University of Turin
Department of Computer Science

RESEARCH DOCTORATE IN COMPUTER SCIENCE
XXXV Cycle

Embracing Complexity for Integrative Multi-Omics Approaches in Life Sciences

Nicola Licheri

Tutor:
Prof. Marco Beccuti
Prof. Francesca Cordero

Supervisor of the Doctoral Program:
Prof. Viviana Patti

Academic Year: **2022/2023**

Scientific Disciplinary Sector: **INF/01**

Acknowledgments

Ringrazio la gang dell'ufficio 24 e il Qbio Group tutto (Beatrice, Giulio, Laura, Simone, Riccardo, Irene, Daniele, Giulia, Elena, Sandro Gepiro, Daniela, Dora, Marco e la Prof.ssa Roberta Sirovich) per avermi accompagnato durante il mio percorso di dottorato, e coi quali ho condiviso gioie e dolori.

Un ringraziamento speciale va al Prof. Marco Beccuti e alla Prof.ssa Francesca Cordero per i loro preziosi consigli e insegnamenti, e per aver creduto in me e spesso più di me.

Grazie alla mia famiglia che mi ha sempre supportato in ogni scelta e decisione, permettendomi di conseguire questo risultato. Infine, ringrazio una persona a me molto cara, Federica, per essermi stata accanto e avermi dato forza lungo questo percorso tosto ma appassionante.

Abstract

Living systems are regulated by a myriad of complex reactions involving a plethora of biological actors, such as genes, transcripts, epigenetic modifications, non-coding RNAs, the microbiota community, and many others. With the advent of next-generation sequencing (NGS) technologies in 2005, Life Science research started to benefit from omics data, namely molecular snapshots of specific biological layers at fine-grained resolution, by reporting thousands of features per sample. The availability of omics data opened up a new era of computational challenges to deal with the volume, heterogeneity and complexity of the data, the lack of knowledge, and the missing parameters. In the literature, several approaches were proposed according to the questions addressed during the analysis. Most of these approaches might be classified into two groups, *tabular* and *graph* dataset, depending on the formalism exploited to represent the data. In the *tabular* dataset, samples are described by vectors of omics features, obtained by NGS data analysis workflow. Samples can be characterized on multiple omics layers, which, together with clinical information, enable multi-omics integration and offer a holistic point of view. Data-driven methods such as machine learning are crucial for dealing with these issues, but their workflow is not a straight line and often requires a trial-and-error approach. Differently, *graph* formalism is used to study the system by considering different levels of abstraction and by modeling the interactions characterizing the system under study through edges connecting labeled vertices, which are the elements of the domain under study. In these contexts, the identification of substructures (subgraph isomorphism problem) is a frequent but challenging task and requires advanced heuristics and indexing techniques to make it efficient. The choice of indexing data structure, as well as the type of topological features, are crucial for the development of efficient subgraph searching solutions. In the literature, there exist very efficient solutions in terms of running time, however, a relatively high amount of memory is required to store the index, compared to the other state-of-the-art approaches.

In this thesis, we face the challenges of both data analysis techniques. In particular, for multi-omics tabular representation, we propose an approach for multivariate filter feature selection defined as an optimization problem on a graph representation of the initial dataset. Alternatively, in the context of graph data, we proposed the use of decision diagrams, a family of graph-based data structures that allows compact encoding of large sets of structured data by exploiting their regularities and symmetries to reduce memory occupation. However, the efficiency of deci-

sion diagrams is strictly related to the development of domain-dependent heuristics to identify good variable orderings.

These theoretical results were then implemented into (i) a general framework for multi-omics integration via automated machine learning, called FeatSEE, and (ii) index-driven subgraph searching algorithm based on decision diagrams, called GrapesDD.

The novelties and strengths of the FeatSEE framework can be summarized into four points: (i) it is implemented in Python 3 and based on well-known libraries for data science, (ii) the implementation of high-level end-to-end containerized modules (e.g. graph-based feature selection) providing a friendly interface to the machine learning workflows (data preprocessing, feature selection and extraction, model selection and evaluation), (iii) a high level of portability and reproducibility granted by the containerization of the whole framework; (iv) a well-defined schema and related infrastructure to allow users to integrate their analysis workflow in the framework easily.

Its effectiveness was shown in different complex systems, such as the identification of a miRNA signature for Colorectal Cancer, a fluxomics application derived from Flux Balance Analysis on transcriptomics data, and an in vitro fertilization dataset reporting embryos' morphokinetics features.

In GrapesDD, decision diagrams have been investigated as indexing data structures for reducing the index's memory occupation without degrading the running time of the overall algorithm. The trie of the state-of-the-art tool software for parallel searching on biological graphs GRAPES, has been replaced with a decision diagram. It represents a new way of facing the graph indexing problem by exploiting the advanced manipulation algorithms offered by decision diagrams. Moreover, an entropy-based variable reordering heuristics has been proposed to further optimize the memory occupation of the index itself.

The effectiveness of these approaches was shown on real and artificial biological graph datasets.

List of publications

International journals

1. S. Canosa†, **N. Licheri**†, L. Bergandi, G. Gennarelli, C. Paschero, M. Beccuti, D. Cimadomo, G. Coticchio, L. Rienzi, C. Benedetto, F. Cordero‡, and A. Revelli‡. "A novel machine-learning framework based on time-lapse early embryo morphokinetics identifies an embryonic signature associated with day 5 expanded blastocyst development". In: Journal of Ovarian Research. 17.1 (2024): 63.

2. Barbara Pardini†, Giulio Ferrero†, Sonia Tarallo†, Gaetano Gallo, Antonio Francavilla, **Nicola Licheri**, Mario Trompetto, Giuseppe Clerico, Carlo Senore, Sergio Peyre, Veronika Vymetalkova, Ludmila Vodickova, Vaclav Liska, Ondrej Vycital, Miroslav Levy, Peter Macinga, Tomas Hucl, Eva Budinska, Pavel Vodicka, Francesca Cordero‡, and Alessio Naccarati‡. "A fecal miRNA signature by small RNA sequencing accurately distinguishes colorectal cancers: results from a multicentric study". In: *Gastroenterology* (2023)

3. **Nicola Licheri**, Vincenzo Bonnici, Marco Beccuti, and Rosalba Giugno. "GRAPES-DD: exploiting decision diagrams for index-driven search in biological graph databases". In: *BMC bioinformatics* 22 (2021), pp. 1–24

4. Luca Alessandri, Francesca Cordero, Marco Beccuti, **Nicola Licheri**, Maddalena Arigoni, Martina Olivero, Maria Flavia Di Renzo, Anna

Sapino, and Raffaele Calogero. "Sparsely-connected autoencoder (SCA) for single cell RNAseq data mining". In: *NPJ systems biology and applications* 7.1 (2021), p. 1

5. Luca Alessandri, Francesca Cordero, Marco Beccuti, Maddalena Arigoni, Martina Olivero, Greta Romano, Sergio Rabellino, **Nicola Licheri**, Gennaro De Libero, Luigia Pace, et al. "rCASC: reproducible classification analysis of single-cell sequencing data". In: *Gigascience* 8.9 (2019), giz105

6. Giulio Ferrero†, **Nicola Licheri**†, Lucia Coscujuela Tarrero, Carlo De Intinis, Valentina Miano, Raffaele Adolfo Calogero, Francesca Cordero, Michele De Bortoli, and Marco Beccuti. "Docker4Circ: a framework for the reproducible characterization of circRNAs from RNA-Seq data". In: *International Journal of Molecular Sciences* 21.1 (2019), p. 293

Book chapters

1. Giulio Ferrero, **Nicola Licheri**, Michele De Bortoli, Raffaele A Calogero, Marco Beccuti, and Francesca Cordero. "Computational Analysis of circRNA Expression Data". In: *RNA Bioinformatics* (2021), pp. 181–192

Publications on proceedings of scientific conferences

1. Riccardo Aucello, **Nicola Licheri**, Elena Rosso, Giulio Ferrero, Sandro Gepiro Contaldo, Simone Pernice, Pietro Liò, Francesca Cordero, and Marco Beccuti. "Functional data integration approach combining mechanistic models and multiphase feature selection methodologies". In: *18th conference on Computational Intelligence Methods for Bioinformatics & Biostatistics (CIBB 2023)*. September 1-5, 2023, Padova, Italy. 2023

2. Carlo Alberto Barbano, Marco Beccuti, Francesca Cordero, Desislav Nikolaev Ivanov, **Nicola Licheri**, Simone Pernice, Alberto Presta, Riccardo Renzulli, and Marco Grangetto. "Artificial intelligence methods for biomedical imaging and omics data". In: *Ital-IA 2023: 3rd National Conference on Artificial Intelligence, organized by CINI, May 29–31, 2023, Pisa, Italy* (2022)

3. **Nicola Licheri**, Elvio Amparone, Vincenzo Bonnici, Rosalba Giugno, and Marco Beccuti. "An Entropy Heuristic to Optimize Decision Diagrams for Index-driven Search in Biological Graph Databases." In: *CIKM Workshops*. 2021

4. Giulia Piaggeschi†, **Nicola Licheri**†, Greta Romano, Simone Pernice, Laura Follia, and Giulio Ferrero. "MethylFASTQ: a tool simulating bisulfite sequencing data". In: *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. IEEE. 2019, pp. 334–339

Contents

List of Publications **iv**

Contents **vii**

1 Introduction **1**

I Background **7**

2 Machine learning for multi-omics integration **8**

2.1 Omics data 9

2.1.1 Biomarkers 9

2.1.2 History of sequencing 10

2.1.3 Types of omics data 12

2.1.4 Workflow of analysis in bioinformatics 14

2.1.5 The reproducibility crisis 15

2.2 Machine learning 20

2.2.1 Definition 21

2.2.2 The workflow 26

2.2.3 The curse of dimensionality 29

2.3 Multi-omics data integration 32

2.3.1 Challenges 34

3 Graph Indexing **36**

3.1 Formalisation . . . 38
3.2 Filter-and-verification methods . . . 38
3.2.1 Index construction . . . 39
3.2.2 Filtering . . . 39
3.2.3 Verification . . . 40
3.3 State of the Art . . . 40
3.3.1 GRAPES algorithm . . . 42

4 Decision diagrams 45
4.1 General definition . . . 46
4.1.1 Notion of ordering . . . 47
4.1.2 Reduction rules . . . 47
4.2 Classes of decision diagrams . . . 48
4.2.1 Binary Decision Diagrams . . . 49
4.2.2 Multi-valued decision diagrams . . . 49
4.2.3 Multi-terminal decision diagrams . . . 49
4.2.4 Edge-valued decision diagrams . . . 50
4.3 Decision diagram implementation . . . 51

II Theoretical results 53

5 Graph-based feature selection for multi-omics integration 54
5.1 Feature graph formalisation . . . 54
5.1.1 Definition . . . 55
5.2 Optimization problem for mRMR on graph . . . 55
5.2.1 A metric for mRMR over the feature graph . . . 56
5.2.2 Evolutionary and genetic algorithms . . . 57
5.3 Feature selection via genetic algorithm . . . 58
5.3.1 Chromosome encoding . . . 58
5.3.2 Selection mechanism . . . 60
5.3.3 Reproduction mechanisms . . . 61

6 Decision diagrams applied to graph indexing 63
6.1 Decision diagrams for indexing . . . 63
6.1.1 Problem variables . . . 64
6.2 Encoding the problem . . . 65

6.2.1 Using multi-terminal 65
6.2.2 Using multi-way decision diagram 67
6.3 Index construction 68
6.4 Index filtering 71
6.4.1 Indexing the query graph 71
6.4.2 Feature extraction from the index MTMDD 73
6.4.3 Constraints verification 73
6.5 Variable ordering 74
6.5.1 An entropy-based metrics 74
6.5.2 The heuristic algorithm 75

III Applications and tool implementation 78

7 FeatSEE 79
7.1 The Framework 80
7.2 Python module 81
7.2.1 Data representation 81
7.2.2 Feature graph 83
7.2.3 Machine-learning models 85
7.3 FeatSEE end-to-end modules 87
7.3.1 Data preparation module 88
7.3.2 Evaluation module 90
7.3.3 Ensemble feature selection module 92
7.3.4 Filter-based feature selection module 93
7.3.5 Explainable feature extraction module 97

8 GRAPES-DD 100
8.1 Overview 101
8.2 Indexing stage 101
8.2.1 Index preparation 102
8.2.2 Index construction 102
8.3 Filtering stage 105
8.3.1 Query preprocessing phase 105
8.3.2 Feature extraction phase 105
8.3.3 Constraints verification phase 107
8.4 Entropy-based variable ordering 108

8.4.1 Label variables reordering 108

9 Applications **112**

9.1 Biomarker discovery for colorectal cancer 112

9.1.1 Introduction 113

9.1.2 Material and methods 116

9.1.3 Results 121

9.1.4 Discussion 130

9.2 An application combining mechanistic and data-driven approaches 137

9.2.1 Introduction 137

9.2.2 Material and methods 138

9.2.3 Results 142

9.2.4 Discussion 144

9.3 Explaining early embryonic development via time-lapse features 146

9.3.1 Context 146

9.3.2 Material and methods 147

9.3.3 Results 149

9.3.4 Discussion 152

9.4 Index-driven subgraph search exploiting decision diagrams . . . 155

9.4.1 Datasets description 155

9.4.2 Experimental setup and output 158

9.4.3 Discussion 168

10 Conclusion and future work **170**

Bibliography **176**

Chapter 1

Introduction

Living organisms are complex and difficult to understand and to be explained. They obey the laws of physics, but these laws are not enough to explain their behaviour. Components of a complex system participate in many different interactions, generating unpredictable emergent properties [1]. For instance, the gut microbiota, which consists of a community of millions of microbes, has recently been considered an organ due to the plethora of interactions with the host and its importance in health and disease [2]. Biology has done well in breaking down a complex system into its parts and understanding the purpose of each of these components. However, holistic approaches are needed to study a combination of the system's components, because the crucial emergent properties are not predictable by studying the single components in isolation, to reduce the system's complexity to an ordered sequence of transformation of information between the several layers of biological information.

The development of high-throughput Next-Generation Sequencing (NGS) has enabled the production of massive quantities of multi-omics data, profoundly impacting biomedical research by providing a more comprehensive understanding of biological systems and the underlying molecular mechanisms driving life and disease progression. Omics data are the layers of biological knowledge (e.g. genomics, transcriptomics, proteomics, metabolomics, etc.), and it follows that each of these layers is deeply connected to the others in

one or multiple directions.

Multi-omics data provides a window into the inner workings of the human body, allowing us to understand its complexities and how it functions in health and disease. By analyzing multi-omics data, we can gain insights into the molecular mechanisms that underlie human biology, including how genes are expressed, proteins are produced, and metabolites are regulated, and these insights can lead to novel discoveries about disease prevention, diagnosis, and treatment. Furthermore, multi-omics data can be used to model and simulate human physiology, which can help us understand how the body responds to different stimuli and interventions. This information can be used to develop personalized medicine approaches that are tailored to individual patients.

Multi-omics data are a concentration of knowledge and multiple approaches can be used to either extract or represent such a piece of knowledge. The choice of representation format is dictated by the specific research question and the type of analysis being performed. Researchers often employ a combination of formats to gain a comprehensive understanding of the biological system under investigation.

- **Capturing and storing knowledge** Ontologies and databases provide a structured foundation and centralized repositories enabling researchers to access and manipulate information easily, and a formal representation of knowledge in a specific domain, enabling the definition of concepts, relationships, and properties of the domain that provides a structured and consistent way to represent and share knowledge, enabling reasoning, inference, and knowledge discovery, respectively.

- **Data-driven methods for unveiling underlying patterns** Statistical models and machine learning algorithms empower researchers to extract meaningful insights from multi-omics data. Statistical models, rooted in probability and statistics, identify hidden correlations between different omic layers and disease phenotypes, revealing potential biomarkers, therapeutic targets, and risk factors. Machine learning algorithms, harness the power of data-driven modeling, predict disease outcomes, classify biological samples, and identify drug targets, paving the way for personalized medicine and accelerated drug discovery.

- **Expressing relationships with graphs** Graphs are mathematical structures widely used for expressing complex relationships among ele-

ments when representing different types of omics data, as well as biomedical and biological information. On top of these representations, several analyses are performed. These interconnected nodes and edges, representing biological entities and their interactions, reveal patterns and relationships that would remain hidden in raw data, shedding light on regulatory processes, signaling pathways, and disease mechanisms.

In this thesis, our attention is directed towards two distinct yet closely interconnected subjects, namely data-driven methods for biomarker discovery in multi-omics contexts, and the utilization of graph-based representations, including subgraph searching within collections of graphs. Although data are represented using different formalisms, the overarching goals can be similar, as both endeavours aim to identify patterns, be they biomarkers or subgraphs, in large and heterogeneous collections of biological data.

In the context of data-driven methods, we consider multi-omics data as they are obtained from bioinformatics workflows of analysis processing raw data produced by NGS technologies. The output of NGS data analysis is composed of large matrices describing each sequenced sample using a quite large number of molecular measurements known as features, such as transcript quantification (transcriptomics, miRNomics), DNA methylation (methylomics), and copy number variations (CNVs) or the presence of mutations (genomics). Machine learning, as the study of algorithms that allow computer programs to automatically improve through experience, allows us to exploit the large amount of available data to infer logic and patterns by exploiting general learning algorithms without the need to develop an ad-hoc solution as in traditional programming. However, the application of machine learning techniques is not trivial and several challenges have to be taken into consideration. One of the main challenges is related to the high dimensionality of the data, known as the curse of dimensionality, which is strictly related to the high number of features respecting the number of available samples, which is generally quite low. This phenomenon brings several problems such as noisiness and error-pronesses of the data, as well as the abundance of redundant and irrelevant information.

