u(T \setminus S) \implies \gamma_S(u) = 0$. Indeed, if $R = T \setminus S$, any $T \subseteq N$ can be expressed as $T = R \cup (T \cap S)$, obtaining $R \sqcup S = T \cup S$, and concluding:

$$\forall R \subseteq N \setminus S,\, u(R \cup S) = u(R) \iff \forall T \subseteq N,\, u(T \cup S) = u(T \setminus S).$$

Note that the coalition $\emptyset$ is both a null and dummy coalition.

Definition 4.3.9 ($\mathcal{X}$-Shapley). *Given* $u \in \mathcal{G}^N$, *define the function* $\mathcal{X}$, *called also* $\mathcal{X}$*-Shapley or coalitional Shapley, as*

$$\forall S \subseteq N, \quad \mathcal{X}_S(u) := \frac{1}{2^n}\Big(u(N) - u(N \setminus S) + u(S)\Big) \tag{4.81}$$

Proposition 4.3.10 (Properties of $\mathcal{X}$-Shapley). *For each* $u \in \mathcal{G}^N$*:*

(i) $\mathcal{X}$ *is a game map.*

(ii)

$$\forall u \in \mathcal{G}^N, \quad \mathcal{X}_N(u) = \frac{1}{2^{n-1}}u(N). \tag{4.82}$$

(iii) Denoting by $\bar{u}$ *the dual game of Definition 4.1.21, then*

$$\forall u \in \mathcal{G}^N, \quad \mathcal{X}(u) = \frac{1}{2^n}(\bar{u} + u) \tag{4.83}$$

Proof. Prove the single statements in sequence.

Proof of (i). To show the claim, it is sufficient to show that $\forall u \in \mathcal{G}^N$, $\Big(\mathcal{X}_S(u)\Big)_{S\subseteq N}$ is a game itself, that is, $\mathcal{X}_\emptyset(u) = 0$:

$$\mathcal{X}_\emptyset(u) = \frac{1}{2^n}\Big(u(N) - u(N) + u(0)\Big) = 0$$

Proof of (ii).

$$\forall u \in \mathcal{G}^N, \quad \mathcal{X}_N(u) = \frac{1}{2^n}(u(N) - u(\emptyset) + u(N)) = \frac{2u(N)}{2^n} = \frac{1}{2^{n-1}}u(N)$$

Proof of (iii). Follow from the definition of $\bar{u}$ and $\mathcal{X}$:

$$\forall u \in \mathcal{G}^N, \forall S \subseteq N, \quad \mathcal{X}_S(u) = \frac{1}{2^n}(u(N) - u(N \setminus S) + u(S)) = \frac{1}{2^n}(\bar{u}(S) + u(S))$$

□

Proposition 4.3.11 ($\mathcal{X}$-Shapley satisfies the axioms for game maps). *$\forall u \in \mathcal{G}^N$, the game map $\mathcal{X}$ satisfies the axioms CS_S, EFF_S, NLL_S, SYM_S, LIN_S, and BLT_S.*

Proof. Show the axioms for $\mathcal{X}$ one by one.

$\mathbf{CS}_S$ For a given game u, compute $\mathcal{X}_S + \mathcal{X}_{N\setminus S}$:

$$\begin{aligned}\mathcal{X}_S(u) + \mathcal{X}_{N\setminus S}(u) &= \frac{1}{2^n}(u(N) - u(N \setminus S) + u(S) + u(N) - u(S) + u(N \setminus S)) \\ &= \frac{1}{2^{n-1}}u(N) \overset{(4.82)}{=} \mathcal{X}_N(u)\end{aligned}$$

$\mathbf{EFF}_S$. For each game u:

$$\begin{aligned}2\sum_{S\subseteq N}\mathcal{X}_S(u) &= \sum_{S\subseteq N}\mathcal{X}_S(u) + \sum_{S\subseteq N}\mathcal{X}_{N\setminus S}(u) = && \text{bijection } S \Longleftrightarrow N \setminus S \\ &= \sum_{S\subseteq N}\Big(\mathcal{X}_S(u) + \mathcal{X}_{N\setminus S}(u)\Big) = \\ &= \sum_{S\subseteq N}\mathcal{X}_N(u) && \text{By CS}_S \text{ for } \mathcal{X} \\ &= 2^n\frac{1}{2^{n-1}}u(N) && \text{By Equation (4.82)} \\ &\Longrightarrow \sum_{S\subseteq N}\mathcal{X}_S(u) = u(N).\end{aligned}$$

$\mathbf{NLL}_S$. For a given game u, fix $S \subseteq N$ null coalition. Choose $T = N \setminus S$ and apply Definition 4.3.4:

$$u(T) = u(N \setminus S) = u\Big((N \setminus S) \cup S\Big) = u(N).$$

Choose $T = \emptyset$ and apply the same definition: $u(T) = u(\emptyset) = u(\emptyset \cup S) = 0$. Finally, apply the definition of $\mathcal{X}$ to S:

$$\mathcal{X}_S = \frac{1}{2^n}(u(N) - u(N \setminus S) + u(S)) = \frac{1}{2^n}(u(N) - u(N) + 0)) = 0.$$

SYM$_S$. Given a game u, let $\sigma \in \mathcal{S}_N$ be such that $\sigma^*(u) = u$ and let $S, T \subseteq N$ be such that $\sigma(S) = T$ and $\sigma(T) = S$. Observe that $\sigma(N \setminus S) = N \setminus T$ and $\sigma(N \setminus S) = N \setminus T$. Therefore:

$$\begin{aligned}\mathcal{X}_S &= \mathcal{X}_{\sigma(T)} \\ &= \frac{1}{2^n}\Big(u(N) - u(\sigma(N \setminus T)) + u(\sigma(T))\Big) = \\ &= \frac{1}{2^n}\Big(u(N) - \sigma^*(u(N \setminus T)) + \sigma^*(u(T))\Big) = \\ &= \frac{1}{2^n}\Big(u(N) - u(N \setminus T) + u(T)\Big) = \mathcal{X}_T.\end{aligned}$$

LIN$_S$. Fix $S \subseteq N$. Given $u, u' \in \mathcal{G}^N$; $a, a' \in \mathbb{R}$

$$\begin{aligned}\mathcal{X}_S(au + a'u') &= \frac{1}{2^n}(au(N) + a'u'(N) - au(N \setminus S) - a'u'(N \setminus S) + au(S) + a'u'(S)) = \\ &= \frac{1}{2^n}\Big(au(N) - au(N \setminus S) + au(S) + a'u(N) - a'u(N \setminus S) + a'u(S)\Big) \\ &= a\,\mathcal{X}_S + a'\,\mathcal{X}_S\end{aligned}$$

BLT$_S$. If $u \in \mathcal{G}^N$, $S \subseteq N$, and $u(S) = u(N \setminus S)$, then

$$\mathcal{X}_S = \frac{1}{2^n}\Big(u(N) - u(N \setminus S) + u(S)\Big) = \frac{1}{2^n}u(N) = \mathcal{X}_{N \setminus S}.$$

□

Lemma 4.3.12 (About game maps satisfying EFF$_S$ and CS$_S$). *Let γ be a game map. If γ satisfies EFF$_S$ and CS$_S$, then*

$$\forall u \in \mathcal{G}^N, \quad \gamma_N(u) = \frac{1}{2^{n-1}}\,\gamma_N. \tag{4.84}$$

Proof. Recall the definition of constant sum CS$_S$ for γ:

$$\forall S \subseteq N,\ \gamma_S + \gamma_{N \setminus S} = \gamma_N. \tag{4.85}$$

By EFF_S for γ:

$$\begin{aligned}
\sum_{S\subseteq N} \gamma_S(u) &= u(N) \\
\sum_{S\subseteq N} \gamma_S(u) + \sum_{S\subseteq N} \gamma_{N\setminus S}(u) &= 2\,u(N) && \text{bijection } S \longleftrightarrow N\setminus S \\
\sum_{S\subseteq N} \Big(\gamma_S(u) + \gamma_{N\setminus S}(u)\Big) &= 2\,u(N) \\
\sum_{S\subseteq N} \gamma_N &= 2\,u(N) && \text{Use } \text{CS}_S \\
2^n\,\gamma_N &= 2\,u(N) \\
\gamma_N(u) &= \frac{1}{2^{n-1}}\,u(N)
\end{aligned}$$

□

Remark 4.3.13 (BLT_S for $\mathcal{A}_3$). Fix a coalition $S \neq \emptyset$ and consider the basis of unanimity games $\{\theta_S\}_{S\neq\emptyset}$ as in Definition 4.1.33. If $T \subseteq N$, there are two possible cases:

(i) $T \cap S = S \iff T \supseteq S$.

$$\theta_S(T) = 1 \iff T \supseteq S \iff N \setminus T \not\supseteq S \iff \theta_S(N \setminus T) = 0.$$

In particular $\theta_S(T) \neq \theta_S(N \setminus T)$.

(ii) $T \cap S \neq S \iff T \not\supseteq S$, then we have $\theta_S(T) = 0$ and

$$\theta_S(N \setminus T) = \begin{cases} 0 & T \cap S \neq \emptyset \\ 1 & T \cap S = \emptyset \end{cases}$$

In particular, $\theta_S(T) = \theta_S(N \setminus T) \iff T \cap S \neq \emptyset$ and $T \cap S \neq S$. Focusing on the $\mathcal{A}$-Partition of the powerset $\mathcal{P}_N$ defined in Lemma 4.2.20, note that $\mathcal{A}_3 = \{T \subseteq N \mid \theta_S(T) = \theta_S(N \setminus T)\}$. This means that the BLT_S axiom on unanimity games involves all and only the coalitions in $T \in \mathcal{A}_3$.

Theorem 4.3.14 (Axiomatic uniqueness of $\mathcal{X}$-Shapley). *Coalitional Shapley $\mathcal{X}$ is the unique game map satisfying the axioms of EFF_S, BLT_S, NLL_S, LIN_S and CS_S.*

Proof. By Proposition 4.3.11, $\mathcal{X}$ satisfies the listed axioms, so the existence claim holds.

Proof of uniqueness. The uniqueness proof here provided follows the scheme of Theorem 4.1.43. In particular, it asserts that a game map is uniquely determined on a basis of $\mathcal{G}^N$ by the listed axioms.

Consider a generic game map γ. Fix a nonempty coalition $S \neq \emptyset$ and consider θ_S from the basis of unanimity games $(\theta_S)_{S\neq\emptyset}$ of $\mathcal{G}^N$, as in Definition 4.1.33. From Lemma 4.3.12

$$\gamma_N(\theta_S) = \frac{1}{2^{n-1}}\theta_S(N) = \frac{1}{2^{n-1}}.$$

Recall the $\mathcal{A}$-partition of $\mathcal{P}_N$ of Lemma 4.2.20: the following Steps 1-5 are aimed to compute θ_S on each partition subset.

Step 1. Observe that $\mathcal{A}_2 = \{T \cup S \mid T \subseteq N \setminus S\}$ contains all and only the sets containing S; hence, by the definition of θ_S, $\theta_S(T) = 1 \iff T \in \mathcal{A}_2$. Therefore, given a generic $T \subseteq N$, the formula of $\theta_S(T)$ becomes:

$$\theta_S(T) = \begin{cases} 0 & T \in \mathcal{A}_1 \sqcup \mathcal{A}_3 \\ 1 & T \in \mathcal{A}_2 \end{cases} \tag{4.86}$$

Step 2. This step show that, given a coalition $T \in \mathcal{A}_1 = \{T \subseteq N \setminus S\}$, T is a null coalition for θ_S, that is, $\forall R \subseteq N \setminus T,\ \theta_S(R \sqcup T) = \theta_S(T)$. To prove the claim, fix $T \in \mathcal{A}_1$ and take $R \subseteq N \setminus T$, that is equivalent to say $R \in \mathcal{A}_2$ by (4.42) of Corollary 4.2.25. There are two cases:

(i) $R \supseteq S$. Then $\theta_S(R) = 1 = \theta_S(R \cup T)$.

(ii) $R \not\supseteq S$. Then $\theta_S(R) = 0$, and $\exists i \in S$ such that $i \notin R$. Because $T \in \mathcal{A}_1$, then $T \subseteq N \setminus S$, therefore it holds also that $i \notin T$. So $i \notin T \cup R$, that means $T \cup R \not\supseteq S$, so $\theta_S(T \cup R) = \theta_S(R) = 0$.

The above two observations show that, if adding T to R, θ_S does not change. Formally, $\forall T \in \mathcal{A}_1, \forall R \in \mathcal{A}_2$, $\theta_S(T \cup R) = \theta_S(R)$. Therefore, $T \in \mathcal{A}_1$ is a null coalition. Because γ satisfy NLL_S by hypothesis,

$$\forall T \in \mathcal{A}_1, \quad \gamma_T(\theta_S) = 0. \tag{4.87}$$

Step 3. This step shows that, given a coalition $R \in \mathcal{A}_2$, $\gamma_R(\theta_S) = \frac{1}{2^{n-1}}$. Consider $T = N \setminus R$. From Corollary 4.2.25, $T \in \mathcal{A}_1$, and from Equation (4.87), $\gamma_T(\theta_S) = 0$. Because $R \sqcup T = N$, apply CS_S to γ:

$$\gamma_T(\theta_S) + \gamma_{N\setminus T}(\theta_S) \overset{\mathrm{CS}_S}{=} \gamma_N(\theta_S) \overset{(4.84)}{=} \frac{1}{2^{n-1}}$$
$$\gamma_R(\theta_S) = \gamma_{N\setminus T}(\theta_S) = \frac{1}{2^{n-1}}$$

Until this point, $\gamma_T(\theta_S)$ is completely determined if $T \in \mathcal{A}_1 \sqcup \mathcal{A}_2$. By Corollary 4.2.23, γ is completely characterized if $s = 1$, because $s = 1 \implies \mathcal{A}_3 = \emptyset$. In particular, if $n = 2$, γ is completely characterized over all the basis games $\left(\theta_S\right)_{S \neq \emptyset}$.

Step 4. This step shows that $\forall T \in \mathcal{A}_3$, $\gamma_T(\theta_S) = \frac{1}{2^n}$. From what observed above, assume $n > 2$ and $s \geq 1$. Deduce that $\mathcal{A}_3 \neq \emptyset$, then it is possible to fix a $T \in \mathcal{A}_3$. It holds that $T \cap S \neq S \implies T \not\supseteq S \implies \theta_S(T) = 0$. In addition, using Corollary 4.2.25, it holds also that $N \setminus T \in \mathcal{A}_3$, and reasoning as before $\theta_S(N \setminus T) = 0$. Then, apply the axiom BLT_S:

$$\begin{aligned} & \forall T \in \mathcal{A}_3,\ \theta_S(T) = \theta_S(N \setminus T) = 0 \\ \overset{BLT_S}{\implies} \quad & \forall T \in \mathcal{A}_3,\ \gamma_T(\theta_S) = \gamma_{N \setminus T}(\theta_S). \end{aligned} \tag{4.88}$$

Furthermore, the axiom CS_S applied to γ and T says that:

$$\gamma_T(\theta_S) + \gamma_{N \setminus T}(\theta_S) \overset{CS_S}{=} \gamma_N(\theta_S) = \frac{1}{2^{n-1}}. \tag{4.89}$$

Combining Equation (4.88) and (4.89) results in $\gamma_T(\theta_S) = \frac{1}{2^n}$. Now γ is determined over $\mathcal{A}_3$.

Step 5. The previous steps completely characterize γ on each basis game θ_S, for $S \neq \emptyset$. Finally, considering a game map $\overline{\gamma} \neq \gamma$ satisfying all the axioms in the hypothesis, it holds that $\overline{\gamma} = \gamma$ on the basis games $(\theta_S)_{S \neq \emptyset}$, giving a contradiction. In conclusion, the uniqueness result is proven. □

Remark 4.3.15. Notably, the theorem above leverages the axioms of EFF_S, NLL_S, and LIN_S, which are simply a generalization of the same properties for values. Instead, SYM_S is not assumed, although satisfied by $\mathcal{X}$-Shapley, as stated by Proposition 4.3.10. Comparing with Theorem 4.1.43, SYM_S is substituted by CS_S and BIL_S. This discussion suggests a research direction to investigate the actual role of SYM_S.

4.3.2 Counter-objections Characterization

This section shows a stability concept for $\mathcal{X}$-Shapley inspired by the characterization recalled in Section 4.1.5 for the Shapley value ϕ. In this case, the definition of objection or counter-objection derives from the following reasoning: during the negotiation, a coalition S could instantiate an objection against another disjoint coalition T from a candidate allocation γ, that is, a game map. Consequently, the coalition T could counter-object against S. If the objection and counter-objection are not balanced, they cause γ changing to a different $\overline{\gamma}$. Thus, the allocation γ is not stable. Otherwise, if there is a balancing, the allocation γ is stable. The main

result of this section is Theorem 4.3.18: it shows the uniqueness of $\mathcal{X}$ among the class of game maps satisfying balanced contribution property for coalitions.

Fix a game $u \in \mathcal{G}^N$, and recall Definition 4.1.16 of subgame. The following discussion defines the possible objections for a candidate game map γ of a coalition $S \subseteq N$ against $T \subseteq N$, with $S \cap T = \emptyset$, indicating $s := |S|$, $t := |T|$ as usual. Assume that S wants a greater allocation than the current negotiated $\gamma_S(N, u)$, and asks T to give more to S:

1. S threatens to leave the game, so that T would gain less, normalized by 2^{-s}:
$$2^{-s}\,\gamma_T(N \setminus S, u^{N\setminus S}) \leq \gamma_T(N, u); \tag{4.90}$$
2. S threatens to persuade other coalitions disjoint from T to exclude T from the game, so that S would gain more, normalized by 2^{-t}:
$$2^{-t}\,\gamma_S(N \setminus T, u^{N\setminus T}) \geq \gamma_S(N, u). \tag{4.91}$$

Then, the coalition T could counter-object by saying one of the following in response, respectively:

1. T says that if S leaves T would lose, but if T leaves then S would lose at least as much:
$$\gamma_S(N, u) - 2^{-t}\,\gamma_S(N \setminus T, u^{N\setminus T}) \geq \gamma_T(N, u) - 2^{-s}\,\gamma_T(N \setminus S, u^{N\setminus S}); \tag{4.92}$$
2. T says that if it is excluded, S would gain, but if T persuades the other coalitions to exclude S, then T would gain at least as much:
$$2^{-s}\,\gamma_T(N \setminus S, u^{N\setminus S}) - \gamma_T(N, u) \geq 2^{-t}\,\gamma_S(N \setminus T, u^{N\setminus T}) - \gamma_S(N, u). \tag{4.93}$$

The balancing between the statements expressed by Equation (4.90) and (4.92), or Equation (4.91) and (4.93) shows the equilibrium between the negotiation power between of the two disjoint coalitions. Formally, the balancing is expressed by the following definition:

Definition 4.3.16 (Balanced contribution for coalitions - BCP$_S$). *A game map γ satisfies the balanced contribution property for coalitions (BCP$_S$) if, $\forall S, T \subseteq N$ such that $S \cap T = \emptyset$,*

$$\gamma_S(N, u) - 2^{-t}\gamma_S(N \setminus T, u^{N\setminus T}) = \gamma_T(N, u) - 2^{-s}\gamma_T(N \setminus S, u^{N\setminus S}). \tag{4.94}$$

Remark 4.3.17. Consider a game $u \in \mathcal{G}^N$ and the pair of coalitions $(S, T) = (\emptyset, N)$. Then, the identity of (4.94) holds for each game map γ, indeed:

$$\gamma_\emptyset(N, u) - 2^{-n}\gamma_\emptyset(\emptyset, u^\emptyset) = \gamma_N(N, u) - 2^{-0}\gamma_N(N, u).$$

In addition, observe that if a game map γ satisfies BCP$_S$, then also $\overline{\gamma} = a\gamma$ satisfies the same, with $a \in \mathbb{R}$.

Theorem 4.3.18 ($\mathcal{X}$-Shapley and balanced contributions). *$\mathcal{X}$-Shapley is the unique game map satisfying EFF_S and BCP_S.*

Proof. First, prove the existence by showing that $\mathcal{X}$ satisfies the hypothesis; second, prove uniqueness.

Proof of existence. Given a game u, let $S, T \subseteq N$ be such that $S \cap T = \emptyset$. Denote by S^c and T^c the complementary set in N of the two coalitions, respectively. Substitute the formula for $\mathcal{X}$ in Equation (4.94), and ask (symbol $\overset{???}{=}$) if the following identity holds true:

$$\begin{aligned}
&\frac{1}{2^n}\Big(u^N(N) - u^N(N \cap S^c) + u^N(S)\Big) \\
&- \frac{2^{-t}}{2^{n-t}}\Big(u^{N\setminus T}(N \cap T^c) - u^{N\setminus T}(N \cap T^c \cap S^c) + u^{N\setminus T}(S \cap T^c)\Big) \\
&\overset{???}{=} \\
&\frac{1}{2^n}\Big(u^N(N) - u^N(N \cap T^c) + u^N(T)\Big) \\
&- \frac{2^{-s}}{2^{n-s}}\Big(u^{N\setminus S}(N \cap S^c) - u^{N\setminus S}(N \cap S^c \cap T^c) + u^{N\setminus S}(S^c \cap T)\Big).
\end{aligned}$$

By Definition 4.1.16 of subgame:

$$\begin{aligned}
&\frac{1}{2^n}\Big(u^N(N) - u^N(N \cap S^c)) + u^N(S)\Big) \\
&- \frac{1}{2^n}\Big(u^N(N \cap T^c) - u^N(N \cap S^c \cap T^c) + u^N(S)\Big) \\
&\overset{???}{=} \\
&\frac{1}{2^n}\Big(u^N(N) - u^N(N \cap T^c) + u^N(T)\Big) \\
&- \frac{1}{2^n}\Big(u^N(N \cap S^c) - u^N(N \cap S^c \cap T^c) + u^N(T)\Big).
\end{aligned}$$

the identity is proven.