Differently, in the context of graphs for knowledge representation, a common task is the search for one substructure within one graph, called target. The problem is referred to as one-to-one subgraph search, and it is known to be NP-complete. Heuristics and indexing techniques can be applied to facilitate

the search. Indexing techniques are also exploited in the context of searching in a collection of target graphs, referred to as one-to-many subgraph problem. Filter-and-verification methods that use indexing approaches provide a fast pruning of target graphs or parts of them that do not contain the query. The expensive verification phase is then performed only on the subset of promising targets. Indexing strategies extract graph features at a sufficient granularity level for performing a powerful filtering step. Features are memorized in data structures allowing efficient access. The choice of the data structure and the type of features to be stored within the index are crucial, and determining the indexing size, querying time and filtering power, which are key points for the development of efficient subgraph searching solutions.

In the plethora of available data structures, we consider decision diagrams (DDs), which are a relatively niche but powerful tool that can be used to solve a wide variety of problems. Decision diagrams are symbolic data structures represented as directed acyclic graphs (DAGs) that allow us to represent and manipulate multivariate functions whose variables have structured domains. They have been extensively used in industrial hardware verification and model checking, due to their ability to symbolically encode complex boolean functions having a huge number of input variables, and to exploit the structure and regularity of the data for an efficient representation in the DAG. However, the size of the DD representing a given function depends critically on the variable ordering¸and the identification of the optimal variable ordering is NP-complete. As a consequence, the efficiency of the applications based on decision diagrams are strongly dependent on the development of domain-specific heuristics to select a good ordering.

Finally, this thesis is organized into three parts:

1. **Background**

   In this first part, the main topics and notation used in the rest of the thesis are introduced.

   In Chapter 2, a thorough introduction to omics data is presented, highlighting the challenges associated with their acquisition through the application of bioinformatics workflows. The chapter also delves into the difficulties related to ensuring the computational reproducibility of these analyses. Subsequently, a comprehensive overview of machine learning is provided, addressing its main challenges and introducing feature engi-

neering approaches, encompassing both feature selection and feature extraction techniques. Finally, this chapter illustrates the state-of-the-art integration strategies for multi-omics data driven by machine learning. Chapter 3 introduced the use of graphs in life science and the challenge of subgraph searching, as well as the different approaches for graph indexing and the state of the art in the literature. Finally, a state-of-the-art tool called GRAPES, which has been shown to have good performance in terms of speed-up for both one-to-one and one-to-many cases is illustrated in detail. Finally, Chapter 4 provides a general overview of decision diagrams, a family of graph-based data structures that efficiently allow us to represent large sets of structured data by exploiting representation based on directed acyclic graphs. In particular, decision diagrams represent functions having many variables and possibly different domains, and these functions can be efficiently manipulated through specific operators.

2. **Theoretical results**

   In the second part, Chapter 5 focuses on the introduction of a new graph-based formalism called the feature graph, to represent feature redundancies in a multi-omics dataset, together with feature importances concerning a given target to be predicted. Exploiting such a formalism, we define a metric for evaluating generic sets of feature based on the graph topology, which is then exploited to define a feature selection algorithm as an optimization problem to maximize the defined metric.

   Chapter 6 is dedicated to the application of decision diagrams for graph indexing. In particular, there are proposed different ways to represent the index using different types of decision diagrams, and a strategy to efficiently filter the index given a query graph, by applying operators between decision diagrams. Finally, we presented an entropy-based heuristics for variable reordering intending to identify a quasi-optimal variable ordering that allows to reduce the memory occupation of the index.

3. **Tools and applications**

   In the last part, we introduced the implementations of the above theoretical results (Chapters 7-8), and some case studies faced with such implementations (Chapter 9).

Specifically, in Chapter 7, the graph-based feature selection algorithm has been implemented and included as a standalone module in the FeatSEE framework, which allows non-expert people to build user-defined workflow of analysis exploiting machine learning techniques without the need for advanced skills. Then, Chapter 9 provides three case studies faced using FeatSEE: biomarker discovery in Colorectal Cancer (CRC) miRNome data (Section 9.1), functional data integration with fluxomics data (Section 9.2), and explainable data analysis of an in-vitro fertilization dataset (Section 9.3).

On the other part, Chapter 8 introduces GRAPES-DD, a modified version of the GRAPES tool in which a decision diagram has been applied as the core indexing data structure. Moreover, since the order of the variables within the decision diagram is crucial for its efficiency, a heuristic variable reordering algorithm based on entropy estimation is presented. Then, in Chapter 9, experimentation results are presented: indexing and filtering stages using both real and synthetic collections of graphs, as well as the application of the variable reordering heuristic algorithm on real graph collections (Section 9.4).

Part I

Background

Chapter 2

Machine learning for multi-omics integration

The big data era in which we are all living is characterized by the ever-increasing volume, velocity, and variety of data that is being generated. In the biological and clinical fields, such a paradigm shift is carried on by the large-scale generation of omics data, which are created by measuring the abundance of different biological macromolecules (e.g. DNA, transcripts, metabolites) exploiting high-throughput and next-generation sequencing technologies. This data can be effectively used to gain insights into the complex interactions among biological molecules within the cell, predict disease risk, and develop personalized treatments. However, the sheer volume and complexity of multi-omics data make it difficult to analyze using traditional methods. Machine learning techniques offer a solution to this problem since such algorithms can actually learn from omics datasets to identify patterns and make predictions. This allows researchers to make better decisions based on data, and to automate tasks that would otherwise be more time-consuming and error-prone.

This chapter is composed of three sections. The first section provides a general overview of different classes of omics data, which describe different layers of biological knowledge, and the sequencing technologies and software workflows to produce such kind of data starting from raw biological materi-

als. The second section introduces machine learning techniques, by providing an overview of its three basic ingredients, namely the experience, the task and the performance evaluation. Then, the classical workflow for machine learning applications and the challenges to face and possible strategies are illustrated. Finally, the third section provides an overview of machine-learning-driven multi-omics integration strategies and of the main challenges to cope with.

2.1 Omics data

Life sciences encompasses a wide range of topics, from the structure and function of cells to the evolution of organisms. As the basic unit of life, cells are the smallest units that can live on their own and that make up all living organisms. The biology of the cell is a complex and dynamic system. Cells are made up of many different molecules, organelles, and proteins that interact with each other in a complex way. To understand how cells work and cooperate, we need to be able to measure and analyze these macromolecules and their interactions.

Deep sequencing technologies have made it possible to obtain a huge number of molecular measurements within a tissue or cell. These technologies can be applied to a biological system of interest to obtain a snapshot of the underlying biology at a resolution that has never before been possible. Broadly speaking, the scientific fields associated with measuring such biological molecules in a high-throughput way are called *omics* [3]. Computational methods such as bioinformatics and systems biology allow researchers to develop a new understanding of the molecular and genetic basis of physiological and pathological conditions. By measuring, in each patient sample, thousands of biological characteristics, which are generally called biomarkers, scientists are identifying previously unknown, molecularly defined disease states and searching for complex biomarker signatures that predict responses to therapy and disease outcomes.

2.1.1 Biomarkers

Definition 1 (Biomarker)**.** *A biomarker is a measurable biological characteristic that can indicate the presence, or progression of a disease, or predict an*

individual's response to treatment. Biomarkers may be detected and analyzed in tissue and surrogate tissues (i.e. blood, urine, stool, saliva, sputum, etc.).

Definition 2 (Biomarker signature)**.** *A group of biomarkers that are used together to provide a more comprehensive assessment of a disease or condition is called biomarker signature.*

In particular, for the bioinformatics context, biomarkers are typically molecular measurements obtained from deep sequencing technologies. Biomarkers can be used in a variety of settings, including clinical trials, diagnostic testing, and disease monitoring. In particular, they belong to one or more of the following categories:

- diagnostic biomarkers determine the presence and type of a specific disease or condition (e.g. cancer);
- prognostic biomarkers give information on the patient's overall disease outcome with or without treatment;
- predictive biomarkers help to identify which treatment the patient is most likely to respond to or benefit from [4].

2.1.2 History of sequencing

The modern history of DNA sequencing began in 1977, when Sanger reported his method to identify the order of nucleotides of DNA fragments [5]. However, this technique is a low-throughput sequencing method that processes one fragment at a time. It is based on the synthesis of complementary DNA fragments of starting genetic material, which was then read using gel electrophoresis.

In the next decade, there were new improvements in the Sanger method, and, in 1987, Applied Biosystem developed the first automated DNA sequencer [6].

In 1990, the ambitious Human Genome Project[7] began in the United States, under the direction of the National Institute of Health and the U.S. Department of Energy, with a 15-year and $3 billion plan to complete the genome sequence. The goal of this project was to determine the complete sequence of the human genome in order to understand human evolution, the causation of diseases and so on[6].

In 1998, Applied Biosystem developed an automated high-throughput capillary DNA sequencer based on the Sanger method. This device, along with whole-genome random shotgun method, has been used to perform human genome sequencing. The sequenced fragments were then assembled using dedicated assembly algorithms. Human genome sequencing was initiated in September 1999 and completed in June 2000; the first assembly was completed in October 2000 [6].

The automated Sanger method has dominated the sequencing world for almost two decades. Despite many technical improvements, its own limitations showed a need for new and better technologies for sequencing large genomes.

NGS became available in 2005. The major advance of NGS is the ability to produce an enormous volume of data, cheaply and in a short amount of time. In particular, the sequencing of a genomic region multiple times, sometimes hundreds or even thousands of times is known as *deep sequencing*. Deep sequencing is fundamental for the detection of rare clonal types, cells, or microbes comprising as little as 1% of the original sample. It is useful for studies in cancer, microbiology, and other research involving the analysis of rare cell populations.

The availability of multiple instruments, produced by different manufacturers, represented a paradigm shift from the past, where a single instrument produced by Applied Biosystem dominated the market. Many of these innovative approaches were initially developed for the 1000 Genome Project[8], whose purpose was to provide a global description of common human Single Nucleotide Polymorphism (SNP)s by applying genome and exome sequencing to many individuals from 26 different populations.

This so-called 'massively parallel' sequencing technology differs significantly from the Sanger method. Although each manufacturer has its own technology, all devices share a set of attributes. First, the initial preparatory steps are fewer and easier to perform than Sanger sequencing. NGS procedures begin with the construction of a library formed by ligating platform-specific synthetic DNAs (adapters) onto the ends of the fragments to be sequenced. The adapters allow the fragments to be attached on a solid surface during amplification and sequencing. Second, library fragments are amplified by a polymerase-mediated reaction that produces thousands of copies of each fragment; this step is required so that the sequencing reaction produces a signal detectable by the instrument's optical system. Amplification also provides a

source of sequencing error because polymerases are not 100% accurate. Third, these instruments perform the sequencing of all fragments at the same time, through a series of repeating steps that are performed and detected automatically.

2.1.3 Types of omics data

The field of omics is rapidly growing, and new omics disciplines are being developed all the time. As these new disciplines emerge, they will provide us with even more insights into the complex workings of biological systems.

Omics data can be directly derived either from (i) deep sequencing experiments through a dedicated workflow, such as transcriptomic, epigenomics, metabolomics, and metagenomics, or from (ii) pre-existing omics by exploiting ad-hoc prediction tools and public databases, such as fluxomics, and interactomics.

Transcriptomics

RNA-seq is the NGS method that sequences the transcriptome, that is, all the RNA transcript sets expressed by the genome in cells, tissues, and organs at different stages of an organism's life cycle [9, 10, 11]. RNA-seq provides technical reliability and sensitivity and unambiguous maps of the transcribed regions of the genome with high accuracy in quantitative expression levels, identification of tissue-specific transcript variants and isoforms (SNPs and mutations), transcription boundaries and splicing events, transcription factors, and small and long noncoding RNAs (ncRNA) involved in the regulation of gene expression (circRNAs, miRNAs, lncRNAs, etc.) [12].

Epigenomics

Epigenomics is the study of the heritable chemical modifications of the DNA that affect how genes are expressed without changing the DNA sequence itself. Even though almost each cell in an organism shares an identical genotype, organismal development generates a multitude of cell types with distinct profiles of gene expression, and thus different functions. Cellular differentiation may be considered an epigenetic phenomenon, governed by a multitude of molecular mechanisms, among DNA methylation, histone modification and others [13].

Methylomics Methylomics is the genome-wide analysis of DNA methylations and their effects on gene expression and heredity [28]. Methyl-seq uses NGS to analyze and map DNA cytosine methylation at single-base resolution usually by employing bisulfite DNA sequencing [24, 25].

Metagenomics

Beyond genomics, there is metagenomics, namely the study of the total genomic content of a microbial community. The total DNA and/or RNA is isolated from a microbial population without prior cultivation, sequenced, and compared with previously known sequences to identify known species or to discover previously unknown species. Metagenomics enable the characterization of the microbes colonize the human body (i.e. *microbiome*) in numbers that are estimated to outnumber human genes and somatic cells by more than 100-fold. These microbes (viruses, prokaryotes, and eukaryotic microbes) occupy various anatomical habitats including gut, skin, vagina, and oral mucosa and are believed to markedly influence human physiology, nutrition, and health [14, 15].

Metabolomics

Metabolomics is the study of an organism's total metabolic response to an environmental stimulus or a genetic modification [16]. Metabolomics data also provide biochemical and physiological snapshots of processes that are obtained from cellular and tissue experimental studies

Fluxomics

Constraint-based approaches are mathematical modeling approaches based on the definition and manipulation of stoichiometric matrices, commonly used with optimization techniques, such as using linear and mixed-integer programming to maximize an objective function under specific constraints. In particular, the Flux Balance Analysis (FBA) [17] computes the distribution of reaction fluxes in a metabolic system at the equilibrium and finds the feasible fluxes under given constraints and an objective function to maximize/minimize [18].

The use of constraint-based methods such as the FBA to design models of metabolite flow in microbes has connected "omic" to phenotypes in the science of Fluxomics [19].

Interactomics

Transcriptomic and other complex functional genomics data sets that arise from high-throughput experimental biology benefit from analysis in the context of known cellular networks, which provide a holistic framework for interpretation. Although far from complete, large-scale networks have been determined for numerous organisms, including humans and other model organisms [20, 21, 22, 23, 24]. These networks, or interactomes, are commonly represented as graphs, in which nodes correspond to biological components (for example, genes, RNAs, proteins or metabolites), and edges correspond to known interactions among them (for example, physical, regulatory or genetic). Integrative interactomics analyses are typically premised on modularity, which is a key organizational property of cellular networks in which molecules that work together to carry out a specific biological process are enriched in interactions among themselves [25].

2.1.4 Workflow of analysis in bioinformatics

NGS technologies can be applied to different biological materials, such as DNA, RNA or proteins. The output of NGS experiments is composed of FASTQ files, which is a text-based format for representing either nucleotide sequences (DNA/RNA) or amino acid (protein) sequences, which are commonly called reads.

Generally, the NGS data analysis comprehends (i) data cleaning and quality control, (ii) data alignment against some reference, and (iii) data quantification. However, the particular workflow of analysis, and the tools to be used, depend on the specific NGS data under analysis. In any case, the analysis of NGS data is a complex and challenging process that requires expertise and different specialized software.

Despite the availability of a vast set of computational tools and methods for data analysis, it is still challenging for a researcher to organize these tools, integrate them into workable pipelines, find accessible computational platforms, configure the computing environment, and perform the actual analysis.

The difficulties in creating these complicated computational pipelines, installing and maintaining software packages, and obtaining sufficient computational resources tend to overwhelm bench biologists and prevent them from attempting to analyze their data.

2.1.5 The reproducibility crisis

The reproducibility of research is a key element in modern science. It represents the ability to replicate an experiment independently by the location and the operator. Therefore, a study can be considered reproducible only if all used data are available and the exploited computational analysis workflow is clearly described. However, for reproducing a bioinformatics analysis, the raw data and the list of tools used could not be enough to guarantee the reproducibility of the results [26].

Issues

Bioinformatics data analysis pipelines usually consist of several third-party software packages. Each tool has many implicit dependencies on other programs and libraries required by the chosen environment. As a consequence, a workflow built in a specific environment has little chance of running correctly in a different environment without significant effort. Furthermore, software updates might lead to sneaky deployment and reproducibility issues.

The first barrier to computational reproducibility in science has nothing to do with the technological aspects discussed here but rather a reluctance to publish the code used to obtain the results. Many researchers assume that summary algorithm descriptions or pseudo-codes are sufficient to understand methods used in data processing and analysis.

As regards the technical aspects, the common issues that pose barriers to reproducing the original results are the following ones.

Dependency hell Often a software package depends on libraries or other software to work properly. Dependency hell occurs when software works abnormally, and displays errors and bugs. These phenomena can be due to incorrect dependencies integration, conflicting programs or other causes.