Proof of uniqueness. The proof is similar to the one of Theorem 4.1.47 shown in [148, Proposition 291.3].

Proceed by induction on the number of players of the game $u \in \mathcal{G}^N$. Suppose $\exists\, \gamma \neq \overline{\gamma}$ game maps satisfying BCS_S and EFF_S. Prove the base case of the induction. Assume $n = 1$. The only pair of disjoint sets to check the property of BCP_S is $(S, T) = (\emptyset, N)$. In this case, Remark 4.3.17, shows that BCP_S holds true for all game maps. But there is a unique game map satisfying EFF_S for $n = 1$, that is $\gamma_\emptyset(u) = 0$; $\gamma_N(u) = u(N)$. Therefore, the base case is proven.

To prove the induction, assume the inductive hypothesis:

$$\forall u \in \mathcal{G}^M \text{ such that } |M| < n,\ \forall S \subseteq M, \quad \gamma_S(M,u) = \overline{\gamma}_S(M,u). \tag{4.95}$$

Assume to have $u \in \mathcal{G}^N$. Let two coalitions $S, T \subseteq N$ be such that $S \cap T = \emptyset$, being $s := |S|$, $t := |T|$. Hence:

$$\gamma_S(N,u) - 2^{-t}\,\gamma_S(N \setminus T, u^{N\setminus T}) = \gamma_T(N,u) - 2^{-s}\,\gamma_T(N \setminus S, u^{N\setminus S}) \tag{4.96}$$

$$\overline{\gamma}_S(N,u) - 2^{-t}\overline{\gamma}_S(N \setminus T, u^{N\setminus T}) = \overline{\gamma}_T(N,u) - 2^{-s}\,\overline{\gamma}_T(N \setminus S, u^{N\setminus S}). \tag{4.97}$$

Subtracting Equation (4.96) and (4.97), and using the inductive hypothesis (4.95):

$$\gamma_S(N,u) - \overline{\gamma}_S(N,u) = \gamma_T(N,u) - \overline{\gamma}_T(N,u). \tag{4.98}$$

The above equation holds for all disjoint coalitions S, T. Now, fix $T = \emptyset$. Then, $\gamma_S(N,u) - \overline{\gamma}_S(N,u) = 0$. Therefore, $\forall S \subseteq N$, $\gamma_S(N,u) = \overline{\gamma}_S(N,u)$. This is a contradiction with the hypothesis of $\gamma \neq \overline{\gamma}$. □

Remark 4.3.19 (Differences between BCP_i and BCP_S). The definitions of balanced contribution property for coalitions (BCP_S, Definition 4.3.16) and players (BCP_i. Definition 4.1.45) allow making a comparison between the $\mathcal{X}$-Shapley game map and the Shapley value ϕ. (i) The difference is clearly expressed by the coefficients 2^{-s} and 2^{-t}, which multiplies the game map computed on the respective coalitions and alternative subgames. For game maps satisfying BCP_S, it is possible to compare the allocation on a subgame and the one on the full game using such a scaling. In particular, the scaling is exponential in the number of players of the game. Clearly, the scaling distinguishes BCP_S from BCP_i, where there is no such scaling. Indeed, the value obtained by each player is of the same scaling magnitude of its subgames. (ii) Both the uniqueness theorems involve efficiency for players or coalitions (EFF_i or EFF_S). In the case of game maps, efficiency is used only in the base case of uniqueness induction proof. (iii) In the case of values, the subgames involved in the definition of objections and counter-objections drop only one player from the game; instead, in the case of game maps, eliminating one coalition causes all its subcoalitions being eliminated.

4.3.3 Further Properties of $\mathcal{X}$-Shapley

Notation 4.3.20. Given a game u, call $c_u := \mathcal{X}(u) \in \mathcal{G}^N$ the image game of u via the game map $\mathcal{X}$.

Proposition 4.3.21 (ϕ_i of the image game via $\mathcal{X}$-Shapley). *Given a game u and a player $i \in N$, the Shapley value ϕ_i computed on c_u is:*

$$\phi_i(c_u) = \frac{1}{2^{n-1}}\phi_i(u) \tag{4.99}$$

Proof.

$$\phi_i(c_u) = \sum_{S \subseteq N \setminus \{i\}} \frac{s!(n-s-1)!}{n!} \Big(\mathcal{X}_{S \cup i}(c_u) - \mathcal{X}_S(c_u) \Big). \tag{4.100}$$

Consider the expression $\Big(\mathcal{X}_{S \cup i}(c_u) - \mathcal{X}_S(c_u) \Big)$:

$$\begin{aligned} \mathcal{X}_{S \cup i} - \mathcal{X}_S &= \frac{1}{2^n} \Big[\Big(u(N) - u(N \setminus (S \cup i)) + u(S \cup i) \Big) - \Big(u(N) - u(N \setminus S) + u(S) \Big) \Big] \\ &= \frac{1}{2^n} \Big[u(N \setminus S) - u(N \setminus (S \cup i)) + u(S \cup i) - u(S) \Big]. \end{aligned} \tag{4.101}$$

Therefore, Equation (4.100) becomes:

$$\begin{aligned} \phi_i(c_u) = \frac{1}{2^n} \Big[& \sum_{S \subseteq N \setminus \{i\}} \frac{s!(n-s-1)!}{n!} (u(S \cup i) - u(S)) + \\ & \sum_{S \subseteq N \setminus \{i\}} \frac{s!(n-s-1)!}{n!} \Big(u(N \setminus S) - u(N \setminus (S \cup i)) \Big) \Big]. \end{aligned}$$

Observe that the first summand is $\phi_i(u)$. Furthermore, calling $T = N \setminus (S \cup i)$:

- $t = n - s - 1$;
- $n - t - 1 = n - (n - s - 1) - 1 = s$;
- $N \setminus S = T \cup \{i\}$.

Hence, the second summand of the above expression can be restated as:

$$\sum_{T \subseteq N \setminus \{i\}} \frac{t!(n-t-1)!}{n!} \Big(u(T \cup \{i\}) - u(T) \Big) = \phi_i(u).$$

Finally, Equation (4.100) becomes:

$$\phi_i(c_u) = \frac{1}{2^n} \, 2\phi_i(u) = \frac{1}{2^{n-1}} \phi_i(u).$$

□

Remark 4.3.22. For a game u, note that for each player $i \in N$, the Shapley value $\phi_i(u)$ can be expressed in terms of $\mathcal{X}$-Shapley similarly to the expression in terms of the marginal contribution (Definition 4.101). Indeed, observe that for each $S \subseteq N$:

$$\begin{aligned} \mathcal{X}_{N \setminus S}(u) - \mathcal{X}_S(u) &= \frac{1}{2^n} \Big(u(N) - u(S) + u(N \setminus S) - u(N) + u(N \setminus S) - u(S) \Big) \\ &= \frac{1}{2^{n-1}} (u(N \setminus S) - u(S)) \end{aligned}$$

Using Proposition 4.1.38, it follows that:

$$\phi_i(u) = 2^{n-1} \sum_{S \subseteq N \setminus \{i\}} \frac{s!(n-s-1)!}{n!} \Big(\mathcal{X}_{N \setminus S}(u) - \mathcal{X}_S(u) \Big) \tag{4.102}$$

This observation can be repeated for each value which depends on the marginal contribution, e.g. Banzhaf value [20].

Remark 4.3.23. Combining Equation (4.102) and again Proposition 4.1.38, it is easy to provide an alternative proof of Proposition 4.3.21.

Proposition 4.3.24 ($\mathcal{X}$-Shapley of $\mathcal{X}$-Shapley). *Given a game u, consider the image game via $\mathcal{X}$, c_u. Then,*

$$\forall S \subseteq N, \quad \mathcal{X}_S(c_u) = \frac{1}{2^{n-1}} \mathcal{X}_S(u) \tag{4.103}$$

Proof. Recall that $\mathcal{X}_N(u) = \frac{1}{2^{n-1}} u(N)$ by Proposition 4.3.10.

$$\begin{aligned}
\mathcal{X}_S(c_u) &= \frac{1}{2^n} \Big(\mathcal{X}_N(u) - \mathcal{X}_{N \setminus S}(u) + \mathcal{X}_S(u) \Big) \\
&= \frac{1}{2^{2n}} \Big[2\, u(N) - \Big(u(N) - u(S) + u(N \setminus S) \Big) + \Big(u(N) - u(N \setminus S) + u(S) \Big) \Big] \\
&= \frac{1}{2^{2n}} \Big[2u(N) - 2\, u(N \setminus S) + 2\, u(S) \Big] \\
&= \frac{1}{2^{n-1}} \mathcal{X}_S(u)
\end{aligned}$$

□

Proposition 4.3.25 (Superadditivity of the image game). *$c_u = \mathcal{X}(u)$ is superadditive if and only if, for each partition $\{R, S, T\}$ of the set of players N,*

$$-u(R \cup S \cup T) + u(R \cup S) + u(R \cup T) + u(S \cup T) - u(R) - u(S) - u(T) \geq 0 \tag{4.104}$$

Proof. By Definition 4.1.17, the condition for c_u superadditive is:

$$\forall S, T \subseteq N,\ S \cap T = \emptyset, \quad c_u(S \cup T) \geq c_u(S) + c_u(T). \tag{4.105}$$

This holds if and only if:

$$\mathcal{X}_{S \cup T}(u) \geq \mathcal{X}_S(u) + \mathcal{X}_T(u) \tag{4.106}$$

$$-u(N \setminus (S \cup T)) + u(S \cup T) \geq -u(N \setminus S) + u(S) + u(N) - u(N \setminus T) + u(T) \tag{4.107}$$

$$-u(N \setminus (S \cup T)) + u(S \cup T) - u(N) + u(N \setminus S) - u(S) + u(N \setminus T) - u(T) \geq 0 \tag{4.108}$$

Denoting by $R = N \setminus (S \cup T)$, then $\{R, S, T\}$ forms a partition of N. Therefore, (4.108) becomes:

$$-u(R \cup S \cup T) + u(R \cup S) + u(R \cup T) + u(S \cup T) - u(R) - u(S) - u(T) \geq 0$$

So the claim is proven. □

Corollary 4.3.26 (Constant sum and superadditive game via $\mathcal{X}$-Shapley). *Given a constant sum and superadditive game u, then its image game via $\mathcal{X}$, c_u, is superadditive.*

Proof. Consider as in Proposition 4.3.25 a partition R, S, T of N. Because u is constant sum and R is the complementary set of $S \cup T$, it is possible to write:

$$u(R \cup S \cup T) = u(R) + u(S \cup T).$$

This can be used in Equation (4.104):

$$\begin{aligned} -u(R) + u(R \cup T) + u(R \cup S) - u(R) - u(S) - u(T) \geq 0 \\ +u(R \cup T) - u(R) - u(T) + u(R \cup S) - u(R) - u(S) \geq 0. \end{aligned}$$

The last inequality is true by superadditivity of u applied to the pairs R, T and R, S (Definition 4.1.17). □

Remark 4.3.27. $c_{c_u} = \mathcal{X}(\mathcal{X}(u))$ is superadditive if u is superadditive (and so on, applying $\mathcal{X}$ iteratively). This result could also have been proven also by using Proposition 4.3.24.