Imprecise documentation Documentation on how to build and/or install the software is a frequent barrier to replication, particularly for not practical

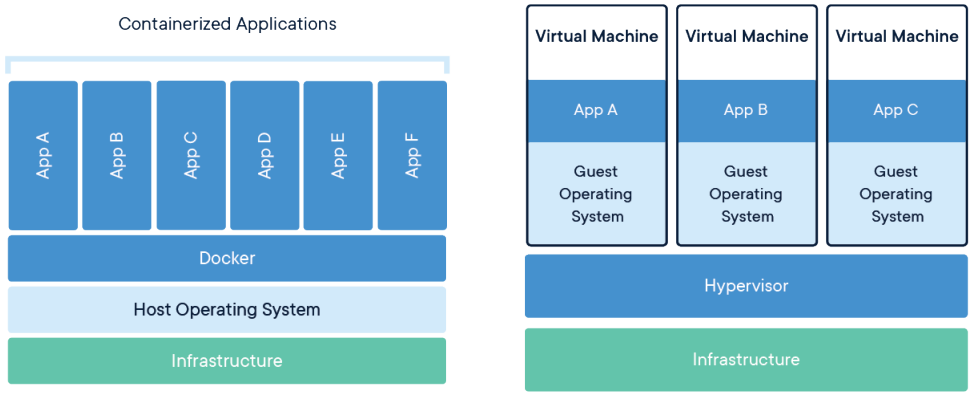


Figure 2.1: Containers and Virtual Machines. Containers and virtual machines have similar resource isolation and allocation benefits but function differently. Containers share the operating system with the host machine. Virtual machines perform hardware virtualization through a hypervisor software. Hypervisor allows running one or more operating systems on top of it as if they run on a physical machine. Adapted from [27].

users in such kind of activities. Superficial documentation of tool parameters is another obstacle to reproducing specific studies.

Code rot Software dependencies are not static units but receive updates that may fix bugs, add new features and deprecate the old ones. Any update can potentially change the software behaviour, and so the results generated by the dependent tool.

Barriers to adoption and re-use Typically in scientific computing, the coordination among multiple tools is managed via Makefiles and various scripts. As more and more functionality is added, the difficulty of coordinating multiple tools continues to increase. As a consequence, it increases also the maintenance effort of the entire software architecture.

Virtualization technologies

Virtualization technologies were proposed to mitigate the issues listed so far, even if they cannot completely solve the issues without additional efforts.

Virtualization technologies provide a way of abstracting hardware resources (e.g. CPUs, memory, storage, networks, etc) from the applications that use them. There are two main types of virtualization technologies, namely Virtual

Machine (VM) approaches, and containerization technologies. On one hand, VMs create an isolated operating system environment. This means that each VM can run its own operating system, and the applications running on one VM cannot see or interact with the applications running on another VM. On the other hand, containerization technologies allow us to package an application and all its dependencies into a single entity (i.e. image) that can be run sharing the host machine's kernel. Despite the differences between the two approaches, virtualization is a powerful technology that offers several improvements regarding resource efficiency, scalability, security, and manageability of software and workflows.

Virtual machines A VM is essentially emulating a real computer (guest machine) that executes programs like a real computer. VMs run on top of a physical machine (host machine) using a hypervisor, a piece of software that performs hardware virtualization. The guest machine contains a copy of the OS files, all the packages and libraries needed to run the application and a virtualized hardware stack. As a result, the size of the image of a VM is generally quite heavy. Moreover, this approach is a kind of black box and thus ill-suited for reproducibility, because they are not systematically described or accessible with a standard tool or protocol. As a consequence, other research cannot easily extend the virtual machine in a consistent and scalable way [28].

OS-level virtualization OS-level virtualization technology was recently proposed in the area of bioinformatics as an efficient solution to simplify the distribution, usage and maintenance of bioinformatics software [29]. OS-level virtualization exploits the kernel of an operating system of the host to create isolated execution environments, which are called *containers* and can be used to run different applications on the same physical hardware. A popular implementation of OS-level virtualization technology is the Linux Container project (LXC), from which both Docker and Singularity [30] are based.

OS-level virtualization is more lightweight than the VM-based approaches, which use a hypervisor to handle the communication between the VM itself and the physical hardware. This is because containers share the kernel of the host operating system, which means that they do not need to have their copy of the OS. This makes containers more lightweight, efficient and easier to manage. Containerization technologies allow us to run applications in an

isolated environment and to efficiently distribute the package, in the form of images, in a portable manner across different platforms. In a similar way to VMs, such images provide that all the required software is already installed, configured and tested.

In conclusion, VMs and OS-level virtualization are similar in their goals, since they both provide (i) **analysis portability**, isolating an application into a self-contained unit that can run anywhere, and (ii) **analysis reproducibility**, freezing the version of tools and library used. The main difference between the two approaches is that the VM must include a full copy of the operating system, whereas OS-level virtualization share the host machine's kernel with other containers (Fig. 2.1). OS-level virtualization provides an abstraction at the app layer, whereas VMs are a physical hardware abstraction. As a result, the former is much more lightweight and with higher performance than virtual machines, even though VMs are more flexible and isolated, and hence, secure.

Docker Among the container platforms proposed in the literature, Docker (`http://www.docker.com`) is getting the standard environment to quickly build, deploy, scale and manage containerized applications under Linux systems. In summary, Docker's strengths are its high level of portability, which allows users to easily register and share containers over different hosts, and to achieve more effective resource use and a faster deployment compared with other similar software. It is an open-source project based on LXC, that allows OS-level virtualization, portable deployment of containers across platforms, and git-like versioning, among others.

Docker images can be created interactively, but this approach makes the image opaque. The correct way to proceed is through a Dockerfile, a plain-text file similar to a Makefile, which adheres to a specific syntax and uses a specific set of instructions to build a given Docker image. The Dockerfile is processed by the Docker daemon to automatically build the image. A Docker image is composed of multiple layers stacked on top of each other, where each layer generally corresponds to an instruction in the Dockerfile. Such layers are cached to be reused during future builds and shared among different images. Adopting Dockerfiles has many advantages:

- Images can be very large, whereas a Dockerfile is just a small plain text file that can be easily stored and shared.

- Text files are suited for use with a version management system such as git, which can track the evolution of the Dockerfile.
- It provides human-readable support for all the requirements to execute the tool. It includes also all software dependencies down to the level of the OS.
- It is very easy for other users to extend or customize the image by editing the Dockerfile.

Changes in the dependencies can be reduced because Docker defines the software environment to a particular Linux distribution. It prevents potential problems, but this cannot completely avoid the challenge of code-rot. To deal with it, Docker allows us to archive a binary copy of the working image. It is portable and can be read by other Docker installations, providing a way to run the exact versions of all software involved. To simplify the sharing process, Docker provides a cloud platform, Docker Hub, in which pre-built images can be stored and downloaded by other users. It also supports image versioning by attaching tags to the image. Docker Hub is a free service and its code is open source, so that one can run its own private version on its own server.

The most obvious advantage of this approach is to replace the tedious installation of numerous pieces of software, with complex dependencies, by simply downloading a single pre-built ready-to-run image containing all the software already configured. Another advantage of Docker is that each run of an image is executed in an isolated container, created starting from the immutable image. This prevents possible conflicts with other installed programs and guarantees that each process runs in a predictable system configuration that cannot change over time.

Multiple containers can run on the same machine at the same time, sharing the OS kernel with other containers. Containers take up much less space than VMs, so the system can handle more applications with fewer resources. Studies show that Docker containers introduce a negligible overhead for CPU and memory performance and that applications running in a container perform equally or better when compared to traditional virtual machine technology[28].

Enhancing computational reproducibility

As said before, virtualization technologies are very helpful but are not sufficient to completely resolve the issues of reproducibility in bioinformatics. In 2013, Sandve et al. propose 10 simple rules for enhancing the reproducibility in computational research, including (i) keep track of intermediate results and of how results are produced, (ii) avoid manual data manipulations in favour of automated scripts, (iii) version control of those scripts, (iv) take note of random seeds and (v) provide public access to data, metadata and code [31]. Sandve's rules are the foundations of the Reproducible Bioinformatics Project (RBP) [32], whose aim is to achieve a framework for developing reproducible workflows of analysis by means of (i) a set of Docker images wrapping all the tools needed and (ii) a R library.

To further enhance the development of workflows of analysis, software solutions known as Workflow Management Systems (WfMS) definitely help to automate and manage the flow of work within pipelines and workflows, by (i) validating input and intermediate results, (ii) handling conditional step executions, (iii) applying parallelization whenever is possible, (iv) handling error, and many others. There exists a vast plethora of more than 150 different WfMSs that differs among them through their interface (e.g. local or remote), the IT skills required to be used, and the platform and system requirements among others. An example of such software is NextFlow, a workflow management system that uses Docker technology for the multi-scale handling of containerized computation to allow users to write self-contained and truly reproducible computational pipelines [33].

2.2 Machine learning

Machine learning is the branch of artificial intelligence (AI) focused on the development of systems able to solve a task without being explicitly programmed. In recent years, machine learning techniques have become essential and pervasive tools in both academic and industry contexts, due to the availability of (i) large datasets and (ii) high-performance systems, and (iii) the development of more sophisticated algorithms.

This section is composed of three parts that provide a brief introduction to the main themes related to the broad field of machine learning. The first part provides a brief introduction to the fundamental concepts. The second part is

devoted to the main challenge in machine learning that is known as *curse of dimensionality*, namely the approaches to cope with high-dimensional data. The third part describes the typical workflow of machine learning applications.

2.2.1 Definition

In [34], Mitchell provided a general definition of what machine learning is:

> *A computer program A is said to learn from experience E concerning some task T and performance measure P if its performance P at task T improves with experience E.*

The Experience

The concept of experience, namely the knowledge source from which machine learning algorithms learn, is known as *dataset*: a collection of n examples, also known as samples of instances, which are described by a set of m *features*, one of the building blocks of machine learning. Generally speaking, a dataset $\mathcal{D} = (\mathcal{X}, \mathcal{M})$ is composed of (i) the feature matrix $\mathcal{X} \in \mathbb{R}^{n \times m}$ representing examples on rows and features on columns, and (ii) $\mathcal{M}$ the collection of the dataset's metadata (e.g. feature names, feature types, samples' ID, and other samples metadata).

Features refer to individual measurable properties or characteristics of the objects or entities belonging to the dataset $\mathcal{D}$. Let F_i indicates a generic feature and let X_i be the domain of such a feature. Since features can represent almost anything that is measurable, these can be categorized with respect to their semantics, which influence the characteristics of the feature domains. Features can be distinguished as (i) quantitative, (ii) ordinal and (iii) categorical (also known as nominal), with respect to the measures of the central tendency they support among the mean, median and mode.

- *Quantitative features* have a meaningful numerical scale and admit the calculation of all three measures of central tendency.
- *Ordinal features* represent categories that have an order but not scale. They can be used to compute the median and mode, but not the mean.
- *Categorical features* describe discrete properties (e.g. gender) that have neither order nor scale. Only the mode can be estimated from them.

Formally, let $\mathcal{D} = (\mathcal{X}, \mathcal{M})$ be an (unlabeled) **dataset** composed of a $n \times m$ feature matrix $\mathcal{X}$ reporting n examples described by m features $F_1, \ldots, F_m$, and the collection of dataset metadata $\mathcal{M}$, comprehending features' names, types, domains, etc.

The feature matrix $\mathcal{X} \in \mathbb{R}^{n \times m}$ reports the examples $\{x_1, \ldots, x_n\}$ in the form of m-length feature vectors $x_i = \langle x^1, \ldots, x^m \rangle$ for $i = 1, \ldots, m$. Let $x_i^j \in \mathcal{X}_i$ be the value of feature F_j belonging to the example x_i, which is defined of the feature domain $\mathcal{X}_i$, which is a generic subset of $\mathbb{R}$. Let denote $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_m$ as the *instance space*, that represent the set of structured data $\mathcal{X}$ in a (possibly very) high-dimensional space.

Finally, we distinguish between unlabeled and labeled datasets. In the unlabeled dataset $\mathcal{D}_U = (\mathcal{X}, \mathcal{M})$, examples x_i are "simply" described by features, and they are exploited by unsupervised learning algorithms to discover hidden patterns in the data.

Differently, the labeled dataset $\mathcal{D}_L = (\mathcal{X}, \mathcal{Y}, \mathcal{M})$ includes the target vector $\mathcal{Y} \in \mathbb{R}^n$, which builds a mapping $f : \mathcal{X} \to \mathcal{Y}$ from examples x_i to a desired output (i.e. target) y_i for all $i = 1 \ldots, n$. Supervised learning algorithms take advantage of labeled datasets to learn an approximation of the underlying relationship $\hat{f} : \mathcal{X} \to \hat{\mathcal{Y}}$ between the features and the targets.

Task

Tasks refer to specific problems that machine learning algorithms aim to solve. A task defines the goal the algorithm aims to achieve using the provided data and training process. Tasks are addressed by *models*, which are the outputs of a particular learning algorithm applied to a training dataset.

Tasks can be categorized as predictive and descriptive. The goal of predictive tasks is to take advantage of historical data to build a model able to make predictions about future events. This is achieved by learning salient patterns and relationships in the data, which are then used to make predictions on new and unseen data. On the other hand, the descriptive tasks' goal is to explain the data rather than make predictions. They focus on summarizing and interpreting data to gain insights and understand patterns and relationships within the data.

There exist a variety of tasks. In the following, there are presented some of them, which are grouped with respect to the type of dataset they experience.

1. **Unsupervised tasks** identify properties and interesting patterns intrinsic in unlabeled datasets.

    - Clustering is the task of grouping similar examples based on the similarities within the features, in order to discover the underlying structure of the data without prior knowledge.
    - Anomaly detection consists in the identification of anomalies in the data, namely instances that considerably differs from the norm or expected patterns.

2. **Supervised tasks** exploit the labeled dataset to learn some mapping $f : X \to Y$ from the features to some targets.

    - In regression tasks, the goal is to learn a mapping from instances to a continuous target.
    - In classification tasks, the goal is to learn a mapping from instances to one of N_c categories C. In particular, one can distinguish between binary and multiclass classification tasks, which hold $N_c = 2$ and $N_c > 2$, respectively.

As said before, a task is addressed by a model, which is created from a learning algorithm given a dataset. Fixed the kind of task $\mathcal{T}$, there are several different learning algorithms to create as many models that allow us to achieve $\mathcal{T}$. Moreover, models can be broadly grouped into three non-mutually exclusive categories, concerning the mathematical interpretation of the corresponding learning algorithms.

1. *Geometric models* represent instances as points in a high-dimensional Euclidean space and exploit spatial concepts (e.g. distances, lines, planes) to make decisions.

2. *Probabilistic models* assume there is an underlying random process modelling the relationship between the data x and the target variable y, and try to model it using probability distributions.

3. *Rule-based models* partition the instance space into instance space segments through a set of logical rules, using either a list or a tree-based structure.

Moreover, multiple individual models can be combined through different techniques into an *ensemble model*, in order to improve overall performance and robustness [35]. Of course, there are multiple approaches to combining multiple models in an ensemble.

- *Bagging*, which stands for *bootstrap aggregation*, involves training multiple models independently on bootstrap replicates of the dataset (i.e. random samples with replacement from the original training set) [36]. The predictions from individual models are then combined through averaging (for regression) or majority voting (for classification) to obtain the final prediction. Random Forest [37] is a popular bagging-based algorithm that uses decision trees as base models.

- *Boosting* involves a series of weak models that are sequentially trained to correct the errors made by the previous models. Each subsequent model is trained to focus more on the instances that were incorrectly predicted by the previous models [38]. Examples of boosting algorithms include AdaBoost [39] and Gradient Boosting [40].

- *Stacking* combines predictions from multiple models by training a meta-model that learns to make predictions based on the outputs of the individual models. It allows the meta-model to learn the optimal way of combining the predictions from the base models, potentially achieving higher performance.

Performance

There exists a vast plethora of performance measures, which are task-specific and used to evaluate the abilities of machine learning models. The choice of which performance metrics to consider may seem straightforward, but it strongly depends on the specific goal of the application under consideration.

For instance, the goal of classification tasks is to discriminate among $n_c \geq 2$ classes. The confusion matrix is used to summarize the performance of classifiers by means of a $n_c \times n_c$ matrix such that rows describe the actual labels and columns represent the predicted labels. Then, the element (i, j) counts how many times the model classifies as c_j the examples labeled as c_i.

For example, the generic confusion matrix for binary classification is shown in Table 2.1. True positives (TP) and true negatives (TN) are correctly labeled

by the model, as positive and negative, respectively. In contrast, false positives (FP) are the examples that belong to the negative class but are recognized as positive, and false negatives (FN) are examples belonging to the positive class but incorrectly classified as negatives.

		Predicted Class	
		Positive	Negative
Actual Class	Positive	TP	FN
	Negative	FP	TN

Table 2.1: Confusion matrix for binary classification task

The confusion matrix enables the estimation of a series of performance metrics, as well as the class distribution of the data. Let $Pos = TP + FN$ and $Neg = TN + FP$ be the number of actual positive and negative examples in the dataset.

Accuracy Is the proportion of examples for which the model predicts the correct output.

$$accuracy = \frac{TP + TN}{Pos + Neg}$$

Precision and recall The recall and precision metrics provide insights into a model's ability to correctly identify positive instances (recall) and the accuracy of its positive predictions (precision), respectively. They are defined as follows:

$$recall = \frac{TP}{TP + FN} = \frac{TP}{Pos} = tpr$$

$$precision = \frac{TP}{TP + FP}$$

It is worth noting that recall and precision are often inversely related: increasing one metric may result in a decrease in the other. Achieving a balance between recall and precision is a common challenge in classification tasks, and the harmonic mean of these metrics, known as the F1-score, is often used to assess the overall performance of a model.

$$F1 - score = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Area under the ROC curve A widely employed performance metric for binary classification that allows us to condense the confusion matrix is the AUC-ROC, which stands for Area Under the Receiver Operating Characteristic Curve. The ROC curve is a probability curve plotting TPR (i.e. true positive rates) on y- axis against FPR (i.e. false positive rates) on the x-axis and provides a performance measure across all possible classification thresholds. The AUC value is interpretable as the probability that the model ranks a random positive example more highly than a random negative example. A higher AUC indicates better performance: an AUC equal to one means a perfect classifier, whereas a random classifier has an AUC of 0.5.