Proposition 4.3.28 ($\mathcal{X}$-Shapley of a cohesive game). *If u is a cohesive game (Definition 4.1.17), then*

$$\mathcal{X}_S(u) \geq \frac{1}{2^{n-1}} u(S).$$

In particular, the above inequality becomes an equality if and only if u is also constant sum.

Proof. If u is a cohesive game, then, because $\{N \setminus S, S\}$ is a partition of N:

$$\forall S \subseteq N, \quad u(N) \geq u(N \setminus S) + u(S)$$

Hence,

$$\begin{aligned} \mathcal{X}_S(u) &= \frac{1}{2^n}\Big(u(N) - u(N \setminus S) + u(S)\Big) \\ &\geq \frac{1}{2^n}\Big(u(N \setminus S) + u(S) - u(N \setminus S) + u(S)\Big) \qquad (4.109) \\ &= \frac{1}{2^{n-1}} u(S). \qquad (4.110) \end{aligned}$$

Assuming further that

$$\forall S \subseteq N, \mathcal{X}_S(u) = \frac{1}{2^{n-1}} u(S),$$

then for each coalition $u(N) - u(N \setminus S) + u(S) = 2\,u(S)$, so u is constant sum. Vice versa, if u is constant sum, then the inequality of (4.109) becomes an equality. □

Remark 4.3.29. Let us consider a game $u \in \mathcal{G}^N$ with $n = 2$. Then the usual Shapley value is given by

$$\phi_1(u) = \frac{1}{2}\Big(u(\{1,2\}) - u(2) + u(1)\Big)$$
$$\phi_2(u) = \frac{1}{2}\Big(u(\{1,2\}) - u(1) + u(2)\Big).$$

The above observation extends from two player games to quotient games by the following proposition.

Proposition 4.3.30 ($\mathcal{X}$-Shapley and the quotient game of 2 players). *Given a game u, fix a coalition S, consider partition of N, $\mathfrak{P} = \{S, N \setminus S\}$ and the quotient game $u^{\mathfrak{P}}$ as in Definition 4.1.22. Then,*

$$\mathcal{X}_S = \frac{1}{2^{n-1}} \phi_1(u^{\mathfrak{P}}) \quad \forall S \subseteq N \tag{4.111}$$

$$\mathcal{X}_{N\setminus S} = \frac{1}{2^{n-1}} \phi_2(u^{\mathfrak{P}}) \quad \forall S \subseteq N \tag{4.112}$$

Proof.

$$\begin{aligned} \phi_1(u^{\mathfrak{P}}) &= \frac{1}{2}\Big(u(N) - u(N \setminus S) + u(S)\Big) = \\ &= 2^{n-1}\mathcal{X}_S(u) \end{aligned} \tag{4.113}$$

$$\begin{aligned} \phi_2(u^{\mathfrak{P}}) &= \frac{1}{2}\Big(u(N) - u(S) + u(N \setminus S)\Big) = \\ &= 2^{n-1}\mathcal{X}_{N\setminus S}(u) \end{aligned} \tag{4.114}$$

□

Remark 4.3.31. From Proposition 4.3.30, for a fixed game u, the axiom CS_S for $\mathcal{X}$-Shapley explains as:

$$\begin{aligned} \mathcal{X}_S(u) + \mathcal{X}_{N\setminus S}(u) &= \frac{1}{2^{n-1}}\Big(\phi_1(u^{\mathfrak{P}}) + \phi_2(u^{\mathfrak{P}})\Big) \\ &\overset{\phi\ \mathrm{EFF}_i}{=} \frac{1}{2^{n-1}}(u^{\mathfrak{P}})(N) = \frac{1}{2^{n-1}}u(N) = \mathcal{X}_N(u) \end{aligned}$$

that is the axiom EFF_i translated to the quotient game $u^{\mathfrak{P}}$.

Remark 4.3.32. The results shown in this section suggest a new, interesting game map. In particular, it is easy to observe that in Proposition 4.3.21, 4.3.28, and 4.3.30 4.3.24 it appears the factor $\frac{1}{2^{n-1}}$, . Thus, a new game map can be defined as follows.

Definition 4.3.33 ($\mathcal{X}'$-Shapley). *For each game u and coalition S, define*

$$\mathcal{X}'_S(u) = 2^{n-1}\mathcal{X}_S(u) = \frac{1}{2}\Big(u(N) - u(N \setminus S) + u(S)\Big). \tag{4.115}$$

Remark 4.3.34. By linearity, all the equations of the propositions listed above cancel the scale factor $\frac{1}{2^{n-1}}$. Therefore, $\mathcal{X}'$-Shapley is a game map such that:

(i) it preserves the Shapley value, that is, defining c'_u as the image game via $\mathcal{X}'$, $\phi_i(c'_u) = \phi_i(u)$;

(ii) $\mathcal{X}'(c'_u) = c'_u$;

(iii) if u is a cohesive game, $c'_u(S) \geq u(S)$

(iv) it is the Shapley value of the quotient game of two players S and $N \setminus S$.

Clearly, $\mathcal{X}'$ game map satisfies all the axioms, except EFF_S. But the EFF_S can be restated as an average efficiency axiom that $\mathcal{X}'$ satisfies:

Definition 4.3.35 (AvEFF_S). *Given a game map γ, it is said that γ satisfies the Average Efficiency property for coalitions (AvEFF_S) if,*

$$\forall u \in \mathcal{G}^N,\ \frac{1}{2^{n-1}} \sum_{S \subseteq N} \gamma_S(u) = u(N) \tag{4.116}$$

Remark 4.3.36. The $\mathcal{X}'$ game map has a second interesting property, derived from $\mathcal{X}$-Shapley. Restating the balanced contribution property BCP_S of $\mathcal{X}$-Shapley without the scaling factors, it is possible to obtain an almost identical result to Theorem 4.3.18. Formally, first define BCP'$_S$ and then state the new Theorem of stability for $\mathcal{X}'$.

Definition 4.3.37 (Balanced contribution for coalitions without scaling). *A game map γ satisfies the balanced contribution property for coalitions, without scaling, (BCP'$_S$) if, $\forall S, T \subseteq N$ such that $S \cap T = \emptyset$,*

$$\gamma_S(N, u) - \gamma_S(N \setminus T, u^{N \setminus T}) = \gamma_T(N, u) - \gamma_T(N \setminus S, u^{N \setminus S}) \tag{4.117}$$

Theorem 4.3.38 ($\mathcal{X}'$-Shapley and balanced contributions). *$\mathcal{X}'$-Shapley is the unique game map satisfying AvEFF_S and BCP'$_S$.*

Proof. The proof requires showing the same identity of Theorem 4.3.18, without the scaling factors. □

4.4 Conclusion and Perspectives for Future Research

In this chapter, the properties of the $\mathcal{X}$-Shapley game map have been compared with those of the Shapley value ϕ. The parallelism between the two concepts is evident from the illustrated characterizations: the axiomatic, the counter-objections, and the Hodge theoretic one. In particular, focusing on the Hodge theoretic characterization, $\mathcal{X}$-Shapley has been characterized in a similar fashion to ϕ. This suggests a strong link between the defined game graphs and the game-theoretic properties, in both the graph representations for players or coalitions. However, this chapter is only a starting point for the study of the connection between cooperative game theory and graph theory, and it leaves many open questions for further investigations. This section tries to describe and discuss at least some open questions for further research.

4.4.1 On the Game-Theoretic Interpretation of $\mathcal{X}$-Shapley

The derivation of $\mathcal{X}$-Shapley via the Hodge theoretic characterization is quite straightforward, given the one of Stern et al. [180]. Nonetheless, from the game-theoretic viewpoint, the interpretation of concepts and definitions presented in Section 4.2 is more complex.

As expressed in Remark 4.3.2, the game map definition is a simple translation of the concept of value for players, including the requirement of allocating 0 to the empty set. Nonetheless, the concept of allocation for coalitions is not easy to be understood in application domains. The Shapley value derives from the idea that the grand coalition has formed, and the total payoff should be split among the players. Instead, the process of allocating part of the grand coalition payoff to single coalitions has no direct meaning in game theory. Giving worth to coalitions formed by multiple players contrasts with the assumption of starting already with the computed payoff for each coalition.

By changing the point of view over the matter, it could emerge a more practical interpretation path. In fact, a value represents an allocation among players, and the Shapley value represents one fair version. The game map is a transformation from a game to a game on the same set of players. Thus, game theory could look at $\mathcal{X}$-Shapley as a transformation from a game to a fair version of a game, where the fairness derives from the axioms listed in Definition 4.3.7. More precisely, $\mathcal{X}'$-Shapley of Definition 4.3.33 suits better for this aim because it does not involve the rescaling factor, which depends on the number of players. In this sense, $\mathcal{X}'$-Shapley of an image game via $\mathcal{X}'$-Shapley remains the same; the suggested interpretation is that a fair cooperative game should satisfy the axioms and should not change to become fair, as the image game through $\mathcal{X}'$-Shapley does.

The average efficiency property of $\mathcal{X}'$-Shapley, as stated in Definition 4.3.35,

provides a different scale to observe the worth computation procedure in a fair game. The null coalition axiom has the same interpretation as for values: a fair game should give worth 0 to null coalitions. The constant sum and bilaterality properties show a deep connection between complementary coalitions in a fair game. In particular, whatever worth is gained by one coalition, the complementary should gain the remainder, and their allocation should be equal if they gain the same. The symmetry property of a value is different from the one of coalitions, because the comparison affects every pair of players in the first case, while in the second involves only pairs of coalitions with the same size. The role of symmetric coalitions and its relationship with the other properties is yet to be understood; thus, it is left to future works.

Moreover, recall that a cohesive game is motivated by the rationality of players for the grand coalition formation (Remark 4.1.18). Then, looking again at Remark 4.3.32, $c'_S(u) \geq u(S)$, so $c'_u(S)$ allocates more than what already gained by $u(S)$ for each coalition, and the equality holds if u is constant sum. Therefore, $c'_u(S)$ is to be preferred over $u(S)$, if S is rational. The property of the cohesiveness of the image game suggests links between the interpreted fair game $c'_S(u)$ and the rationality of players.

The balanced contribution property provides another clue for parallelism with the Shapley value, and specifically the version without scaling. Indeed, it shows a stability result for $\mathcal{X}$ (or $\mathcal{X}'$) of the negotiation between coalitions: the agreement between coalitions is stable for the allocation given by $\mathcal{X}$ (or $\mathcal{X}'$) as far as every coalition objection is balanced by a counter-objection of a disjoint coalition. Further investigation could broaden the link between the graph-theoretic differential and the balanced contribution property, similar to what was done by [98] for the Shapley value.

Clearly, the proposed interpretations are far from complete or formal. Thus, new research efforts and ideas could help to broaden the usage of $\mathcal{X}$-Shapley.

4.4.2 Open Questions of Cooperative Game Theory

Theorem 4.1.43 shows a uniqueness results from a set of axioms. For the original Shapley value, the uniqueness result is accompanied by examples for axioms independence (see [148, Exercise 294.2]). In other words, whatever axiom is dropped from the hypothesis, there exists a second value different from the Shapley value satisfying the remaining axioms. Thus, the uniqueness result becomes false when not assuming all the axioms, and the axioms are not redundant. The same is not straightforward for game maps. Two easy examples are found when dropping EFF_S or NLL_S. For EFF_S, it is easy to show that the game map $\gamma_S(u) = 2\,\mathcal{X}_S(u)$ satisfies the other axioms, but not EFF_S (or every other rescaling of $\mathcal{X}$, as $\mathcal{X}'$). For NLL_S,

the following game map is an example:

$$\gamma_S(u) = \begin{cases} 0 & S = \emptyset \\ \frac{1}{2^{n-1}} u(N) & S = N \\ \frac{1}{2^n} u(N) & \text{otherwise} \end{cases} .$$

For the other axioms, an example needs yet to be found.

The role of constant sum and bilaterality properties in the framework of graph theory is a second open question. Recall that the properties shown in Theorem 4.2.15 are true for each entry of the S-differential game vectors u_S. Properties similar to constant sum and bilaterality seem not to hold for the games u_S.

A third open question is a complete comparison with respect to other recent concepts giving a version of value to coalitions, for example [100]. Clearly, the aim to provide a value of a coalition to another coalition leads to a different formulation of the solution concept. In addition, no Hodge theoretic characterization is present in [100]. A second comparison to face is the one with the interaction indices of [90]. Indeed, both game maps and interaction indices belong to the class of set functions [89]. The interaction of a set of players and the $\mathcal{X}$-Shapley of the same are different objects that deserve further analysis. An investigation about a possible relationship is left for future work.

Another question concerns the study of $\mathcal{X}$-Shapley if not all the subset coalitions are admissible. Indeed, the work of [138] include a graph describing the possibilities or impossibilities of negotiation. The graph nodes represent players, and connected components describe the feasibility of the negotiation among players in each component. The work of [129] is a recent paper moving forward in the same direction: it applies a version of the Shapley value to a negotiation described by Simplicial complexes.

A fifth open question regards the possible relationships between values computed on the image game via $\mathcal{X}$-Shapley, c_u, or computed on the original game u. Indeed, by Proposition 4.1.38, it is possible to link the value computed on c_u and on u, if the solution concept depends linearly on the marginal contribution of players. By Proposition 4.3.21 and Remark 4.3.32 it holds that the Shapley value on the image game via $\mathcal{X}'$ remains equal to the original Shapley for each player in the game, that is $\forall i \in N$, $\phi_i(c'_u) = \phi_i(u)$. Instead, the understanding of the relationship is not evident if the computed value does not depend on the marginal contribution of players, as the values suggested in [132]. For instance, consider the Deegan-Packel value [52], which depends only on the coalition payoff $c_u(S)$ of the coalitions S containing a specific player:

$$\psi_i^{\mathrm{DP}}(c_u) = \sum_{S \subseteq N : i \in S} \frac{\mathcal{X}_S(u)}{s}$$

4.4.3 $\mathcal{X}$-Shapley and Explainable Artificial Intelligence

A recent appealing intertwining between cooperative game theory and Machine Learning is the Shapley value application to Explainable Artificial Intelligence (XAI) [126]. In XAI, the Shapley value is commonly employed to compute feature importance for an analyzed model, thus providing clues for explanations. The presented $\mathcal{X}$-Shapley game map could potentially represent another game theoretic result significant for XAI. Indeed, at least three different research directions derive from $\mathcal{X}$-Shapley in XAI.

The first direction is the computation of an importance directly for sets of features, instead of single features. In fact, there are already studies trying to extend the Shapley value to groups of features [110]. Being an allocation for coalitions, $\mathcal{X}$-Shapley can be suitable for this aim. The second direction consists of the computation of an interaction index between features, inspired by the interaction indices presented by [90] in cooperative game theory. Being an allocation for coalitions, $\mathcal{X}$-Shapley can be suitable for this aim. As before, some researchers have recently explored possibilities for an application in the XAI domain, for example, [183]. A third direction comes from the Hodge theoretic characterization of $\mathcal{X}$-Shapley. In particular, for the Shapley value, the Hodge theoretic characterization has been used by [121] to inspect explanations. The claim is that, by computing all the entries of the S-differential games, it could be possible to derive new usages to apply $\mathcal{X}$-Shapley on XAI. Summarizing, other research areas could benefit from deepening the knowledge of $\mathcal{X}$-Shapley. Specifically, the critical areas where XAI is needed to employ black-box Machine Learning models.

Finally, the following example shows how to compute a coalitional version of the SHAP attributions based on $\mathcal{X}$-Shapley, similar to the reported Example 4.1.53 from [2].

Example 4.4.1 ($\mathcal{X}$-SHAP for a linear model and independent features). Let $\widehat{\boldsymbol{F}}$, $\boldsymbol{x}^*$ and $u^{\widehat{\boldsymbol{F}}}$ as in Section 4.1.6. Furthermore, assume that $\widehat{\boldsymbol{F}}$ is a linear model and the features of X are independent, as in Example 4.1.53. For a fixed subset of features $S \in N$, it holds that:

$$\mathcal{X}_S(u^{\widehat{\boldsymbol{F}}}) = \frac{1}{2^n}\Big(\sum_{j\in S}\varphi_j(\widehat{\boldsymbol{F}}) + \sum_{j\in N\setminus S}\theta_j\mathbb{E}\left[x_j\right] + \sum_{j\in S}\theta_j x_j^*\Big) = \frac{1}{2^n}\Big(\sum_{j\in S}\varphi_j(\widehat{\boldsymbol{F}}) + u^{\widehat{\boldsymbol{F}}}(S)\Big).$$

Specifically, if $S = \{i\}$ is composed by a single player:

$$\mathcal{X}_S(u^{\widehat{\boldsymbol{F}}}) = \frac{1}{2^n}\Big(\varphi_i(\widehat{\boldsymbol{F}}) + \sum_{j\in N\setminus i}\theta_j\mathbb{E}\left[x_j\right] + \theta_i x_i^*\Big) = \frac{1}{2^n}\Big(\varphi_i(\widehat{\boldsymbol{F}}) + u^{\widehat{\boldsymbol{F}}}(i)\Big).$$

Therefore, $\mathcal{X}$-SHAP for S becomes:

$$\begin{aligned}\mathcal{X}_S(u^{\widehat{\boldsymbol{F}}}) &= \frac{1}{2^n}\Big(u^{\widehat{\boldsymbol{F}}}(N) - u^{\widehat{\boldsymbol{F}}}(N \setminus S) + u^{\widehat{\boldsymbol{F}}}(S)\Big) \\ &= \frac{1}{2^n}\Big(\sum_{j\in N}\theta_j x_j^* - \Big(\sum_{j\in S}\theta_j \mathbb{E}\left[x_j\right] + \sum_{j\in N\setminus S}\theta_j x_j^*\Big) + \Big(\sum_{j\in N\setminus S}\theta_j \mathbb{E}\left[x_j\right] + \sum_{j\in S}\theta_j x_j^*\Big)\Big) \\ &= \frac{1}{2^n}\Big(\sum_{j\in S}\theta_j\Big(x_j^* - \mathbb{E}\left[x_j\right]\Big) + \sum_{j\in N\setminus S}\theta_j \mathbb{E}\left[x_j\right] + \sum_{j\in S}\theta_j x_j^*\Big)\end{aligned}$$

In particular, if the features are zero-mean, that is $\forall j \in N, \mathbb{E}\left[x_j\right] = 0$, then

$$\mathcal{X}_S(u^{\widehat{\boldsymbol{F}}}) = \frac{1}{2^{n-1}}\sum_{j\in S}\theta_j x_j^* = \frac{1}{2^{n-1}}\sum_{j\in S}\varphi_j^*(u^{\widehat{\boldsymbol{F}}})$$

This example is a simple case that allows analytically linking the relevance computed by φ_j and the relevance by $\mathcal{X}$. The challenge is to derive an algorithmic version of the above reasoning that is suitable for analytically intractable Machine Learning models, like the ones belonging to the class of Deep Learning.

List of Figures

List of Tables

List of Definitions

List of Theorems, Corollaries, Propositions and Lemmas

Bibliography

[1] Karthik V. Aadithya et al. "Efficient Computation of the Shapley Value for Centrality in Networks". In: *Internet and Network Economics*. Ed. by Amin Saberi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–13. ISBN: 978-3-642-17572-5.

[2] Kjersti Aas, Martin Jullum, and Anders Løland. "Explaining individual predictions when features are dependent: More accurate approximations to Shapley values". In: *Artificial Intelligence* 298 (2021), p. 103502. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2021.103502. URL: https://www.sciencedirect.com/science/article/pii/S0004370221000539.

[3] Russell L Ackoff. "From data to wisdom". In: *Journal of applied systems analysis* 16.1 (1989), pp. 3–9.

[4] Julius Adebayo et al. "Debugging Tests for Model Explanations". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 700–712.

[5] Julius Adebayo et al. "Sanity checks for saliency maps". In: *Advances in neural information processing systems* 31 (2018).

[6] P.M. Adler. *Fractures and Fracture Networks*. Kluwer Academic, Dordrecht, 1999.

[7] Maximilian Alber et al. "iNNvestigate neural networks!" In: *J. Mach. Learn. Res.* 20.93 (2019), pp. 1–8.

[8] Réka Albert and Albert-László Barabási. "Topology of Evolving Networks: Local Events and Universality". In: *Phys. Rev. Lett.* 85 (24 Dec. 2000), pp. 5234–5237. DOI: 10.1103/PhysRevLett.85.5234. URL: https://link.aps.org/doi/10.1103/PhysRevLett.85.5234.

[9] G. Aldrich et al. "Analysis and Visualization of Discrete Fracture Networks Using a Flow Topology Graph". In: *IEEE Transactions on Visualization and Computer Graphics* 23.8 (2017), pp. 1896–1909.

[10] Encarnacion Algaba, Vito Fragnelli, and Joaquin Sanchez-Soriano. *Handbook of the Shapley value*. CRC Press, 2019.

[11] David Alvarez Melis and Tommi Jaakkola. "Towards robust interpretability with self-explaining neural networks". In: *Advances in neural information processing systems* 31 (2018).

[12] Marco Ancona, Cengiz Oztireli, and Markus Gross. "Explaining deep neural networks with a polynomial time algorithm for shapley value approximation". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 272–281.

[13] Christopher Anders et al. "Fairwashing explanations with off-manifold detergent". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 314–323.

[14] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information fusion* 58 (2020), pp. 82–115.