2.2.2 The workflow

The machine learning workflow involves a series of steps for developing and deploying machine learning models. It typically begins with data collection and preparation, followed by feature engineering and model selection. The selected model is then trained and evaluated using appropriate performance metrics. Finally, the trained model is deployed into a production environment, where it can make predictions on new, unseen data. The workflow often includes iterations and fine-tuning to improve model performance and ensure the desired outcomes are achieved.

Problem definition

Defining a problem involves clearly understanding the task at hand and determining how machine learning can be applied to solve it. The fundamental steps to define a problem in machine learning are to (i) identify the objectives, (ii) define the task type and (iii) identify input and output data.

Data collection

Gather the relevant data required for the ML task. This may involve collecting data from various sources, such as databases, APIs, or external datasets.

Data preparation

The primary objective of data preparation is to transform the raw data into a clean, structured, and consistent format that is suitable for the subsequent steps in the machine learning pipeline, such as feature engineering, model training, and evaluation.

Feature engineering

Feature engineering involves the process of selecting, transforming, and generating new features from the given data to improve the performance and effectiveness of machine learning models. This also involves dimensionality reduction and feature transformation techniques to fulfil the requirement of specific learning algorithms.

Model selection

The choice of the appropriate model that is suitable for the problem at hand depends on different factors, such as the type of task, the nature of the data and the available data and computational resources. A vast plethora of available machine learning algorithms exist. The choice of the most suitable depends on the type of task, the nature of the data, and the available data and computational resources.

Model training and evaluation

Usually, there is interest in evaluating how well the machine learning algorithm performs on data that it has not seen before, as a proxy of real data used by a model deployed in the real world. Therefore, the performance measures are evaluated on a test set, namely a bunch of data that is separate from the data used for training.

Cross-validation techniques Cross-validation (CV) techniques are exploited to get the average behaviour of the model being used to resample different training and test set pairs from the original training set.

There exist different CV techniques, of which the most common ones are k-fold CV and leave-one-out CV (LOOCV).

- In k-fold CV, the dataset is split into k equally sized folds. A total of k models are trained, each one using $k-1$ folds as the training set and the remaining as test set. A model is trained and evaluated k times, each time using a different fold as the validation set and the remaining $k-1$ folds as the training set.

- In Leave-One-Out CV (LOOCV), a dataset composed of n examples is split into as many folds: each example is treated as a separate validation set, and n models are trained on $n-1$ samples and tested on the remaining ones.

Once cross-validation is terminated, a set of performance metrics, one for each fold, is obtained. By averaging the results from multiple folds of cross-validation, you can reduce the variance of the results and get a more accurate estimate of the model's performance. Moreover, cross-validation can be repeated multiple times to increase the number of observations and obtain a more robust final estimation. In the case of imbalanced datasets, stratification strategies can be used to guarantee that each fold has the same class distribution as the original dataset.

Hyperparameter tuning

Hyperparameters in machine learning algorithms are settings or configurations that are not learned from the training data but are required to be set before training the model. They have a significant impact on the performance of the model and need to be carefully chosen to achieve the best results. The specific hyperparameters can vary depending on the learning algorithm you are using, and the process of systematically optimizing the hyperparameters of a machine learning algorithm is known as hyperparameter tuning. Hyperparameter tuning involves experimenting with different combinations of these settings to find the best configuration that maximizes the model's accuracy, generalization, or other relevant performance metrics. This process typically requires running multiple training iterations with different hyperparameter values and selecting the combination that yields the best results on a validation dataset.

2.2.3 The curse of dimensionality

ML algorithms aim to learn the underlying patterns within the data and to build models able to generalize. Unfortunately, this process is not so straightforward and two main issues may occur.

1. **Overfitting**. A model is said to *overfit* the data when it has a high bias, and then it is not able to generalize. It happens either when (i) the model is too complex (i.e. too many parameters), or (ii) there are too few available data.

2. **Underfitting**. A model is said to *underfit* the data when it has a high variance and then it is unable to capture the relationships between features because it is too simple (i.e. not enough parameters), or because there are not enough training data.

These issues are strictly related to the curse of dimensionality, which arises when the number of dimensions (i.e. features) in a data set increases. Indeed, the number of available samples is generally limited for different reasons (e.g. costs, rare events), whilst the number of features describing each sample can be arbitrarily large, especially in omics and multi-omics contexts.

The curse of dimensionality has important consequences in the learning process. As the number of dimensions increases, the data points become more and more spread out, which can lead to data sparsity. This means that there may be very few data points in each dimension, which can make it difficult to identify patterns in the data. Moreover, as the number of dimensions increases, the data becomes more sensitive to noise. Then, even small amounts of noise can have a significant impact on the data, which can lead to issues for machine learning algorithms.

The curse of dimensionality is strictly related to the Garbage-In Garbage-Out (GIGO) principle, which states that the quality of a trained model is only as good as the quality of the training data. This means that if the data is not clean and well-prepared, then the machine-learning algorithm will not be able to produce accurate results.

To deal with the curse of dimensionality and the GIGO principle is important to make sure that the data is clean and well-prepared before being used. Moreover, dimensionality reduction techniques, such as feature extraction and feature selection, allow for a reduction in the number of dimensions

in the dataset, which improves the interpretability of the data and reduces the sensitivity to noise.

Feature extraction

Feature extraction techniques create a new set of representative features that capture the underlying patterns or characteristics in the data. They are often methods based on matrix decomposition techniques, such as Principal Component Analysis (PCA), Non-negative Matrix Factorization (NMF), and Independent Component Analysis (ICA).

Feature selection

Feature selection is the process of identifying only the most informative features concerning the task at hand, and removing the noisy non-informative, irrelevant, and redundant features. This can help to reduce data sparsity and improve the performance of machine learning algorithms, as well as reduce model complexity (i.e. avoid overfitting) and improve interpretability (less is more).

Given a set of m features, there are 2^m possible feature sets. The exponential number of candidate solutions makes not feasible the exhaustive search. Therefore, most approaches apply a 'greedy' search algorithm that never reconsiders the choices it makes. Feature selection techniques can be broadly categorized into four main categories, as follows:

1. **Filter methods** are independent of the learning algorithm and perform a statistical analysis over the features space to select a discriminative subset of features. Filter methods can be broadly categorized into univariate and multivariate approaches. Univariate methods assess feature relevance independently, utilizing a specific criterion. While they excel at identifying irrelevant features, they struggle to eliminate redundant ones. This is because univariate filter methods solely evaluate features in isolation, and do not consider the redundancies between them. In contrast, multivariate methods incorporate feature correlation into their assessment process, enabling them to address both irrelevant and redundant features. However, while multivariate methods generally outperform univariate methods, they come at the cost of increased computational complexity. A relevant approach for filter-based feature selec-

tion the so-called minimum Redundancy Maximum Relevance (mRMR) method, which was initially proposed to cope with some relevant bioinformatics problems [41].

2. **Wrapper methods** exploit a search algorithm, either deterministic or guided by a heuristic, that is wrapped around a classification model to evaluate candidate feature subsets by training and testing a model. The idea is that feature selection is 'wrapped' in a search procedure that usually involves training and evaluating a model with a candidate set of features [42].

3. **Embedded methods** integrate the feature selection step during the learning procedure into a single process. During the training step, the classifier adjusts its internal parameters and determines the appropriate weights/importance given to each feature to produce the best classification accuracy [43].

4. **Hybrid methods** firstly reduce the features through the application of a filter method, then the reduced feature set is passed through a wrapper or embedded method to obtain the final feature subset [44, 45].

The combinatorial nature of the feature selection problem allows us to face this problem in several ways. An interesting high-level approach is switching from tabular representation of the dataset to a graph-structured one, which opens the gates to a plethora of different population-based metaheuristic approaches such as Evolutionary Algorithm (EA) and Genetic Algorithm (GA), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) that allow to advantage of the graph topology in the feature selection process. In the literature, different graph-based feature selection methods have been proposed based on filter [46, 47] and hybrid [48] approaches.

Ensemble feature selection Analogously to ensemble learning, ensemble feature selection combines the results of multiple feature selection algorithms to select a final set of features. This can be done in a variety of ways, but the most common approach is to use a voting system. In a voting system, each feature selection algorithm votes for the features that it believes are most important. The features that receive the most votes are then included in the final set of features.

Ensemble feature selection has several advantages over traditional feature selection algorithms. First, it can improve the accuracy of the feature selection process. This is because ensemble methods can help to mitigate the biases of individual feature selection algorithms. Second, ensemble feature selection can be more robust to noise in the data. This is because ensemble methods can combine the results of multiple algorithms, which can help to reduce the impact of noise on the feature selection process.

Data perturbation involves creating multiple data subsets by randomly sampling the original dataset. Each data subset is then used to train a separate feature selection algorithm. The features that are consistently ranked highly across the different data subsets are then included in the final set of features. Function perturbation involves training multiple feature selection algorithms on the original dataset. The different feature selection algorithms can be different types of algorithms, or they can be different parameter settings of the same algorithm. The features that are selected by the majority of the feature selection algorithms are then included in the final set of features.

2.3 Multi-omics data integration

Recent interest has focused on measuring multiple omics data types on a single set of samples, in order to integrate different types of molecular measurements into an omics-based test. Such multidimensional datasets have the potential to provide deep insight into biological mechanisms and networks, allowing for the development of more powerful clinical diagnostics.

1. **Early Integration**.

   Early integration methods consider developing a model using a joint data matrix that is obtained by combining multiple omics datasets. Commonly, early integration does not require any pre-processing and simply consists of the concatenation of individual omics to form a single large matrix of multi-omics data, which is then used for supervised or unsupervised analysis.

2. **Mixed Integration**.

   The mixed integration strategy addresses the shortcomings of the early integration by transforming independently each omics dataset into a

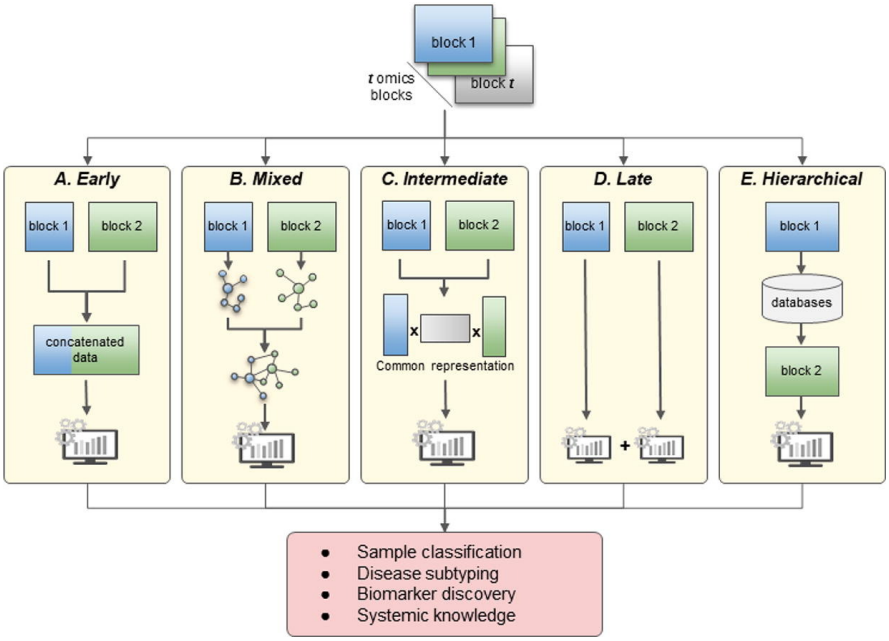


Figure 2.2: Integration strategies. Image from [49].

new representation, such as graphs or kernel matrices, which are then combined to construct the total model. Graphs provide a formal means to transform and portray relationships between different omics. Kernel methods enable the transformation of data from its original space into a higher dimensional feature space, in which simple linear models can be trained.

3. **Intermediate strategies**

   Intermediate integration strategies create multiple intermediate models for the different omics data and then build a final model from various intermediate models. Integration approaches facilitate the understanding of interactions amongst different omics for a certain phenotype (for example, survival in pancreatic cancer). The final multi-dimensional joint model can be built using an ML algorithm that uses the most relevant variables from each omics model.

4. **Late Integration**

   The most straightforward integration strategy is to apply machine learning models separately on each dataset and then combine their respective predictions, namely Late integration. Its strength relies on its capacity

to use readily available tools designed specifically for each omics type, and compared to the other strategies, it does not suffer the challenges of trying to assemble different kinds of data. The shortcoming of such an integration strategy is that it cannot capture inter-omics interactions and at no point in the learning process can the different machine learning models share knowledge and utilize the complementarity information between omics.

5. **Hierarchical Integration**

   Hierarchical integration strategies are characterized by integrating external information, such as public databases and scientific literature.

2.3.1 Challenges

Many challenges arise when integrating multi-omics datasets. Some of them are general to machine learning approaches, whereas others are specific to the life science field.

Among the challenges related to the machine learning field (i) the presence of missing values and (ii) imbalanced datasets.

Missing data can be addressed in different ways. If enough samples are available, listwise deletion, namely removing samples with missing data, may be acceptable. If not, different statistical methods can be used to impute the missing values.

Class imbalance occurs when the distribution of classes in the training dataset is skewed, which can be a significant problem when working on rare events, such as an uncommon trait in a population. Several methods can be used to resolve this problem [50], such as sampling and cost-sensitive learning. Sampling tries to balance the dataset before the integration process, where either the majority class is randomly under-sampled, or the minority class is oversampled by creating new artificial observations, or a combination of both methods [51, 52, 53]. Cost-sensitive learning is directly integrated into the algorithm and balances the learning process by giving more weight to misclassified minority observations.

Life science challenges are related to the noisiness and complexity of biological data, as well as their error-prone for multiple causes (e.g. batch effect). Data heterogeneity is a major challenge in omics research that must be handled correctly. Omics datasets can have different data distributions or

data types, as well as differ vastly in size, namely in the number of features. The heterogeneity matter is strictly connected to the curse of dimensionality because the high number of features is often paired with a relatively small sample size, for economic reasons, scarcity of the phenotype of interest, lack of volunteers, etc. Moreover, relevant patterns often involve many molecules from different omics layers, which implicates the need to consider multiple omics at the time [54]. The different integration strategies presented above address those problems differently, by either reducing the number of features, transforming the input data into a more concise representation, or integrating them at the end of the analysis [49, 55].

Chapter 3

Graph Indexing

Graphs are mathematical objects used to represent items, also called vertices, and relations between them. In the bioinformatics context, they are exploited to express relationships at any biochemical, biological, and medical level. For example, graphs can represent physical molecule structure by expressing chemical bonds among atoms [56]. At the cellular system level, graphs are instead applied to represent biological actors, such as genes, proteins, or RNAs, and their relations, such as physical interactions or causal inference [57, 58, 59]. Differently, in medical applications, graphs are exploited in decision support systems to connect patient data with disease states and treatments [60]. For what concerns integration and analysis of multi-omics data, graphs are becoming popular for integrating biomedical information with data regarding multiple omics. In such a model, items compose a heterogeneous set of biological and meta-biological objects. Graphs of genetic interactions are enriched by embedding their relationship with diseases, drugs, anatomic phenotypes, biological functions, or cellular localization. Then, multi-omics data linked to the genetic actors are integrated. The result is a knowledge base that can be exploited for drug repurposing, for prioritizing disease-associated genes, or for patient classification and biomarker identification [61, 62, 63].

A key computational task regarding graphs is the search for specific topologies contained within them, which is known to be NP-complete. Such task

is widely involved in many bioinformatic approaches as well as in the field of computational chemistry. Subgraph searching is a preliminary step in finding motifs in biological networks [64, 65, 66]. Network motifs are statistically over-represented sub-structures. They are building blocks of complex networks[67]. Several types of motifs have been discovered [68] such as the feed-forward loops that define patterns in gene regulatory networks [69]. Detection of motifs is a computationally challenging problem which requires the exhaustive search of subgraphs within a given network. Subgraph searching is also applied for tuning model parameters in biomolecular simulations [70]. In this context, graph-based representation of molecules facilitates the searching of fragments in large collections of molecules. Reliable model parameters are estimated based on the frequency of retrieved fragments. Moreover, collections of metabolic networks are queried in order to identify conserved pathways [71]. Because of the complexity of the querying task, many approaches limit the search to simple structures such as paths or small subgraphs[72]. Subgraph searching is also applied for biological network alignment, that is a powerful instrument for predicting functionalities of newly discovered elements [73]. Alignment can exploit the search of small subgraphs, also called seeds, within the set of networks that have to be aligned, in order to reduce the computational time requirements [74]. Other alignment tools, such as RINQ [75], use indexing schemes. Index-based strategy drives the alignment process to specific portion of the graphs and avoids expensive computations. Subgraph searching is also a baseline procedure in biomedical database systems [1] consisting of genes, compounds, diseases, symptoms, side effects and annotations, integrated in networks. The networks are queried in order to prioritize gene-disease associations [61] or for drug re-purposing studies [76]. However, querying biological networks is a challenging task which, in many cases, increases its complexity with the query size [77].

In this chapter we formally introduce the subgraph isomorphism problem on labelled graphs and the index-based approaches to cope with such a problem. Finally, a state-of-the-art software is presented.

[1] `https://het.io/`

3.1 Formalisation

Let $D = \{G_1, \ldots, G_m\}$ be a collection of connected labeled graphs. Graphs within the collection are either directed or undirected. Directed graphs are characterized by edges having a direction, then each edge has a source vertex and a destination vertex. In the undirected case, the edges do not have a direction, then each edge can be traversed in either direction. Labels are domain-specific knowledge that is attached to the vertices of the graphs. Let $\Sigma = \{\sigma_1, \ldots, \sigma_T\}$ be the label alphabet of D and let us introduce the labeling function $M : V \to \Sigma$ mapping a graph vertex to its label.

Definition 3 (Graph)**.** *A graph G_i is a triple (V_i, E_i, L_i), where:*

- *$V_i = \{v_j\}, j = 1, \ldots, n_i$ is the set of vertices in G_i;*
- *$E_i = \{(v_k, v_j), j, k = 1, \ldots, n_i\}$ is the set of edges linking vertices in V_i*
- *$L_i = \{M_i(v_j) \forall v_j \in V_i\}$ is the set of labels of vertices of G_i.*

Definition 4 (Graph isomorphism)**.** *Two graphs $G_i = (V_i, E_i, L_i)$ and $G_j = (V_j, E_j, L_j)$ are isomorphic if and only if there exists a bijective function $I : V_i \to V_j$ mapping each vertex of G_i to a vertex of G_j such that $(u, v) \in E_i$ if and only if $(I(u), I(v)) \in E_j$, and vice versa. For labeled graphs, the constraint of label compatibility must also be respected: $M(v) = M(I(v)) \forall v \in V_1$.*

Definition 5 (Subgraph isomorphism)**.** *A subgraph isomorphism of the query graph $Q = (V_q, E_q, L_q)$ in the target graph $G(V, E, L)$ is an injective function $I : V_q \to V$ such that $(u, v) \in E_q$ if and only if $(I(u), I(v)) \in E$, $M(u) = M(I(u))$, and $M(v) = M(I(v))$.*

3.2 Filter-and-verification methods

The naive way to cope with the subgraph isomorphism problem is to check the query graph for subgraph isomorphism against each graph in the dataset, but the NP-completeness of the problem makes such an approach infeasible.

Different algorithms have been proposed to improve the performances of the combinatorial search by exploiting heuristic methods for pruning the search space, such as VF2 [78] and VF3 [79], or by changing the order in which query vertices are matched, such as RI [80].

However, when the number of graphs in the collection is relatively high, or when the target graphs have relatively large sizes, this procedure often gets too time-consuming. To this end, multiple methods based on graph indexing have been proposed in order to reduce the set of graphs against which to test for subgraph isomorphism.

These solutions are based on an index built on top of structural features extracted from the graphs. Then, given a query, the index is exploited to filter out either portion of or entire graphs that definitely do not contain the query. The graphs that passed out the filtering phase, which are usually much smaller in size than the whole dataset, may actually be a false positive (i.e. do not contain the query) and are finally tested for subgraph isomorphism with an exact approach.

The algorithms belonging following the filter-and-verification approach operate in three stages: (i) index construction, (ii) filtering and (iii) verification.

3.2.1 Index construction

Structural features are extracted from the graph in the collection and stored in an appropriate data structure, such as a hash table or a tree-based structure. Different algorithms take into consideration different kinds of features. The choice ranges from simple paths, trees, cycles, subgraphs, or a combination of these. The feature extraction is based either on the exhaustive enumeration of all features of a particular kind, or using mining-based approaches. Mining-based approaches require a high amount of time because of the mining step, however, they are able to build more compact indexes with respect to the approaches based on the exhaustive enumeration. Moreover, some algorithms attach location information (e.g. the ID of the first vertex) to the features. Regardless of the strategy actually used, a limit is imposed on the size of the indexed features, where the size is defined as the number of edges comprising it. Once the index is built, it is then saved to secondary memory, from which it will be loaded for future uses.

3.2.2 Filtering

Given a query graph, the filtering stage initially proceeds to extract from it the features of the same form as those used to create the index. Then, these features are then matched against the index resulting in a *candidate set*,

namely a set of graphs containing all the features of the query, hence possibly matching the query itself.

3.2.3 Verification

The candidate set provided by the filtering stage may contain false positives, since candidate graphs may contain all the features of the query graph but not the graph itself. Therefore, the verification stage consists of the application of an exact subgraph isomorphism algorithm against the graphs in the candidate set returned by the filtering stage.

3.3 State of the Art

The subgraph searching problem consists in finding a query graph within a target graph. It is a well-studied computational problem which is known to be NP-complete [81]. A generalization of such formulation considers more than one target graph. This is typically referred to as *one-to-many* in contrast to the original formulation that is referred to as *one-to-one*. Techniques for solving the one-to-one problem are mainly based on heuristics to speed-up the searching of a mapping function. Instead, the main efforts for solving the one-to-many problem are focused on developing a good filtering strategy for discarding target graphs belonging to the collection that do not contain the query graph. Performance in terms of construction time, size, querying time and filtering power are key concepts for their development. Such a performance is strictly related to the type of feature that is taken into account.
In details one-to-one approaches can be divided in two categories: *pure subgraph isomorphism* and *assisted solvers*. The first category is composed by algorithms that are focused on improving the performances of the combinatorial search by exploiting heuristic methods for pruning the search space, such as VF2 [78] and VF3 [79], or by changing the order in which query vertices are matched, such as RI [80]. The second category comprises algorithms able to efficiently reduce the number of target vertices that are candidate to match with query vertices. This reduction is obtained by indexing the target graph and by comparing the features assigned to target vertices with those of the query vertices. Indexing means that a predefined type of features are extracted from the graph and they are stored in a data structure in order to recognize in which parts of the graph, or in which graphs of a collection, a given fea-

ture occurs. Once candidates are retrieved, this information is also used for generating a quasi-optimal ordering of the query vertices. In this perspective, GraphQL [82] uses a pseudo subgraph isomorphism test, while TurboISO [83] exploits a tree-structured auxiliary index, and CSL [84] postpones Cartesian products with a matching order that prioritizes the query vertices in the core structure, similar to RI.
One-to-many approaches can be differentiated by the type of features they take into account (e.g. paths, trees, cycles or subgraphs) and how they extract them. GraphGrep [85], GraphGrepSX [86], GRAPES [87] and SING [88] extract paths by indexed graphs with simple enumeration procedures, but they differ in the type of data structure and additional information they use. Simple enumeration is also used by CT-Index [89] for extracting trees and cycles, and by GDIndex [90] and GCode [91] for extracting subgraphs. On the contrary, mining-based algorithms recognize *frequent* features with *ad hoc* procedures. SwiftIndex [92] and TreePi [93] extract frequent trees, as well as Tree+Delta [94] which also retrieves frequent substructures. Mining of subgraph is also performed by CP-index [95], gIndex [96], FG-Index [97] and Lindex+ [98]. Alternatively, signatures based on the pairs of vertex labels of the graphs can be exploited [99]. Mining-based approaches require high amount of time because of the mining step, however they are able to build more compact indexes with respect to the approaches based on the exhaustive enumeration.

In recent years, one-to-one approaches have reached a high performance. In many cases, they outperform the indexing methodologies of one-to-many approaches by simply scanning all the target graphs in a collection. However, when the number of graphs in the collection is relatively high, or when the target graphs have relatively large size, indexing techniques are still predominant, and hybrid approaches are investigated [100]. In [101], authors proposed an algorithm for the one-to-many problem which exploits a technique that it is usually embedded in one-to-one approaches, such as GraphQL, TurboIso and CFL. The technique consists in a pre-processing step for detecting the set of target vertices that are most probable to be matched with a given query vertices by looking at their connectivity. Authors have equipped the verification phase of GraphGrepSX, GRAPES and CT-Index with such a technique showing that modified one-to-many algorithm, in particular GRAPES, sensibly outperform GraphQL, TurboIso and CFL for the verification step. However,

such a modification is added up to the original indexing techniques of the algorithms, thus it only helps in increasing the filtering power but it does not solve problems linked to the size and build time of the original index. Similar considerations can be done for cache-assisted frameworks [102, 103]. In this perspective, compression of the index plays a central role for both one-to-one and one-to-many approaches [104, 105].

3.3.1 GRAPES algorithm

A performance study [106] reports that index-based approaches have several issues in building indices on large graph databases in terms of number of distinct labels, number of vertices in data graphs, density of target graphs and number of target graphs due to their poor time and space efficiency of index construction. Among the tested algorithms, GRAPES showed the best performance in terms of running time. However, its index requires a relatively high memory amount compared to the other approaches. GRAPES is implemented both as sequential and parallel software using symmetric multiprocessing (SMP) architectures. In addition, GRAPES was developed for achieving good performance in the collection of graphs as well as in scanning a query over a single large target graph.

Indexing phase

GRAPES indexes a database of labeled graphs $G = \{g_1, \ldots, g_n\}$ using labeled paths up to length l as features. For each path of the target graphs, GRAPES stores the identification of its starting vertices and the number of its occurrences in each graph.

In GRAPES labelled paths are stored in a trie, a tree structure which compacts paths by their longest common prefixes. Given two labelled paths, $\hat{p} = (\sigma_1^p, \sigma_2^p, \ldots, \sigma_l^p)$ and $\hat{q} = (\sigma_1^q, \sigma_2^q, \ldots, \sigma_l^q)$, that share the first i labels, $(\sigma_1^p, \sigma_2^p, \ldots, \sigma_i^p) = (\sigma_1^q, \sigma_2^q, \ldots, \sigma_i^q)$, a branch, starting from the root of the tree, is built in order to represent the shared part of the paths. Then, the branch is split into two different branches that represent the non-shared suffixes of the paths, $(\sigma_{i+1}^p, \ldots, \sigma_l^p)$ and $(\sigma_{i+1}^q, \ldots, \sigma_l^q)$. Information regarding the starting vertices, v_1^p and v_1^q, is stored on the corresponding leaves of the branches, as well as the number of times each path occurs in each target graph. The internal organization of the trie is shown in Figure 3.1. If only paths of the

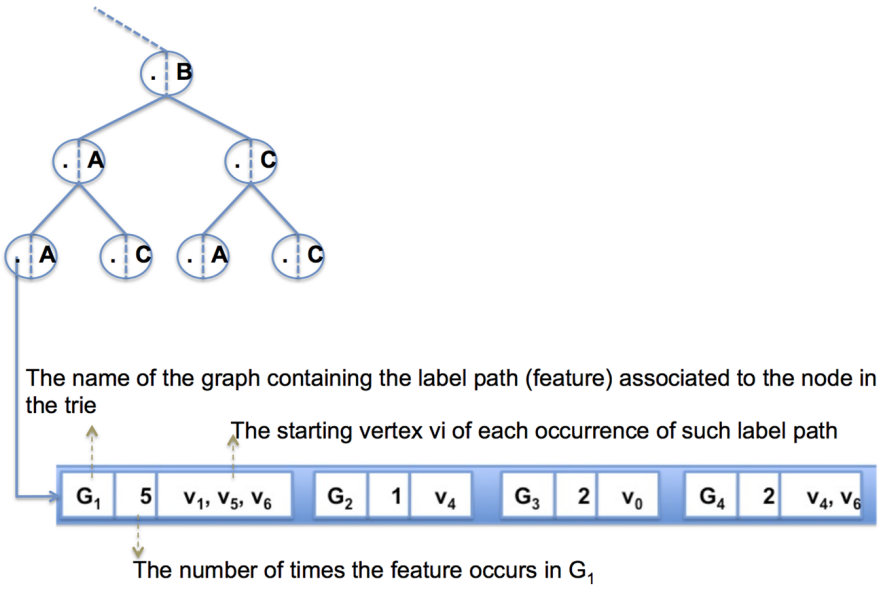


Figure 3.1: Each node in the GRAPES trie links to (i) the list of graphs g_i containing the feature associated with that node, (ii) the number of times the feature occurs in each G_i, and (iii) the starting vertices v_i of each such path. Adapted from [87].

same length were extracted, the information would reside only on the leaves of the trie. By considering paths of variable length up to a maximum length l_p, the information also resides on intermediate nodes of the trie.

Filtering phase

During querying phase, labelled paths are extracted from the query. Similarly to the extraction of paths from target graphs, for each path the number of times it occurs in the query graph is retrieved. Initially, all the target graphs are candidates to contain the query graph. Query paths are searched in the index in order to recognize the target graphs that contain the same paths of the query. For each path, the number of occurrences within the target graph must be equal or exceed the number of its occurrences in the query graph. By using the starting nodes of the paths stored in the index, the initial structures of target graphs are skimmed in order to extract only the vertices that are the starting point of paths in the query graph. Thus, the filtering procedure produces two different results, a list of graphs that may contain the query (since each selected graph contains the same labelled path of the query with

the same amount), and for each selected graph the list of vertices that are candidate to match with the query vertices.

Verification phase

The verification phase is performed with the VF2 algorithm [78] which solves the subgraph isomorphism problem. The problem of searching a query graph within a target graph consists in finding a mapping between the vertices of the query and target graphs such that constraints are satisfied. Constraints regard the compatibility of labels assigned to the vertices and the existence of the query edges between the corresponding query-target mapped vertices.

Chapter 4

Decision diagrams

Decision diagrams (DDs) are a broad family of graph-based data structures for the efficient encoding and manipulation of functions defined on discrete structured domains, namely defined as the cross-product of finite sets.

Initially, they were proposed for industrial hardware verification due to their ability to encode complex Boolean functions on very large domains. Then, they were successfully applied in different research fields ranging from network reliability analysis [107] to performance evaluation of stochastic systems [108]. A core problem in model checking is that space and time requirements increase exponentially with the models' size. One method to alleviate this problem is symbolic model checking, where sets of states are stored in symbolic data structures, as the decision diagrams. In these contexts, they have proven to be effective tools (i) to encode compactly structured sets by exploiting their structure and regularity, and (ii) to manipulate entire sets of elements at once, instead of exploring every single element explicitly.

In this chapter, we formally introduce the family of decision diagram data structures and their most common classes, together with the supported operators to their manipulation. Finally, some implementation details and a decision diagram library are presented.

4.1 General definition

Generally speaking, decision diagrams are rooted directed acyclic edge-labeled graphs representing functions from a set of L variables $x_L \ldots, x_1$ to generic values Y.

Variables x_i are defined on the domains $\mathcal{X}_i$; then the domain of the function represented by the Decision Diagram (DD) is the cross-product of all the domains X_i. In the following, there are considered ordered decision diagrams, namely a total order is defined on its variables (i.e., $x_l \succ x_k \Leftrightarrow l > k$) such that every path through the DD visits nodes according to this ordering. The nodes of a DD are organized in a level-wise fashion and are either terminal or non-terminal. Let p be a generic node and let $p.lvl \geq 0$ be the level of node p within the DD.

- A non-terminal node m at level $1 \leq k \leq L$ is labeled with a variable $var(m) \in \{x_1, ..x_K\}$ and has exactly $N_{var(m)} = |\mathbb{N}_{var(m)}|$ outgoing edges pointing to its children nodes. We refer to the i-th child of node m as $child(m, i)$, with $0 \leq i < N_{var(m)}$.

- A terminal node m such that $p.lvl = 0$ has no children and is labeled with a value $val(m)$, which generally corresponds to one of the return values of the function.

Moreover, non-terminal nodes represent cofactors of the function being represented by the whole decision diagram. Let us consider a generic decision diagram D defined over L variables $(x_L, \ldots, x_1)$, let $D_{x_i=v}$ (where $v \in X_i$) as the decision diagram over variables $(x_L, \ldots, x_{i+1}, x_{i-1}, \ldots, x_1)$ representing the function $f_D(x_L, \ldots, x_{i+1}, v, x_{i-1}, \ldots, x_1)$. Then, given a constant vector $i = (i_L, \ldots, i_1) \in X$, we can evaluate the function encoded by the DD in linear time by traversing it starting from the root to a terminal node.

An example of DD is reported in Figure 4.1(a). It is defined on four variables, $x_4, \ldots, x_1$ and encodes the function counting the occurrences of an element into a multiset [1] S where each element is described by a tuple (x_4, x_3, x_2, x_1) with $x_4 \in \{0, 1, 2, 3\}$, $x_3 \in \{0, 1, 2, 3\}$, $x_2 \in \{0, 1, 2\}$, and $x_1 \in \{0, 1\}$. Thus, the DD path from the root assuming $x_4 = 2, x_3 = 3, x_2 = 0, x_1 = 1$ and leading to terminal node 3 means that the element $(2, 3, 0, 1)$

[1] The multiset (or bag) extends the concept of a set allowing for multiple instances for each of its elements.

has three occurrences in the multiset S. Let EVAL(dd, x) be the function that given a decision diagram dd and a variable assignment x, returns the value of the terminal node linked corresponding to variable assignment x in dd.

4.1.1 Notion of ordering

From this point on, there is the assumption of *ordered decision diagrams*: there exists a bijective mapping between the variables $x_L \ldots, x_1$ and the DD levels, so that variable x_i is assigned univocally to level $k \geq 1$. It is important to notice that the choice of the ordering for the variables of the DD can strongly affect its size, i.e. its number of nodes and edges.