[15] James Atwood and Don Towsley. "Diffusion-Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016, pp. 1993–2001. URL: http://papers.nips.cc/paper/6212-diffusion-convolutional-neural-networks.pdf.

[16] Robert J Aumann and Michael Maschler. "The Bargaining Set for Cooperative Games". In: *Advances in Game Theory* (1964), pp. 443–476.

[17] Robert J. J. Aumann. "Economic Applications of the Shapley Value". In: *Game-Theoretic Methods in General Equilibrium Analysis*. Ed. by Jean-Francois Mertens and Sylvain Sorin. Dordrecht: Springer Netherlands, 1994, pp. 121–133. ISBN: 978-94-017-1656-7. DOI: 10.1007/978-94-017-1656-7_12. URL: https://doi.org/10.1007/978-94-017-1656-7_12.

[18] Sebastian Bach et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015), e0130140.

[19] David Baehrens et al. "How to explain individual classification decisions". In: *The Journal of Machine Learning Research* 11 (2010), pp. 1803–1831.

[20] John F Banzhaf III. "Weighted voting doesn't work: A mathematical analysis". In: *Rutgers L. Rev.* 19 (1964), p. 317.

[21] Albert-László Barabási and Réka Albert. "Emergence of Scaling in Random Networks". In: *Science* 286.5439 (1999), pp. 509–512. DOI: 10.1126/science.286.5439.509. URL: https://www.science.org/doi/abs/10.1126/science.286.5439.509.

[22] Alex Bavelas. "Communication Patterns in Task-Oriented Groups". In: *Journal of the Acoustical Society of America* 22.6 (1950), pp. 725–730. DOI: 10.1121/1.1906679.

[23] M.F. Benedetto et al. "The virtual element method for discrete fracture network simulations". In: *Comput. Methods Appl. Mech. Engrg.* 280.0 (2014), pp. 135–156. ISSN: 0045-7825. DOI: 10.1016/j.cma.2014.07.016.

[24] S. Berrone, S. Scialò, and F. Vicini. "Parallel Meshing, Discretization, and Computation of Flow in Massive Discrete Fracture Networks". In: *SIAM Journal on Scientific Computing* 41.4 (2019), pp. C317–C338. DOI: 10.1137/18M1228736. eprint: https://doi.org/10.1137/18M1228736. URL: https://doi.org/10.1137/18M1228736.

[25] S. Berrone et al. "Advanced computation of steady-state fluid flow in Discrete Fracture-Matrix models: FEM–BEM and VEM–VEM fracture-block coupling". In: *GEM - International Journal on Geomathematics* 9.2 (July 2018), pp. 377–399. ISSN: 1869-2680. DOI: 10.1007/s13137-018-0105-3.

[26] Stefano Berrone and Francesco Della Santa. "Performance Analysis of Multi-Task Deep Learning Models for Flux Regression in Discrete Fracture Networks". In: *Geosciences* 11.3 (Mar. 2021), p. 131. DOI: 10.3390/geosciences11030131. URL: https://doi.org/10.3390%2Fgeosciences11030131.

[27] Stefano Berrone, Sandra Pieraccini, and Stefano Scialò. "A PDE-constrained optimization formulation for discrete fracture network flows". In: *SIAM J. Sci. Comput.* 35.2 (2013), B487–B510. ISSN: 1064-8275. DOI: 10.1137/120865884.

[28] Stefano Berrone, Sandra Pieraccini, and Stefano Scialò. "An optimization approach for large scale simulations of discrete fracture network flows". In: *J. Comput. Phys.* 256 (2014), pp. 838–853. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2013.09.028.

[29] Stefano Berrone, Sandra Pieraccini, and Stefano Scialò. "On simulations of discrete fracture network flows with an optimization-based extended finite element method". In: *SIAM J. Sci. Comput.* 35.2 (2013), A908–A935. ISSN: 1064-8275. DOI: 10.1137/120882883. URL: http://dx.doi.org/10.1137/120882883.

[30] Stefano Berrone, Sandra Pieraccini, and Stefano Scialò. "Towards effective flow simulations in realistic discrete fracture networks". In: *J. Comput. Phys.* 310 (2016), pp. 181–201. DOI: 10.1016/j.jcp.2016.01.009.

[31] Stefano Berrone et al. "A parallel solver for large scale DFN flow simulations". In: *SIAM J. Sci. Comput.* 37.3 (2015), pp. C285–C306. DOI: 10.1137/140984014.

[32] Stefano Berrone et al. "Discrete Fracture Network insights by eXplainable AI". In: *Machine Learning and the Physical Sciences*. Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS). Neural Information Processing Systems Foundation, 2020. URL: https://ml4physicalsciences.github.io/2020/.

[33] Stefano Berrone et al. “Graph-Informed Neural Networks for Regressions on Graph-Structured Data”. In: *Mathematics* 10.5 (2022), p. 786.

[34] Stefano Berrone et al. “Layer-wise relevance propagation for backbone identification in discrete fracture networks”. In: *Journal of Computational Science* 55 (2021), p. 101458. ISSN: 1877-7503. DOI: https://doi.org/10.1016/j.jocs.2021.101458. URL: https://www.sciencedirect.com/science/article/pii/S1877750321001356.

[35] Stefano Berrone et al. “Machine learning for flux regression in discrete fracture networks”. In: *GEM - International Journal on Geomathematics* 12.1 (Apr. 2021), p. 9. DOI: 10.1007/s13137-021-00176-0. URL: https://doi.org/10.1007/s13137-021-00176-0.

[36] Avrim L Blum and Pat Langley. “Selection of relevant features and examples in machine learning”. In: *Artificial intelligence* 97.1-2 (1997), pp. 245–271.

[37] Moritz Böhle et al. “Layer-wise relevance propagation for explaining deep neural network decisions in mri-based alzheimer’s disease classification”. In: *Frontiers in aging neuroscience* 11 (2019), p. 194.

[38] Ulrik Brandes and Thomas Erlebach, eds. *Network Analysis - Methodological Foundations*. Vol. 3418. Theoretical Computer Science and General Issues 1. Springer-Verlag Berlin Heidelberg, 2005. ISBN: 978-3-540-24979-5. DOI: 10.1007/b106453.

[39] Rene van den Brink. “Null or nullifying players: the difference between the Shapley value and equal division solutions”. In: *Journal of Economic Theory* 136.1 (2007), pp. 767–775.

[40] Michael M. Bronstein et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. 2021. DOI: 10.48550/ARXIV.2104.13478. URL: https://arxiv.org/abs/2104.13478.

[41] J. Bruna et al. “Spectral networks and locally connected networks on graphs”. In: *Proceedings of the International Conference on Learning Representations*. 2014.

[42] G. Cammarata et al. “The Hydro-Mechanically Coupled Response of Rock Fractures”. English. In: *Rock Mechanics and Rock Engineering* 40.1 (2007), pp. 41–61. ISSN: 0723-2632. DOI: http://dx.doi.org/10.1007/s00603-006-0081-z.

[43] Alberto Cano, Amelia Zafra, and Sebastián Ventura. “An interpretable classification rule mining algorithm”. In: *Information Sciences* 240 (2013), pp. 1–20.

[44] Giuseppe Carleo et al. “Machine learning and the physical sciences”. In: *Rev. Mod. Phys.* 91 (4 Dec. 2019), p. 045002. DOI: 10.1103/RevModPhys.91.045002. URL: https://link.aps.org/doi/10.1103/RevModPhys.91.045002.

[45] J. Cheriyan and S. N. Maheshwari. "Analysis of preflow push algorithms for maximum network flow". In: *Foundations of Software Technology and Theoretical Computer Science*. Ed. by Kesav V. Nori and Sanjeev Kumar. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 30–48. ISBN: 978-3-540-46030-5.

[46] Fan Chung and S.-T. Yau. "Discrete Green's Functions". In: *Journal of Combinatorial Theory, Series A* 91.1 (2000), pp. 191–214. ISSN: 0097-3165. DOI: https://doi.org/10.1006/jcta.2000.3094. URL: https://www.sciencedirect.com/science/article/pii/S0097316500930942.

[47] European Commission. *2018 reform of EU data protection rules*. May 25, 2018. URL: https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf (visited on 06/17/2019).

[48] Ian Covert, Scott M Lundberg, and Su-In Lee. "Explaining by Removing: A Unified Framework for Model Explanation." In: *J. Mach. Learn. Res.* 22 (2021), pp. 209–1.

[49] Ian Covert, Scott M Lundberg, and Su-In Lee. "Understanding global feature contributions with additive importance measures". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17212–17223.

[50] Eric van Damme and Dave Furth. "Game theory and the market". In: *Chapters in game theory*. Springer, 2002, pp. 51–81.

[51] Morton Davis and Michael Maschler. "The kernel of a cooperative game". In: *Naval Research Logistics Quarterly* 12.3 (1965), pp. 223–259.

[52] John Deegan and Edward W Packel. "A new index of power for simplen-person games". In: *International Journal of Game Theory* 7.2 (1978), pp. 113–123.

[53] M. Defferrard, X. Bresson, and P. Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016, pp. 3844–3852.

[54] Daniele Giorgio Degiorgi and Klaus Simon. "A dynamic algorithm for line graph recognition". In: *Graph-Theoretic Concepts in Computer Science*. Ed. by Manfred Nagl. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 37–48. ISBN: 978-3-540-48487-5. DOI: 10.1007/3-540-60618-1_64. URL: https://doi.org/10.1007/3-540-60618-1_64.

[55] W.S. Dershowitz and C. Fidelibus. "Derivation of equivalent pipe networks analogues for three-dimensional discrete fracture networks by the boundary element method". In: *Water Resource Res.* 35 (1999), pp. 2685–2691. DOI: http://dx.doi.org/10.1029/1999WR900118.

[56] Alberto Diez-Olivan et al. “Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0”. In: *Information Fusion* 50 (2019), pp. 92–111.

[57] Nedialko B. Dimitrov and David P. Morton. “Interdiction Models and Applications”. In: *Handbook of Operations Research for Homeland Security*. Ed. by Jeffrey W. Herrmann. New York (NY): Springer New York, 2013, pp. 73–103. DOI: 10.1007/978-1-4614-5278-2_4.

[58] Sibo Ding. “The α-maximum flow model with uncertain capacities”. In: *Applied Mathematical Modelling* 39.7 (2015), pp. 2056–2063. ISSN: 0307-904X. DOI: https://doi.org/10.1016/j.apm.2014.10.021. URL: https://www.sciencedirect.com/science/article/pii/S0307904X14004946.

[59] Balthazar Donon et al. “Graph Neural Solver for Power Systems”. In: *Proceedings of the International Joint Conference on Neural Networks* 2019-July.July (2019), pp. 1–8. DOI: 10.1109/IJCNN.2019.8851855.

[60] Balthazar Donon et al. “Neural networks for power flow: Graph neural solver”. In: *Electric Power Systems Research* 189.October 2019 (2020), p. 106547. ISSN: 03787796. DOI: 10.1016/j.epsr.2020.106547. URL: https://doi.org/10.1016/j.epsr.2020.106547.