The high storage efficiency of DDs is strongly conditioned by the choice of a "reasonably good" *variable order*, i.e. the assignment of the problem variables to the DD levels. Finding the optimal variable ordering is known to be computationally expensive [109], and as a consequence, the efficiency of approaches based on DDs is largely reliant on the development of domain-specific heuristics to identify an appropriate ordering. Other techniques known as *dynamic variable reordering* can be used to modify the ordering of an existing decision diagram in an attempt to reduce its size. In literature, some heuristics exist to help search at least sub-optimal orders, but these algorithms typically use problem-specific information and it is hard to apply on different application domains [110, 111, 112, 113].

4.1.2 Reduction rules

One of the reasons that allow decision diagrams to often provide compact storage is that they are stored in a reduced form.

Firstly, if nodes m and m' are identical, namely if they are labeled with the same variable and their children are identical (i.e. $var(m_1) = var(m_2) \wedge child(m_1, i) = child(m_2, i)$ for all $i \in \mathbb{N}_{var(m_1)}$), only one copy of the node is stored. We refer to this as the sharing of nodes. Secondly, if the children of node m are all identical, namely if $child(m, i) = child(m, j) \, \forall \, i, j \in \mathbb{N}_{var(m)}$, node m is said to be redundant because the value of the function does not depend on the value of the variable represented by m. Then, such node is removed and incoming edges on it are redirected to its unique child. We refer to this as a skipped level.

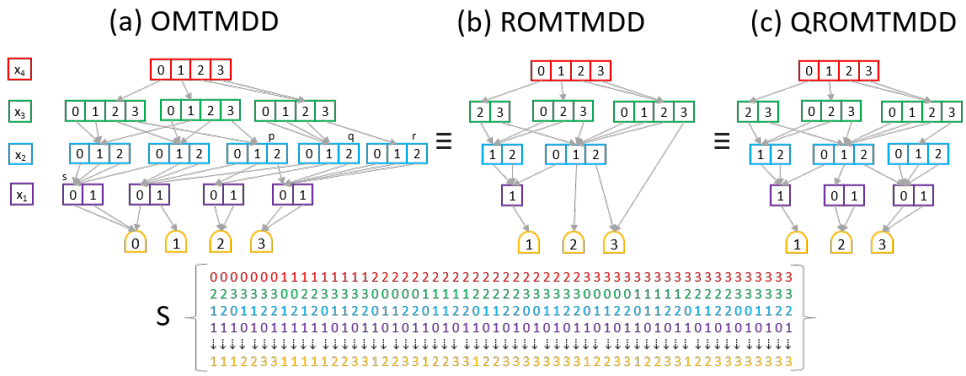


Figure 4.1: Different representation of an MTMDD encoding the function counting the occurrences of an element into the multiset S: (a) an ordered MTMDD; (b) a reduced ordered MTMDD; (c) a quasi-reduced ordered MTMDD

Two popular reduction schemes are fully reduced and quasi-reduced decision diagrams A decision diagram is said to be fully reduced if it contains neither redundant nor duplicated nodes.

A common variation of the above reduction rule is to allow redundant nodes but no duplicate nodes. An ODD is called quasi-reduced (QRDD) if it contains no duplicate nodes and if all paths from the root node to a terminal node visit exactly one node for each variable.

In Figure 4.1 an MTMDD is depicted in its complete (a), fully reduced (b) and (c) quasi-reduced representations. For the sake of clarity, decision diagrams are common to explicitly represent only those paths ending to the terminal nodes different from zero. Because of this, the node s and the edges on and from it, which are represented in Figure 4.1(a), can be omitted.

In Figure 4.1(a) the nodes p and q, colored in cyan, are duplicate nodes, while r and s are examples of redundant nodes. The corresponding reduced version (i.e. without redundant and duplicated nodes) is instead reported in Figure 4.1(b). In Figure 4.1(c) the quasi-reduced (i.e. without duplicated nodes) version of the OMTMDD in Figure 4.1(a) is depicted.

4.2 Classes of decision diagrams

Various classes of decision diagrams have been defined in the literature. In the following, there are presented some of the most common ones and the

operations supported by them.

4.2.1 Binary Decision Diagrams

The Boolean Decision Diagram (BDD) represents a L-variable boolean function of the form $f : \mathbb{B}^L \to \mathbb{B}$, for some finite $L \in \mathbb{N}$.

A non-terminal node at level $L \geq k \geq 1$ corresponds to a choice for the value of the variable x_k, and has two outgoing edges that are labeled with 0 and 1. Such edges point to $child(p, 0)$ and $child(p, 1)$ respectively, which are both at levels below k. Terminal nodes correspond to constant functions returning 0 and 1, respectively.

BDDs are manipulated using logical operators, such as the NOT operator, which returns the complementary BDD, and the AND and OR operators, which return the conjunction and disjunction of two BDDs, respectively.

A BDD that is ordered and reduced provides a canonical representation of the function. Given a boolean function and a variable ordering, there exists exactly one ROBDD representing that function.

4.2.2 Multi-valued decision diagrams

The Multi-way Decision Diagram (MDD) extend the BDD by allowing non-terminal nodes to represent variables defined on integer domains. MDDs encode functions of the form $f : X \to \mathbb{B}$, where the domain X is the cross-product $X = X_L \times \cdots \times X_1$ of L finite sets such that each $X_k = \{0, 1, \ldots n_k - 1\}$, for some $n_k \in \mathbb{N}$.

Then, non-terminal nodes at level k correspond to a multi-way choice for the argument variable x_k, namely a cofactor $f_{x_k=c}$, for some $c \in X_k$.

MDDs are widely used to encode sets of elements, and they can be manipulated using set operators such as union, intersection and difference.

A reduced ordered MDD (ROMDD) is a canonical representation: given any integer function and a variable ordering, there exists exactly one ROMDD representing that function.

4.2.3 Multi-terminal decision diagrams

The Multi-Terminal Boolean Decision Diagram (MTBDD) extends the BDD by allowing arbitrary terminal nodes in order to represent functions of the form $f : \mathbb{B}^L \to \mathbb{R}$. The Multi-Terminal Multi-way Decision Diagram (MTMDD)

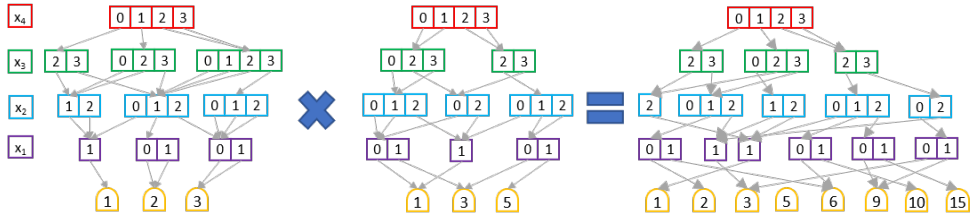


Figure 4.2

naturally merge MTBDD and MDD to encode either integer or real functions of the form $f : X \to \mathbb{R}$. Reduced and quasi-reduced ordered MTMDDs are canonical representations of the function.

Multi-terminal versions of decision diagrams can be manipulated using both arithmetic and relational operators. For instance, the application of the multiplication operator between two MTMDDs d_1 and d_2 is depicted in Figure 4.2. The result of the multiplication is encoded in the terminal nodes. Given a variable assignment $x = \langle x_4, x_3, x_2, x_1 \rangle$, the terminal value corresponding to this variable assignment is given by $\text{EVAL}(dd_1, x) \times \text{EVAL}(dd_2, x)$. It follows that the resulting MTMDD will contain all the variable assignments that correspond to non-zero terminals in both the input MTMDDs.

For instance, let us consider the two following variable assignments: $x_1 = \langle 1, 3, 2, 0 \rangle$ and $x_2 = \langle 1, 3, 2, 1 \rangle$. In dd_1 they both correspond to terminal three, whereas dd_2 has a non-zero terminal only for x_2, whose value is equal to one. As shown in the Figure, the product MTMDD reports the variable assignment x_1 which is linked to terminal $1 \times 3 = 3$, whereas the assignment x_2 is not reported since it is not encoded in dd_2.

4.2.4 Edge-valued decision diagrams

Edge-valued decision diagrams can encode functions with a non-boolean range as multi-terminal decision diagrams, but in such a way that the function's return value is distributed over the edges found along a path, instead of being encoded in terminal nodes. This brings to more complex manipulation algorithms with respect to multi-terminal versions and exponentially more compact representations than their respective multi-terminal versions.

4.3 Decision diagram implementation

In the literature different software libraries implementing decision diagrams were proposed, such as CUDD [114], LibDDD [115], Meddly [116], and Sylvan [117].

All of experimental work was implemented using the Meddly (Multi-way and Edge-valued Decision Diagram LibrarY) package, which natively supports all the decision diagram classes presented so far. In particular, such a library has been chosen because it offers two different user interfaces: (i) a *simple interface* which provides the basic operators to easily create and manipulate DDs, and (ii) an *expert interface* which allows the user to customize the existent operators and/or to define new ones.

A named collection of nodes of a particular variety of DD that are associated with the same domain, is called a forest, namely a very large decision diagram with multiple root nodes. All forms of DDs in Meddly require ordering of variables an ordered collection of $k > 0$ variables with specified sizes is called a domain $\mathcal{D} \in \mathbb{N}^k$. Let $\mathcal{D} = (\mathcal{N}_k, \ldots, \mathcal{N}_1)$ be a generic domain on k variables, where $\mathcal{N}_i \in \mathbb{N}$ is the size of variable x_i, which can assume a value in $[0, \mathcal{N}_i)$ interval.

Within a given forest, Meddly automatically eliminates duplicate nodes using a unique table, imposes forest-specific reduction rules, and handles memory management of the nodes (storing them compactly, garbage collection, etc.) [118]. This means that nodes are not even duplicated across different decision diagrams. Whenever a new node needs to be created, it is first verified whether or not such a node already exists in the forest and, if so, reused.

The operations relevant to this thesis are provided by the simple interface, which are described below:

- createEdge creates a new DD in the given forest by explicitly stating a set of variable assignments and the corresponding return values. For example, let F be a forest defined on the domain $X = X_3 \times X_2 \times X_1$ and $f : X \to Y$ be a function represented by a decision diagram. Given the variable assignments $\mathbf{x_1} = (a_1, b_1, c_1)$, $\mathbf{x_2} = (a_2, b_2, c_2)$, $\mathbf{x_3} = (a_3, b_3, c_3)$, and the corresponding return values $y = (y_1, y_2, y_3)$, the call to createEdge$(F, (\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}), (y_1, y_2, y_3))$ creates a new decision

diagram in the forest F representing the following function:

$$f(\mathbf{x}) = \begin{cases} y_1 & \text{if } \mathbf{x} = \mathbf{x_1} \\ y_2 & \text{if } \mathbf{x} = \mathbf{x_2} \\ y_3 & \text{if } \mathbf{x} = \mathbf{x_3} \\ 0 & \text{otherwise} \end{cases}$$

- EVALUATE determines the value of the function represented by the DD for a given assignment of its variables. Then, the call EVALUATE$(dd, x_1, \ldots, x_k)$ returns the terminal value linked to the path $x_1, \ldots, x_k$ of the decision diagram dd.

- APPLY is used to manipulate DD applying on it a specific DD operator. Meddly supports both unary and binary operators and imposes that operands of binary operators must have the same domain, but they can live in different forests.

Meddly automatically uses and maintains a *computed table* to reduce the computational cost of the APPLY operations. This is essentially a cache for storing the results of operations on DDs. Before any operation is performed, the cache is checked to see if the same operation has already been performed. If so, the results can simply be reused. If not, the result is computed, stored in the cache and then returned.

Part II

Theoretical results

Chapter 5

Graph-based feature selection for multi-omics integration

In this chapter, we propose a general multivariate filter-based feature selection algorithm based on the minimal-Redundancy Maximum-Relevance (mRMR) criteria that is defined via an optimization problem defined over a graph representation built from a tabular training set.

5.1 Feature graph formalisation

In the following, a knowledge representation in the form of a graph is derived from a labeled dataset $\mathcal{D} = (\mathcal{X}, \mathcal{Y}, \mathcal{M})$ built with respect to a generic supervised task $\mathcal{T}$, either classification or regression. With the term *feature graph* we refer to a graph whose vertices represent features, and edges between pairs of vertices (i.e. features) encode feature interactions and the possible existence of some relationship or dependency between the corresponding pair of features.

Both the vertices and edges of the feature graph are weighted. A vertex weight encodes the relevance of the corresponding feature in the task $\mathcal{T}$. An edge weight encodes the magnitude of redundancy between a pair of features, which indicates the degree of replacement of a feature with the other, and vice

versa.

It is challenging to identify which specific statistical measures are more convenient to use in the general case, and this topic is beyond the scope of such a thesis. The main challenge is related to the possible heterogeneity of feature types (e.g. quantitative, categorical, etc.) within a dataset, specifically when multiple omics are considered once (e.g. early integration). However, one can distinguish between the vertex score function and the edge weight function. The vertex score function involves the task $\mathcal{T}$, which is either a categorical or quantitative quantity in classification and regression tasks, respectively.

5.1.1 Definition

Let us introduce two general functions for estimating the weight sets of the graph.

1. the *feature relevance function* $f_{wv} : F \to \mathbb{R}$ estimates vertex weights by computing a relevance score between the values of each feature and the target values Y;

2. the *interaction redundancy function* $f_{we} : F \times F \to \mathbb{R}$ estimates the edge weights by computing a redundancy score between the values of each pair of features.

Let us denote as *feature graph* the complete undirected labeled weighted graph $G = (V, E, L, w_v, w_e)$ such that (i) $|V| = |F|$ and feature f_i is represented by vertex v_i that is weighted exploiting the feature relevance function as $w_{v_i} = f_{wv}(f_i)$, and labeled with metadata associated with feature f_i, whereas (ii) the edge (v_i, v_j) is weighted exploiting the interaction redundancy function, as as $w_{e_{i,j}} = f_{we}(f_i, f_j)$.

Given a feature graph $G = (V, E, \dots)$ and a feature set $S = \{f_i\}$, let us define the *feature subgraph* $G_S = (V', E', \dots)$ that is induced from G by selecting only the vertices appearing in S, and all the edges spawning between them.

5.2 Optimization problem for mRMR on graph

The graph formalism described above allows us to formulate an optimization problem for discovering feature sets that satisfy the mRMR criterion.

In particular, a value $0 < t < 1$ is chosen as the threshold for the values of pairwise feature correlations. A t-thresholded feature graph is obtained from the complete feature graph to define the criteria for considering whether two features are redundant. Then, the goal of feature selection is the identification of a relatively small amount of non-redundant features that are highly predictive with respect to the target.

5.2.1 A metric for mRMR over the feature graph

Let us define a metric for evaluating feature sets founded on the mRMR criteria and based on the structure of the feature graph. The optimization problem consists of searching for disconnected feature subgraphs $g^i = (V^i, E^i, w_v^i, w_e^i)$ such that (i) the sum of their weight is maximized and (ii) the number of the edges $|E^i|$ is minimized. We define the Penalised Vertex Score (PVS) metric for feature set S as:

$$PVS(S_F) = \sum_{v_i \in G_{F_S}} \frac{w_v(v_i)}{1 + deg(v_i)} \tag{5.1}$$

Given a feature set $F = \{f_1, f_2, \ldots, f_n\}$, the feature f_j, $j = 1, \ldots, n$ is (i) scored as its weight $w_v^i(f_j)$ and (ii) penalized proportionally to the degree of vertex v_i, namely to the number of features within F that are redundant to v_i. The problem consists of the identification of solutions maximising the PVS metric

Feature selection is an NP-hard problem. In particular, it is a combinatorial optimization problem, whose problem space has a huge number of configurations and is rich in minima and maxima, namely bad and good feature sets. For N features, there exist 2^N candidate solutions represented by as many feature subgraphs. By defining an upper bound $N_{max} < N$ in the number of selectable features, the problem space has $\sum_{i=1}^{N_{max}} \binom{N}{i}$ candidate solutions.

There is a vast plethora of algorithms for combinatorial optimization. In particular, we pay attention to metaheuristics strategies, which provide general algorithmic frameworks which can be exploited to solve a wide set of different problems with relatively small modifications to make them adapted to a specific problem. They can be broadly categorized as single-solution and population-based. Single-solution algorithms, such as simulated annealing and tabu search, keep only one candidate solution at a time, which is iter-

atively refined by exploring the neighbourhood of the solution itself and exploiting strategies to escape local optima. On the contrary, population-based approaches, such as Evolutionary Algorithms (EAs) and Swarm Intelligence methods, maintain a population of solutions to the problem, which are refined by interacting with each other for multiple iterations. This process eventually leads to the population converging on a good solution to the problem.

Overall, metaheuristics algorithms can be defined as high-level strategies to guide the search process, and their generality is founded on the distinction between the problem space $\mathbb{X}$, also known as *phenome*, which depends on the specific problem, and the search space $\mathbb{G}$, also known as *genome*, which is explored by means of a set of metaheuristics-specific search operators $searchOp : \mathbb{G}^n \to \mathbb{G}^m$, which takes a fixed number $n \in N_0$ of genotypes $g_1, \ldots, g_n$ and returns $m > 0$ elements from the search space. Usually, the problem and the search space differ and a genotype-phenotype mapping $gpm : \mathbb{G} \to \mathbb{X}$ is required to evaluate the fitness of candidate solutions.