[61] Finale Doshi-Velez and Been Kim. “Towards a rigorous science of interpretable machine learning”. In: *arXiv preprint arXiv:1702.08608* (2017).

[62] Filip Karlo Dosilovic, Mario Brcic, and Nikica Hlupic. “Explainable artificial intelligence: A survey”. In: *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE. 2018, pp. 0210–0215.

[63] J.R. de Dreuzy et al. “Synthetic benchmark for modeling flow in 3D fractured media”. In: *Computers & Geosciences* 50.0 (2013), pp. 59–71.

[64] Jack Dunn, Luca Mingardi, and Ying Daisy Zhuo. “Comparing interpretability and explainability for feature selection”. In: *arXiv preprint arXiv:2105.05328* (2021).

[65] Herbert Edelsbrunner and John L Harer. *Computational topology: an introduction*. American Mathematical Society, 2022.

[66] Lilian Edwards and Michael Veale. “Slave to the algorithm: Why a right to an explanation is probably not the remedy you are looking for”. In: *Duke L. & Tech. Rev.* 16 (2017), p. 18.

[67] P. Erdos and A. Rényi. “On Random Graphs”. In: *Publicationes Mathematicae* 6 (1959), pp. 290–297.

[68] Dumitru Erhan et al. “Visualizing higher-layer features of a deep network”. In: *University of Montreal* 1341.3 (2009), p. 1.

[69] Thomas S Ferguson. *A course in game theory*. World Scientific, 2020.

[70] C. Fidelibus, G. Cammarata, and M. Cravero. *Hydraulic characterization of fractured rocks. In: Abbie M, Bedford JS (eds) Rock mechanics: new research.* Nova Science Publishers Inc., New York, 2009.

[71] L. Formaggia et al. "A reduced model for Darcy's problem in networks of fractures". In: *ESAIM: Mathematical Modelling and Numerical Analysis* 48 (04 July 2014), pp. 1089–1116. ISSN: 1290-3841. DOI: http://dx.doi.org/0.1051/m2an/2013132.

[72] L. Formaggia et al. "Optimal techniques to simulate flow in fractured reservoir". In: *ECMOR XIV-14th European conference on the mathematics of oil recovery*. 2014.

[73] Linton C. Freeman. "A Set of Measures of Centrality Based on Betweenness". In: *Sociometry* 40.1 (1977), pp. 35–41. ISSN: 00380431. URL: http://www.jstor.org/stable/3033543.

[74] Alex A Freitas. "Comprehensible classification models: a position paper". In: *ACM SIGKDD explorations newsletter* 15.1 (2014), pp. 1–10.

[75] Daniel Fryer, Inga Strümke, and Hien Nguyen. "Shapley values for feature selection: the good, the bad, and the axioms". In: *IEEE Access* 9 (2021), pp. 144352–144360.

[76] A. Fumagalli and A. Scotti. "A numerical method for two-phase flow in fractured porous media with non-matching grids". In: *Advances in Water Resources* 62 (2013), pp. 454–464. ISSN: 0309-1708. DOI: http://dx.doi.org/10.1016/j.advwatres.2013.04.001.

[77] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. "Large-Scale Learnable Graph Convolutional Networks". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (July 2018). DOI: 10.1145/3219819.3219947. URL: http://dx.doi.org/10.1145/3219819.3219947.

[78] Amirata Ghorbani, Abubakar Abid, and James Zou. "Interpretation of neural networks is fragile". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 3681–3688.

[79] E. N. Gilbert. "Random Graphs". In: *The Annals of Mathematical Statistics* 30.4 (1959), pp. 1141–1144. DOI: 10.1214/aoms/1177706098. URL: https://doi.org/10.1214/aoms/1177706098.

[80] Justin Gilmer et al. "Neural Message Passing for Quantum Chemistry". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1263–1272. URL: https://proceedings.mlr.press/v70/gilmer17a.html.

[81] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Journal of Machine Learning Research* 9 (2010), pp. 249–256. ISSN: 15324435.

[82] Andrew V. Goldberg and Satish Rao. "Beyond the Flow Decomposition Barrier". In: *J. ACM* 45.5 (Sept. 1998), pp. 783–797. ISSN: 0004-5411. DOI: 10.1145/290179.290181. URL: https://doi.org/10.1145/290179.290181.

[83] Andrew V. Goldberg and Robert E. Tarjan. "A New Approach to the Maximum-Flow Problem". In: *J. ACM* 35.4 (Oct. 1988), pp. 921–940. ISSN: 0004-5411. DOI: 10.1145/48014.61051. URL: https://doi.org/10.1145/48014.61051.

[84] Martin Charles Golumbic, ed. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, 1980. ISBN: 978-0-12-289260-8. URL: https://www.sciencedirect.com/book/9780122892608/algorithmic-graph-theory-and-perfect-graphs.

[85] Julio Gonzalez-Diaz, Ignacio Garcia-Jurado, and M Gloria Fiestras-Janeiro. "An introductory course on mathematical game theory". In: *Graduate studies in mathematics* 115 (2010).

[86] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. www.deeplearningbook.org. MIT Press, 2016.

[87] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[88] M. Gori, G. Monfardini, and F. Scarselli. "A new model for learning in graph domains". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2. 2005, 729–734 vol. 2. DOI: 10.1109/IJCNN.2005.1555942.

[89] Michel Grabisch et al. *Set functions, games and capacities in decision making*. Vol. 46. Springer, 2016.

[90] Michel Grabisch and Marc Roubens. "An axiomatic approach to the concept of interaction among players in cooperative games". In: *International Journal of game theory* 28.4 (1999), pp. 547–565.

[91] Riccardo Guidotti et al. "A survey of methods for explaining black box models". In: *ACM computing surveys (CSUR)* 51.5 (2018), pp. 1–42.

[92] David Gunning et al. *DARPA's explainable AI (XAI) program: A retrospective*. 2021.

[93] Isabelle Guyon and André Elisseeff. "An introduction to variable and feature selection". In: *Journal of machine learning research* 3.Mar (2003), pp. 1157–1182.

[94] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[95] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

[96] Peters Hans. *Game Theory: A Multi-Leveled Approach*. 2015.

[97] John C Harsanyi. "A simplified bargaining model for the n-person cooperative game". In: *International Economic Review* 4.2 (1963), pp. 194–220.

[98] Sergiu Hart and Andreu Mas-Colell. "Potential, value, and consistency". In: *Econometrica: Journal of the Econometric Society* (1989), pp. 589–614.

[99] Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer, 2009.

[100] Kjell Hausken. "The Shapley value of coalitions to other coalitions". In: *Humanities and Social Sciences Communications* 7.1 (2020), pp. 1–10.

[101] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-Decem (2016), pp. 770–778. ISSN: 10636919. DOI: 10.1109/CVPR.2016.90. arXiv: 1512.03385.

[102] Kalle Hjerppe, Jukka Ruohonen, and Ville Leppänen. "The general data protection regulation: requirements, architectures, and constraints". In: *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE. 2019, pp. 265–275.

[103] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[104] Frank Huettner and Marco Sunder. "Axiomatic arguments for decomposing goodness of fit according to Shapley and Owen values". In: *Electronic Journal of Statistics* 6 (2012), pp. 1239–1250.

[105] J. D. Hyman et al. "Fracture size and transmissivity correlations: Implications for transport simulations in sparse three-dimensional discrete fracture networks following a truncated power law distribution of fracture size". In: *Water Resources Research* (2016).

[106] Jeffrey D. Hyman et al. "Identifying Backbones in Three-Dimensional Discrete Fracture Networks: A Bipartite Graph-Based Approach". In: *Multiscale Modeling & Simulation* 16.4 (2018), pp. 1948–1968. ISSN: 1540-3459. DOI: 10.1137/18m1180207.

[107] Jeffrey D. Hyman et al. "Predictions of first passage times in sparse discrete fracture networks using graph-based reductions". In: *Physical Review E* 96.1 (2017), pp. 1–10. ISSN: 24700053. DOI: 10.1103/PhysRevE.96.013304.

[108] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, pp. 448–456.

[109] J. Jaffré and J.E. Roberts. "Modeling flow in porous media with fractures; Discrete fracture models with matrix-fracture exchange". In: *Numerical Analysis and Applications* 5.2 (2012), pp. 162–167.

[110] Martin Jullum, Annabelle Alice Redelmeier, and Kjersti Aas. "Efficient and simple prediction explanations with groupShapley: A practical perspective". In: *Italian Workshop on Explainable Artificial Intelligence, XAI.it 2021*. 2021.

[111] M. Karimi-Fard and L.J. Durlofsky. "Unstructured Adaptive Mesh Refinement for Flow in Heterogeneous Porous Media". In: *ECMOR XIV-14th European conference on the mathematics of oil recovery*. 2014.

[112] Anuj Karpatne et al. "Theory-guided data science: A new paradigm for scientific discovery from data". In: *IEEE Transactions on knowledge and data engineering* 29.10 (2017), pp. 2318–2331.

[113] V. King, S. Rao, and R. Tarjan. "A Faster Deterministic Maximum Flow Algorithm". In: *Journal of Algorithms* 17.3 (1994), pp. 447–474. ISSN: 0196-6774. DOI: https://doi.org/10.1006/jagm.1994.1044. URL: https://www.sciencedirect.com/science/article/pii/S0196677484710443.

[114] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[115] T. N. Kipf and M. Welling. "Semi-supervised classification with graph convolutional networks". In: *Proceedings of the International Conference on Learning Representations*. 2017.

[116] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley, 2005. URL: https://www.cs.sjtu.edu.cn/~jiangli/teaching/CS222/files/materials/Algorithm%5C%20Design.pdf.

[117] Pang Wei Koh and Percy Liang. "Understanding black-box predictions via influence functions". In: *International conference on machine learning*. PMLR. 2017, pp. 1885–1894.

[118] Ron Kohavi and George H John. "Wrappers for feature subset selection". In: *Artificial intelligence* 97.1-2 (1997), pp. 273–324.

[119] Narine Kokhlikyan et al. "Captum: A unified and generic model interpretability library for pytorch". In: *arXiv preprint arXiv:2009.07896* (2020).

[120] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems* NIPS (2012). DOI: 10.1201/9781420010749.

[121] Indra Kumar et al. "Shapley Residuals: Quantifying the limits of the Shapley value for explanations". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 26598–26608.

[122] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[123] Qimai Li, Zhichao Han, and Xiao-ming Wu. "Deeper Insights Into Graph Convolutional Networks for Semi-Supervised Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. Apr. 2018. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11604.

[124] Lek-Heng Lim. "Hodge Laplacians on graphs". In: *Siam Review* 62.3 (2020), pp. 685–715.

[125] Zachary C Lipton. "The Mythos of Model Interpretability: In machine learning, the concept of interpretability is both important and slippery." In: *Queue* 16.3 (2018), pp. 31–57.