5.2.2 Evolutionary and genetic algorithms

In the following, an approach based on a genetic algorithm (GA) is proposed, although other population-based methods should be explored. GAs belong to the family of Evolutionary Algorithms (EAs), whose basic algorithm is represented and described in Figure 5.1: they are based on mechanisms inspired by Darwin's theory of evolution (e.g. natural selection, survival of the fittest, reproduction) to iteratively refine a set of candidate solutions [119].

The biological world also highly inspires the terminology used in EAs. In particular, GAs are a family of EAs whose elements in the search space $g \in \mathbb{G}$ are known as *chromosomes*, which are sequences of primitive elements (i.e. bits, natural numbers) called *genes*. The position of the gene within the chromosome is called a *locus*, and the possible values that a gene can assume are called *alleles*.

The search operations exploited by GAs are the crossover and mutation operators, which represent sexual and asexual reproduction, respectively. The crossover operation, also known as recombination, combines two parent chromosomes in new genotypes that inherit traits from both elders. The mutation operator applies small random changes within individual chromosomes: in biology, changes in the DNA provide the genetic diversity among individuals so

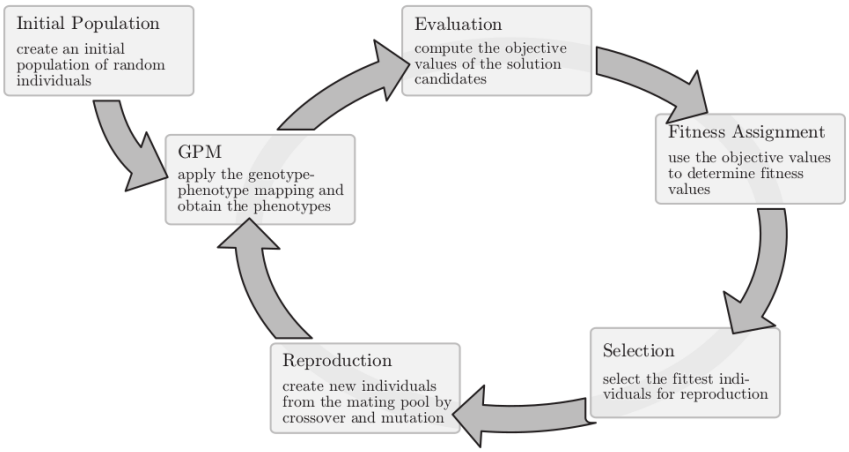


Figure 5.1: Genetic algorithm in a nutshell: (i) the initial population is created; (ii) the genotype-phenotype mapping is applied to the population; (iii) phenotypes are evaluated through the objective function, which values are used to (iv) assign fitness to every $x \in X$; (v) the fittest individuals are selected for (vi) reproduction, which consists of the combination of crossover and mutation probabilistic operators in order to create new evolving individuals inheriting traits from the parents. (vii) The algorithm will stop itself when a certain termination criterion is satisfied. Image from[119].

that natural selection can act on them. Without mutations, evolution would not be possible, both in nature and in GAs.

5.3 Feature selection via genetic algorithm

In this section, a multivariate filter feature selection approach based on the feature graph is presented. The optimization problem based on the mRMR criterion is faced with a genetic algorithm that exploits the topology of the feature graph for both fitness evaluation and mutation operator.

5.3.1 Chromosome encoding

In the following, we introduce two possible chromosome encoding using either boolean or integer genes, and the corresponding genotype-phenotype mapping functions, which allow us to map the chromosomes $g \in \mathbb{G}$ in the corresponding feature subgraph $x \in \mathbb{X}$ that is induced from the whole feature graph by considering only the features encoded in the genotype g.

Let us consider a generic feature graph $G = (V, E, \dots)$ representing relationships of m features, then $m = |V| = |F|$. As a toy example, let us consider $m = 11$ features.

Boolean genotypes

The simplest case is provided by the boolean genotype, which consists of chromosomes in the form of $x \in \mathbb{B}^m$ with a number of genes equal to the number of available features m, which is often a quite large number. Let us consider a chromosome $c = (c_1, \dots, c_m)$ The gene c_i at locus i have two possible alleles:

$$a_i = \begin{cases} 1, & \text{then } f_i \text{ is included} \\ 0, & \text{then } f_i \text{ is excluded} \end{cases}$$

Therefore, the set of features encoded by a generic boolean chromosome is given by the locus having the allele 1.

In the example below, the feature set $\{2, 3, 6, 7, 10\}$ is represented.

0	1	1	0	0	1	1	0	0	1	0
1	2	3	4	5	6	7	8	9	10	11

(5.2)

Integer genotypes

Integer genotypes allow us to define an upper bound on the number of selectable features. They have the form of m-length integer vectors ($0 < k < m$), namely $x \in \mathbb{N}^k$. Let us consider the locus i, $1 \leq i \leq k$ that corresponds to the i-th gene. Such gene has $m + 1$ alleles: $0, 1, \dots, m$. The null allele $a_i = 0$ indicates an intron, namely a non-coding gene that represents no feature at all. Otherwise, the allele $a_i = v$, with $1 \leq v \leq m$ indicates that feature f_v is encoded by the i-th gene; multiple genes may encode the same features, but they are countered only once.

10	2	3	7	0	6
1	2	3	4	5	6

Genotype-Phenotype Mapping (GPM)

The GPM function allows us to map a generic chromosome c to the corresponding feature subgraph, which is induced from the whole feature graph G by considering only the features (i.e. vertices) included in the chromosome c, together with all the edges spawning between such vertices in G.

5.3.2 Selection mechanism

Fitness assignment

In preparation for the fitness assignment to chromosomes, which enables the selection operator, the genotypes of the current population are translated to their corresponding phenotypes via the *gpm* function, obtaining the corresponding feature subgraphs of such chromosomes.The fitness of such phenotypes is computed by means of the PVS metric, defined in Equation 5.1. Then, the value of the objective function is obtained by normalizing the PVS value using a normalization factor a.

Tournament selection

Once fitness values have been computed, the selection operator chooses which chromosomes of the population pass to the reproduction step by forming the mating pool. The selection operator chooses which chromosomes of the population pass to the reproduction step, in which the probabilistic operators of crossover and mutation are applied. The k-tournament selection is applied n times. At each application, it samples k chromosomes from the population, which are compared against each other in a tournament. The winner, namely the chromosome associated with the highest fitness value, enters the mating pool.

Elitism mechanism

The elitism mechanism ensures that at least one copy of the best individual(s) of the current generation is propagated onto the next generation, without any application of reproduction operators.

5.3.3 Reproduction mechanisms

The reproduction operators, namely crossover and mutation, play a crucial role in balancing the exploration and exploitation trade-off [120], which is a fundamental challenge in optimization problems. Exploration means finding new points in areas of the search space which have not been investigated before, whereas exploitation is the process of improving and combining the traits of the currently known solutions. By carefully selecting and adjusting the parameters of these operators, the algorithm can effectively search for better solutions without getting stuck in local optima.

In the context of EAs, the crossover operator is considered to be an operator for exploitation, whereas the mutation operator allows us to explore novel and hopefully better solutions through exploration.

Crossover operation

The crossover operator, also known as recombination, represents sexual reproduction. Pair of chromosomes are sampled from the mating pool and are crossover operator is applied to them with probability p_c. Given two parent chromosomes, one or more offspring chromosomes whose phenotypes inherit from parent chromosomes are built. There is an incredibly wide variety of crossover operators for GAs, such as Single-Point Crossover (SPX), Two-Point Crossover (TPX), Multi-Point Crossover (k-PX) and Uniform Crossover (UX).

Mutation operator

The mutation operator applies small random changes with probability p_m to the single chromosome and represents asexual reproduction. In biology, changes in the DNA provide the genetic diversity among individuals, so that natural selection can act on. Without mutations, evolution would not be possible, both in nature and in EAs. At least three different mutations can be identified in nature:

- insertions: one or more nucleotides are added to the DNA sequence;
- deletions: one or more nucleotides are removed from the DNA sequence;
- pointwise mutation: any change in the sequence rather than insertion and deletion.

In EAs, insertions and deletions consist of switching on an intron (i.e. non-coding gene) with a feature and turning off a gene, respectively. Finally, we introduce a point-wise mutation that consists of swapping a feature with another one in its neighbourhood in the feature graph.

Similarly to the crossover, the mutation operator can be defined by different behaviours: (i) single-point mutations randomly modified the allele of a gene, (ii) multi-point mutation randomly modified the alleles of $0 < n \leq len(g)$ genes in the genotype g.

Chapter 6

Decision diagrams applied to graph indexing

In this chapter, we introduce a solution based on DDs to cope with indexing a collection of target graphs and the subgraph isomorphism problem by extending the index-driven approach provided by GRAPES. In particular, we show how DDs can be exploited to efficiently encode such a problem and how they can be manipulated to develop a systematic filter-and-verification stage.

6.1 Decision diagrams for indexing

GRAPES showed the best performance in terms of running time. However, its index requires a relatively high memory amount compared to the other approaches. GRAPES is implemented both as sequential and parallel software using symmetric multiprocessing (SMP) architectures. In addition, GRAPES was developed for achieving good performance in the collection of graphs as well as in scanning a query over a single large target graph. For these reasons, we decided to improve the performance of the sequential version of GRAPES by reducing the memory required for its index. We investigated the use of decision diagrams for reaching such a goal without degrading the running time of the algorithm.

GRAPES uses a trie (i.e. prefix tree) to store the indexed graphs, since it provides a compact representation of a set of strings by taking advantage of their common prefixes, considerably reducing the data redundancy. Nevertheless, the tree structure of a trie (i.e. only a single edge can point to a node) makes it hard to exploit other types of symmetries present in the indexed graphs, as for instance the sharing of the same (i) starting vertices and/or (ii) relative occurrence number, as well as common substrings which are not prefixes. To deal with these aspects, during my Ph.D., in [121] we proposed to encode the indexed graphs using decision diagrams in place of the GRAPES' trie. Since decision diagrams are graphs, the requirement for a tree structure is relaxed allowing multiple arcs to point to the same node. The main advantage of this is a potentially more compact representation due to the decision diagram's ability to better exploit the regular structure of the data (e.g., common substrings present in the indexed graph paths) thanks to the reduction rules and to manipulation algorithms.

As said before, decision diagrams are a family of data structures. Therefore, there are multiple ways to proceed, which depend both on (i) the kind of decision diagram is used and (ii) how data are encoded in the decision diagram structure.

Let l be the maximum feature length, and consider the graph collection $G = \{g_1, \ldots, g_n\}$ such that $V_i = \{V_{i,1}, \ldots, V_{i,m_i}\}$ is the set composed of m_i vertices belonging to graph g_i. Then, let us define $V = \{V_1, \ldots, V_n\}$ as the set of vertices of the whole graph collection, which is composed of a total of $N_{G_V} = \sum_{i=1}^{n} m_i$ different vertices.

The decision diagram had to encode all the labeled paths p up to length l starting from each vertex $v_i \in V$ belonging to the collection such that v_i belongs to graph g_j. For any path p, the following information has to be attached to it:

1. the location information (g_j, v_i) of p, namely from which vertex v_i belonging to which graph g_j it starts;

2. the total number of occurrences $n_p > 0$ of p in each g_j;

6.1.1 Problem variables

To define the graph indexing problem, a set of variables and the corresponding domains are identified.

- The maximum path length l defines as many *label variables*. The domain of label variables depending of how many labels appear in the graph collection. Given the label alphabet Σ, label variables have to represent $|\Sigma| + 1$ different values. The additional value serves to represent the absence of a label, in order to represent paths shorter than the maximum length l.

- The location information is a pair (g, v) composed of (i) the graph id variable, which can assume $|G|$ different values, and (ii) the vertex id variable, which can assume $\max_i m_i$ different values in order to properly represent all the vertices of the larger graph in the collection.

- The counters for path occurrences are natural numbers $n_c \in \mathbb{N}_+$. Counters are not apriori known, differently from the quantities listed so far, and they must be computed during index construction phase.

Once the graph database has been indexed in a decision diagram, given a query, we have to retrieve from the index the location information (g, v) about those graphs' vertices satisfying the query constraints that are defined on path occurrences. The query graph is used to build its decision diagram representation, using the same structure as the index. The query DD is exploited to manipulate the index using appropriate operators, that depends on the specific decision diagram in use, in order to get the desired location information.

6.2 Encoding the problem

The problem presented above can be encoded in different ways depending on the type of adopted decision diagram and the semantics assigned to its variables.

6.2.1 Using multi-terminal

Among multi-terminal decision diagrams, we distinguish between MTMDD and MTBDD, representing functions in the forms $f_M : \mathbb{N}_1 \times \cdots \times \mathbb{N}_n \to \mathbb{N}$ and $f_B : \mathbb{B}^n \to \mathbb{N}$, respectively. Since the encoding of MTMDD is more straightforward than MTBDD, they are presented first.

Multi-terminal multi-way decision diagrams

MTMDDs represent functions in the form $f : \mathbb{N}^n \to \mathbb{N}$, and allow us to represent the index in a straightforward manner. The index is a function $f_I : G \times V \times \Sigma^l \to \mathbb{N}_+$ that, given a graph id g, a vertex id v and a labeled path p, returns the number of occurrences of p in the graph g, if p starts from vertex v, zero otherwise.

An alternative approach, that has been implemented and is in-depth described in Chapter 8, is based on the unification of the graph id and vertex id variables in a single variable x_{loc} that has a pretty large domain in order to represent a total of $\sum_{i=1}^{n} m_i$ elements, corresponding to the number of vertices of the whole graph collection under consideration. Let us denote such an approach as *flattened location information*.

Multi-terminal binary decision diagrams

MTBDDs represent functions in the form $f : \mathbb{B}^n \to \mathbb{N}$. The MTMDD-based methods can be easily applied to MTBDD by applying a binary encoding on the integer variables of the MTMDD. That is, for each variable x_i of the original MTMDD, which is associated with the domain D_i, there are created $\lceil \log_2(D_i) \rceil$ boolean variables in the corresponding MTBDD.

For example, let us consider the flattened location information approach. Given the maximum feature length l, in the MTMDD version there are $l + 1$ variables: l label variables, plus the location information variable.

Label variables can represent $N_L = |\Sigma| + 1$ possible values. The location information variable enumerates all the vertices in the graph collection, for a total of $N_{G_V} = \sum_{i=1}^{n} m_i$ values.

Therefore, given the maximum path length l, the index MTBDD represents a function $f : \mathbb{B}^n \to \mathbb{N}$, where the number of variables n is calculated as follows.

$$n = l \cdot \lceil \log_2(N_L) \rceil + \lceil \log_2(N_{G_V}) \rceil \tag{6.1}$$

How to improve sharing among DDs

The use of a forest instead of a single decision diagram allows us to split the index across multiple decision diagrams, which internally may share memory thanks to the unique table.

For example, one could split the index in a graph-specific manner, namely to index each graph g_i in a separate MTMDD d_i. Then, the index I is represented by a forest composed of m decision diagrams $F_I = \{i_{g_0}, \ldots, i_{g_{m-1}}\}$. The domain of such forest is restricted to represent only the labeled paths up to length l, and the starting vertex of such paths.

An alternative forest-based solution is to build label-specific indexes, that is, to use a number of MTMDD equal to the number of labels in the graph database, and k-th MTMDD encodes all the paths starting with label σ_k. Let us consider a generic graph-vertex pair (g_i, v_j). Since vertex v_j is univocally mapped via the labeling function to the label l_k, all paths starting from v_j will fall in the k-th MTMDD. As a result, the information regarding the first path label is redundant: this allows us to shrink the domain of the location information variable (i.e. group vertices per label, counts group and pick the max) and to use $l - 1$ variables to encode the labelled path.

6.2.2 Using multi-way decision diagram

A different approach is to use MDDs, namely a decision diagram representing $f : X \to \mathbb{B}$ where $X = \{X_n, \ldots, X_1\}$, that allows representing a set of n-tuples. Since MDDs have only two terminal values, namely 0 and 1, the generic n-tuple have to comprehend a variable to represents the counters of path occurrences. It is worth noting that the domain of such a variable is not a priori known.

To represent the whole index as a BDD there is required to use a number of boolean variables sufficient to encode the counters of path occurrences. However, since such a number is not a prior known, it is required to have an already-built index.

Let us suppose the index is represented as an MTBDD I_{mt}, which is composed of k variables (the value of k is provided by Equation 6.1), and let M be the maximum value among the terminal nodes in I_{mt}.

A first approach is to binary encoding the terminal values of I_{mt} using up to $\log_2 M$ boolean variables so that the BDD indexing the graph collection requires $k + \log_2 M$ variables, such that the (i) first term includes the variables for the labeled paths and the location information, and (ii) the second term corresponds to the variables for encoding the path occurrences.

A second approach that could save some variables with respect to the previous one, is based of counting how many terminal nodes exist in I_{mt}, let

say there are N_t different values, such that $N_t \leq M$. The N_t terminals can be mapped in the interval $[1, M']$ so that only $\log_2 M'$ boolean variables are needed to store the number of occurrences in the encoded form. Such encoding must be carefully taken into account during the filtering phase, in the sense that the occurrence counter must be decoded before being compared against the occurrence numbers in the query.

6.3 Index construction

From this point on, we will consider using the MTMDD approach with the flattened location information (i.e. $l + 1$ variables considering paths up to length l). However, in principle, what is written in the following is easily adaptable also using the other approaches previously described with relatively small efforts.

Given the maximum path length l and a graph database $D = \{g_1, \ldots, g_n\}$, where each $g_i = (V_i, E_i, L_i)$ is composed of m_i vertices $V_i = \{v_{i,1}, \ldots, v_{i,m_i}\}$, let us define two mappings.