[126] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: (2017). Ed. by I. Guyon et al., pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[127] V.M. Malhotra, M.Pramodh Kumar, and S.N. Maheshwari. "An O(|V|3) algorithm for finding maximum flows in networks". In: *Information Processing Letters* 7.6 (1978), pp. 277–278. ISSN: 0020-0190. DOI: https://doi.org/10.1016/0020-0190(78)90016-9. URL: https://www.sciencedirect.com/science/article/pii/0020019078900169.

[128] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[129] Ivan Martino. "Cooperative games on simplicial complexes". In: *Discrete Applied Mathematics* 288 (2021), pp. 246–256. ISSN: 0166-218X. DOI: https://doi.org/10.1016/j.dam.2020.08.035. URL: https://www.sciencedirect.com/science/article/pii/S0166218X2030411X.

[130] Luke Merrick and Ankur Taly. "The explanation game: Explaining machine learning models using shapley values". In: *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer. 2020, pp. 17–38.

[131] Alessio Micheli. "Neural Network for Graphs: A Contextual Constructive Approach". In: *IEEE Transactions on Neural Networks* 20.3 (2009), pp. 498–511. DOI: 10.1109/TNN.2008.2010350.

[132] Andrzej Mlodak. "Some values for constant-sum and bilateral cooperative games". In: *Applicationes Mathematicae* 3.34 (2007), pp. 359–371.

[133] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. "Methods for interpreting and understanding deep neural networks". In: *Digital Signal Processing* 73 (2018), pp. 1–15.

[134] Federico Monti et al. "Geometric deep learning on graphs and manifolds using mixture model CNNs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5115–5124.

[135] Oskar Morgenstern and John Von Neumann. *Theory of games and economic behavior*. Princeton university press, 1944.

[136] Roger B Myerson. "Conference structures and fair allocation rules". In: *International Journal of Game Theory* 9.3 (1980), pp. 169–182.

[137] Roger B Myerson. *Game theory: analysis of conflict*. Harvard university press, 1991.

[138] Roger B Myerson. "Graphs and cooperation in games". In: *Mathematics of operations research* 2.3 (1977), pp. 225–229.

[139] John Nash. "Two-person cooperative games". In: *Econometrica: Journal of the Econometric Society* (1953), pp. 128–140.

[140] John F Nash Jr. "Equilibrium points in n-person games". In: *Proceedings of the national academy of sciences* 36.1 (1950), pp. 48–49.

[141] Nazri Mohd Nawi, Walid Hasen Atomi, and M.Z. Rehman. "The Effect of Data Pre-processing on Optimized Training of Artificial Neural Networks". In: *Procedia Technology* 11 (2013). 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013, pp. 32–39. ISSN: 2212-0173. DOI: https://doi.org/10.1016/j.protcy.2013.12.159. URL: http://www.sciencedirect.com/science/article/pii/S2212017313003137.

[142] John von Neumann. "Zur Theorie der Gesellschaftsspiele". In: *Mathematische Annalen 100, (Translated as "On the Theory of Games of Strategy", pp. 13–42 in Contributions to the Theory of Games, Volume IV (Annals of Mathematics Studies, 40) (A. W. Tucker and R. D. Luce, eds.), Princeton University Press, Princeton, 1959)* (1928), pp. 295–320.

[143] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. "Learning Convolutional Neural Networks for Graphs". In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 2014–2023. URL: https://proceedings.mlr.press/v48/niepert16.html.

[144] William S Noble. "What is a support vector machine?" In: *Nature biotechnology* 24.12 (2006), pp. 1565–1567.

[145] B. Nœtinger. "A quasi steady state method for solving transient Darcy flow in complex 3D fractured networks accounting for matrix to fracture flow". In: *J. Comput. Phys.* 283 (2015), pp. 205–223. ISSN: 0021-9991. DOI: http://dx.doi.org/10.1016/j.jcp.2014.11.038.

[146] B. Nœtinger and N. Jarrige. "A quasi steady state method for solving transient Darcy flow in complex 3D fractured networks". In: *J. Comput. Phys.* 231.1 (2012), pp. 23–38. ISSN: 0021-9991. DOI: http://dx.doi.org/10.1016/j.jcp.2011.08.015.

[147] Takayuki Oishi et al. "Duality and anti-duality in TU games applied to solutions, axioms, and axiomatizations". In: *Journal of Mathematical Economics* 63 (2016), pp. 44–53.

[148] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[149] Guilliermo Owen. "Values of games with a priori unions". In: *Mathematical economics and game theory*. Springer, 1977, pp. 76–88.

[150] Dino Pedreshi, Salvatore Ruggieri, and Franco Turini. "Discrimination-aware data mining". In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008, pp. 560–568.

[151] Vitali Petsiuk, Abir Das, and Kate Saenko. "RISE: Randomized Input Sampling for Explanation of Black-box Models". In: *British Machine Vision Conference (BMVC)*. 2018. URL: http://bmvc2018.org/contents/papers/1064.pdf.

[152] G. Pichot, J. Erhel, and J. de Dreuzy. "A generalized mixed hybrid mortar method for solving flow in stochastic discrete fracture networks". In: *SIAM Journal on scientific computing* 34 (2012), B86–B105. DOI: http://dx.doi.org/10.1137/100804383.

[153] G. Pichot, J. Erhel, and J. de Dreuzy. "A mixed hybrid Mortar method for solving flow in discrete fracture networks". In: *Applicable Analysis* 89 (2010), pp. 1629–643. DOI: http://dx.doi.org/10.1080/00036811.2010.495333.

[154] G. Pichot et al. "A Mortar BDD method for solving flow in stochastic discrete fracture networks". In: *Domain Decomposition Methods in Science and Engineering XXI*. Lecture Notes in Computational Science and Engineering. Springer, 2014, pp. 99–112.

[155] Maziar Raissi and George Em Karniadakis. "Hidden physics models: Machine learning of nonlinear partial differential equations". In: *Journal of Computational Physics* 357 (2018), pp. 125–141. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2017.11.039. URL: http://www.sciencedirect.com/science/article/pii/S0021999117309014.

[156] Maziar Raissi, P. Perdikaris, and Em Karniadakis George. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: http://www.sciencedirect.com/science/article/pii/S0021999118307125.

[157] W. Reilly. *Highway Capacity Manual*. USA: Transport Research Board, 2000.

[158] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "" Why should i trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.

[159] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Anchors: High-precision model-agnostic explanations". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[160] Hodson Richard. "Digital Revolution". In: *Nature, Vol. 563, No. 7733, Suppl. S131* (Nov. 2018).

[161] Alvin E Roth. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[162] Jennifer Rowley. "The wisdom hierarchy: representations of the DIKW hierarchy". In: *Journal of information science* 33.2 (2007), pp. 163–180.

[163] Cynthia Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.

[164] Gert Sabidussi. "The centrality index of a graph". In: *Psychometrika* 31.4 (1966), pp. 581–603. ISSN: 1860-0980. DOI: 10.1007/BF02289527. URL: https://doi.org/10.1007/BF02289527.

[165] Xavier Sanchez-Vila, Alberto Guadagnini, and Jesus Carrera. "Representative hydraulic conductivities in saturated grqundwater flow". In: *Reviews of Geophysics* 44.3 (2006), pp. 1–46. ISSN: 87551209. DOI: 10.1029/2005RG000169.

[166] Franco Scarselli et al. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605.

[167] David Schmeidler. "The nucleolus of a characteristic function game". In: *SIAM Journal on applied mathematics* 17.6 (1969), pp. 1163–1170.

[168] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[169] Lloyd S Shapley. "Notes on the n-Person Game—II: The Value of an n-Person Game.(1951)". In: (1951). URL: https://www.rand.org/content/dam/rand/pubs/research_memoranda/2008/RM670.pdf.

[170] Lloyd S. Shapley. "A value for n-person games". In: *Annals of Mathematics Studies* Contributions to the Theory of Games, II, volume 28 (1953), pp. 307–317.

[171] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3145–3153.

[172] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences". In: *International conference on machine learning*. PMLR. 2017, pp. 3145–3153.

[173] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps". In: *arXiv preprint arXiv:1312.6034* (2013).

[174] Leon Sixt, Maximilian Granz, and Tim Landgraf. "When explanations lie: Why many modified bp attributions fail". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9046–9057.

[175] Dylan Slack et al. "Fooling lime and shap: Adversarial attacks on post hoc explanation methods". In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020, pp. 180–186.

[176] Daniel Smilkov et al. "Smoothgrad: removing noise by adding noise". In: *arXiv preprint arXiv:1706.03825* (2017).

[177] J Springenberg et al. "Striving for Simplicity: The All Convolutional Net". In: *ICLR (workshop track)*. 2015.

[178] S. Srinivasan et al. "Model reduction for fractured porous media: a machine learning approach for identifying main flow pathways". In: *Computational Geosciences* (Mar. 2019). ISSN: 1573-1499. DOI: 10.1007/s10596-019-9811-7. URL: https://doi.org/10.1007/s10596-019-9811-7.

[179] Shriram Srinivasan et al. "Physics-informed machine learning for backbone identification in discrete fracture networks". In: *Computational Geosciences* Mc (2020). ISSN: 1420-0597. DOI: 10.1007/s10596-020-09962-5.

[180] Ari Stern and Alexander Tettenhorst. "Hodge decomposition and the Shapley value of a cooperative game". In: *Games and Economic Behavior* 113 (2019), pp. 186–198. ISSN: 0899-8256. DOI: https://doi.org/10.1016/j.geb.2018.09.006. URL: https://www.sciencedirect.com/science/article/pii/S0899825618301489.

[181] Erik Strumbelj and Igor Kononenko. "An efficient explanation of individual classifications using game theory". In: *The Journal of Machine Learning Research* 11 (2010), pp. 1–18.

[182] Erik Strumbelj, Igor Kononenko, and M Robnik Sikonja. "Explaining instance classifications with interactions of subsets of feature values". In: *Data and Knowledge Engineering* 68.10 (2009), pp. 886–904.

[183] Mukund Sundararajan, Kedar Dhamdhere, and Ashish Agarwal. "The shapley taylor interaction index". In: *International conference on machine learning*. PMLR. 2020, pp. 9259–9268.

[184] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 3319–3328.

[185] Svensk Kärnbränslehantering AB. *Data report for the safety assessment, SR-site*. Tech. rep. TR-10-52. SKB, Stockholm, Sweden, 2010.

[186] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).

[187] "The world's most valuable resource is no longer oil, but data". In: *The Economist* (May 2017).

[188] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. "Fooling automated surveillance cameras: adversarial patches to attack person detection". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2019, pp. 0–0.

[189] Tony Van Gestel et al. "Linear and nonlinear credit scoring by combining logistic regression and support vector machines". In: *Journal of credit Risk* 1.4 (2005).

[190] Felix Wu et al. "Simplifying graph convolutional networks". In: *International conference on machine learning*. PMLR. 2019, pp. 6861–6871.

[191] Zonghan Wu et al. “A Comprehensive Survey on Graph Neural Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2021), pp. 4–24. DOI: 10.1109/TNNLS.2020.2978386.

[192] H Peyton Young. “Monotonic solutions of cooperative games”. In: *International Journal of Game Theory* 14.2 (1985), pp. 65–72.

[193] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.