1. Graph-vertex mapping $f : G \times V \rightarrow N$, which maps graph vertices in the range $0, \ldots, m-1$ where $m = \sum_{i=1}^{n} m_i$ is the total number of vertices within the database D,
2. Label mapping $f : V \times \Sigma \rightarrow N$, which maps labels in the range $0, \ldots, |\Sigma|$.

Let us define the index MTMDD I_{dd} an MTMDD composed of $l + 1$ variables: the location information variable x_{liv} and l label variables $x_{vl_1}, \ldots, x_{vl_l}$, such that $x_{vl_i}, 0 < i \leq l$ is the i-th path label. Let us define the order of variables, that is:

$$x_{liv} \succ x_{vl_1} \succ \cdots \succ x_{vl_l}$$

meaning that x_{liv} stays in the root of the index DD, then the labeled path is represented in the order it has been visited during DFS. On the terminal level, there are stored the total number of occurrences of each path in the graphs, that have to be computed.

Counting paths

The index construction exploits a limited DFS to exhaustively enumerate the labeled paths up to length l_p from each vertex of each graph in the database.

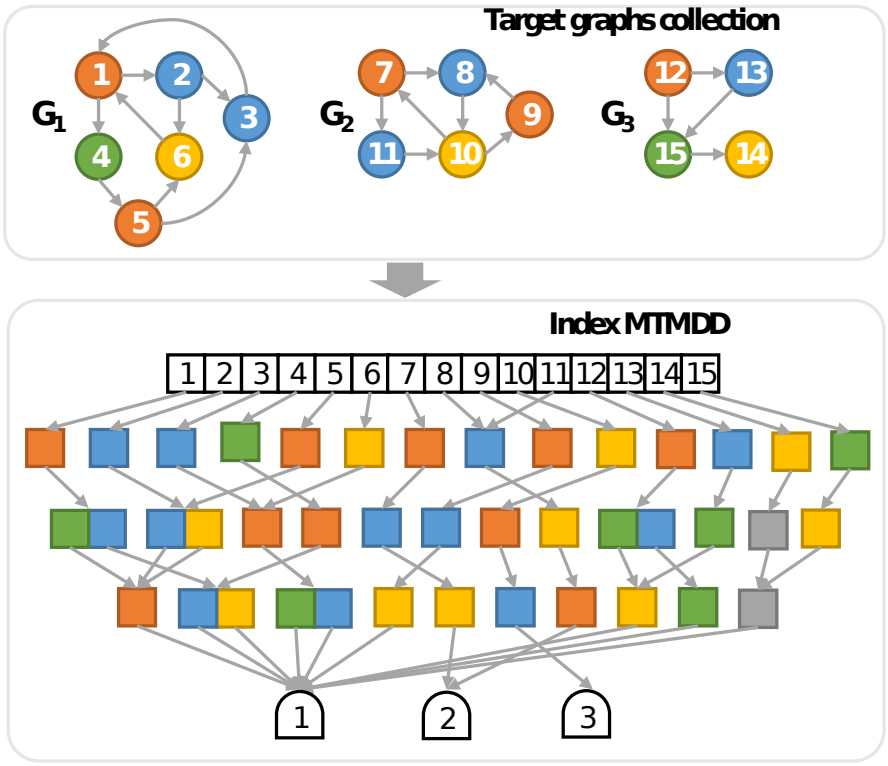


Figure 6.1

The GRAPES' approach for path counting is based on grouping vertices of the same graph with respect to their labels and uses temporary tries to accumulate counters of labelled paths. In the following, is described an equivalent approach based only on decision diagrams.

Let us consider a generic graph G composed of n vertices, $V = \{v_1, \ldots, v_n\}$. A limited DFS is performed starting from v_i in order to extract all the features up to length l_p starting it. Finally, for each feature p, the total number of occurrences in G has to be computed, and it has to be recorded together with the information about the starting vertex. Two temporary decision diagrams, namely an MDD M_1 and an MTMDD M_2, can be exploited as accumulators to eventually build the index MTMDD:

1. M_1 is used to store $l+1$-uples $x = \langle v_i, p_1, \ldots, p_l \rangle$ composed of the feature of length l and the id of the starting vertex;

2. M_2 is used to store l-uples $x = \langle p_1, \ldots, p_l \rangle$, namely the features of length l with no location information at all.

Once the graph G has been visited through limited DFS, M_1 will contain for each feature p in G, the set of vertices of G from which p starts. Similarly, M_2 will contain all the features within G and the total number of occurrences. At this point, is sufficient to merge the information contained within the M_1 and M_2 in order to build the index MTMDD.

For each feature p starting from vertex v contained in M_1, retrieve the total number of occurrences n_o of p in G from M_2 by calling $\text{EVAL}(M_2, p)$, so that the final tuple to be stored in the index is:

$$x = (v, p_1, \ldots, p_l), y = c$$

Graphical example

Figure 6.1 depicts a graph collection formed of three directed labelled graphs, $\{G_1, G_2, G_3\}$, which are composed of six, five, and four vertices, respectively, for a total of 15 vertices and four labels. For sake of clarity, the labels are represented by colors: orange, blue, green and yellow.

Such graph collection has been indexed in an MTMDD by using $l_p = 3$ as the maximum feature length. Then, the index MTMDD is composed of four variables:

- the root node represents the location information variable, whose domain ranges in $\{1, \ldots, 15\}$ to encode all the possible vertices in the graph collection;
- the second, third and fourth levels of the MTMDD are assigned to label variables devoted to representing the first, second and third label of a feature, respectively. In this case, the domain of label variables comprehends five different values (depicted as the grey value), in order to allow the representation of features shorter than three.

Finally, the terminal nodes represented the total path occurrence within the indexed graphs, which in this case range between one and three occurrences.

The reader can see an example of features shorter than l_p for the ones extracted from v_{14} and v_{15} belonging to G_3, which are labeled as yellow and green. In fact, v_14 is a dead end and the only feature extractable from it is the label of v_{14} itself, which is the yellow one. Then, the tuple that is stored in the MTMDD is composed as $\langle 14, yellow, grey, grey \rangle$.

Similarly, there is only one feature that can be extracted starting from v_{15}, namely the path comprehending v_{15} followed by v_{14}, which is a dead end. This corresponds to the tuple $\langle 15, green, yellow, grey \rangle$

6.4 Index filtering

Once the MTMDD index has been built from the graph collection G, it can be exploited to perform the filtering stage given a query graph G_q.

Filtering consists of restricting the set of target graphs to those subgraphs (i.e. connected components) potentially containing the query G_q, in order to reduce the number of vertices on which to apply the subgraph isomorphism algorithm. This consists of the identification of all those subgraphs (i.e. sets of vertices) of the indexed graphs which contain all the features within the query at least as many times the query.

As said in 4.2.3, MTMDDs can be manipulated using both arithmetic and relational operators. Such operators allow the implementation of a filtering strategy for the index given a query graph. Figure 6.2 showed the whole filtering process, which is explained as follows.

6.4.1 Indexing the query graph

Firstly, the query graph is indexed in an MTMDD in a similar way as the index. The query MTMDD contains the information on (i) which features appear in the query graph, (ii) from which vertices they start, as well as (iii) how many times each feature appears in the query graph.

It is worth noting that the location information of the query MTMDD, namely the starting vertices v_i^q from which vertex of G_q the query features start, are completely uncorrelated with the location information belonging to the index MTMDD. This makes it difficult to directly exploit the query MTMDD to solve the query. However, it is a piece of fundamental information to match the indexed vertices against the query vertices.

A second decision diagram, the *template query MTMDD* is built from the query MTMDD. It consists of storing all the query features and the relative occurrence numbers, ignoring the location information (i.e. the starting vertex). In place of it, a `DONT_CARE` node is exploited to represent a node that does not affect the output of the decision diagram. This means that the value of the input to the `DONT_CARE` node can be anything and the output of the

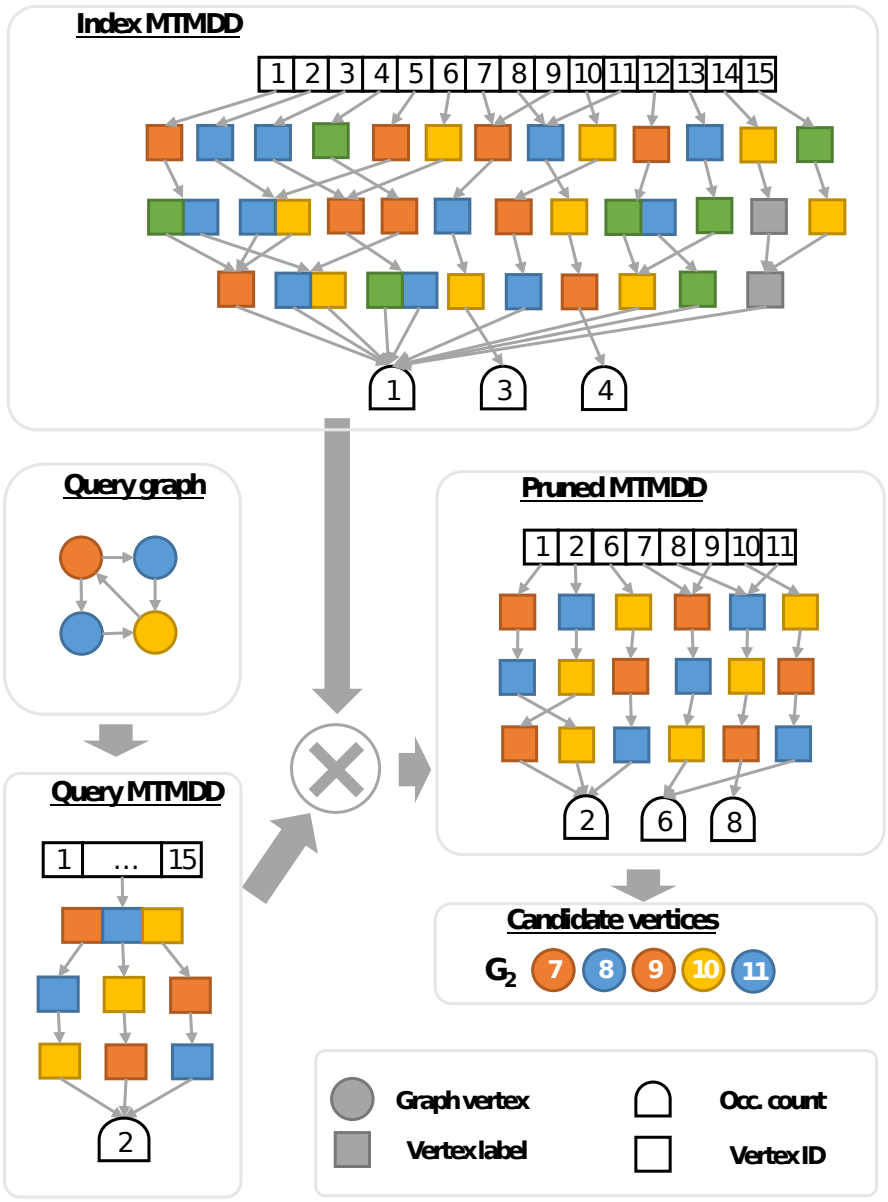


Figure 6.2

decision diagram will be the same. Basically, this represents a redundant node whose definition is provided in Section 4.1.2.

The meaning of the template query MTMDD is that initially, all the target graphs are candidates to contain the query graph.

6.4.2 Feature extraction from the index MTMDD

The template query MTMDD provides a representation of the query graph that is defined on the same graph domain as the index MTMDD.

At this point, the aim is to extract only all the query features in order to identify (i) which subgraphs of the indexed graphs satisfy the query constraints and (ii) which one of the query vertices v_k^q they match. The multiplication operator, which has been introduced in Section 4.2.3, allows the extraction from the index of all and only the features (i.e. labeled paths) appearing in the query MTMDD. This is due to the structure of the template MTMDD and to the semantics of the multiplication operator:

- The redundant location information allows each vertex belonging to the indexed graphs to potentially match the query.
- Since each feature that does not appear in the query is implicitly linked to the zero terminal in the query (template) MTMDD, the result of the multiplication operator is to *delete* from the index all the features except for those of the query graph.

Let denote as *pruned MTMDD* the results of the application of the multiplication operator between the index and the template query MTMDD. It encodes all the information to complete the filtering stage, namely the id of those vertices containing at least one feature of the query.

It is worth noting a side effect of the multiplication operator, namely that the terminal nodes of the pruned MTMDD are the actual products. Let us consider a feature p which appears n times in the query and $m_1, \ldots, m_k$ in the graphs $g_1, \ldots, g_k$. Thus, the terminal node of feature p linked to graph g_i is given by $n * m_i$. This issue has to be taken into consideration during constraint verification.

6.4.3 Constraints verification

To conclude the filtering phase is needed to identify (i) which vertices $\{v_j\}$ of the indexed graphs satisfy all the constraints of the query and (ii) which query vertex v_i^q they match.

The pruned MTMDD and the original query MTMDD are exploited to ultimate the stage. For each vertex u of the query graph G_q and a potentially matching vertex v_j in graph g_i identified by the pruned MTMDD, any feature

p starting from u should also start from v_j. Otherwise, v_j cannot be a match. Moreover, $p(g_i) \geq p(G_q)$ must hold: it is necessary to obtain the actual value of $p(g_i)$ since the terminal value provided by the pruned MTMDD is given by $p(g_i) \times p(G_q)$. The value of $p(g_i)$ is easily retrieved from the index MTMDD. There are discarded those graphs that either (i) have a feature with an occurrence number less than the occurrence number of the query or (ii) do not contain some features of the query graph.

6.5 Variable ordering

As introduced in Section 4.1.1, the ordering for the variables of a decision diagram can have a critical effect on the size of the DD itself, namely its number of nodes and edges). In particular, the order considered so far is totally arbitrary and there are no assumptions that it is a good choice for a specific collection of graphs. To the best of our knowledge, no such heuristic is currently available for reordering the variables of DDs encoding biological graph databases.

The approach presented so far is based on an MTMDD composed of n variables to index a collection of graphs using labeled paths of length up to $n-1$ as features. Hence, the n MTMDD's variables are grouped as one location information variable and $n-1$ label variables.

It is worth noting that these two types of variables are profoundly different, both in the domain spaces (i.e. graph vertices and labels, respectively) with largely different domain cardinalities, and in their semantics. To point out this difference, given a fixed position $x_n \mapsto k$ for the x_n variable, let us define the stratum k as the subset of variable orders $\{O\}_k$ sharing the fixed position for v_n.

6.5.1 An entropy-based metrics

Each variable assignment $x = \langle x_1 \dots x_n \rangle$ within the MTMDD Xcorresponds to an associated terminal value, denoted as $mult(x)$. In the following, we will refer to (i) variable assignments as tuples, (ii) MTMDDs as multisets, and (iii) terminal values as multiplicities.

Given a multiset X, let $H(X)$ be the *entropy* of X, defined according to

the standard definition [122]

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x), \quad \text{with: } p(x) = \frac{mult(x)}{\sum\limits_{x' \in X} mult(x')} \tag{6.2}$$

Let $\mathcal{U} \subseteq \mathcal{N}$ be a subset of the problem's variables. Let $x' = x/\mathcal{U}$ be a new tuple x' obtained from a tuple x by removing all the variables not in $\mathcal{U}$. Let $X/\mathcal{U}$ be the projection of the multiset X over the sole variables $\mathcal{U}$, with

$$mult(x') \;=\; \sum_{x \in X,\ x' = x/\mathcal{U}} mult(x) \tag{6.3}$$

the multiplicity of each tuple x'.

Given a variable order $O = \{k_1 \dots k_n\}$, we define the i-th variable subset $\mathcal{U}_{O,i}$ as the set of the first i variable indices of O. We define the *SOE* metric for a variable order O as

$$SOE(O) \;=\; \sum_{i=1}^{n} H\big(X/\mathcal{U}_{O,i}\big) \tag{6.4}$$

Research question R1 The size of the MTMDD (i.e. the sum of its nodes and edges) correlates with the *SOE* function.

The correlation allows us to determine whether the maximization of the *SOE* metric influences the size of the decision diagram, namely its number of nodes and edges.

To test this hypothesis, we construct the MTMDD for all the variable orders (which is factorial in the number n of variables), and compute a correlation score between the value (6.4) and the final MTMDD size.

6.5.2 The heuristic algorithm

Unfortunately, finding the optimal MTMDD by constructing all the permutations is not feasible in practice, except for a limited number of encoded variables. Therefore, to make the technique broadly applicable in a real-world context, we define a sub-optimal heuristic EntropyHeu that searches a variable order O^* by applying a greedy optimization the local entropy sum at every projection step i.

The pseudo-code of EntropyHeu is shown in Algorithm 1. The function EntropyHeu first computes the ordering for the $\{v_1 \dots v_{n-1}\}$ label variables.

Algorithm 1 Variable ordering selection heuristic

```
1:  function EntropyHeu
2:      O ← EntropyHeuLabels()
3:      minSize ← ∞
4:      for i ← [1 ... n] do
5:          O' ← InsertAt(O, {v_n ↦ i})
6:          DD' ← BuildDD(O')
7:          if SizeDD(DD') < minSize then
8:              minSize ← SizeDD(DD')
9:              O* ← O'
10:     return O*
11: function EntropyHeuLabels
12:     U ← {v_1 ... v_{n-1}}
13:     O ← {}
14:     for i ← [1 ... n − 1] do
15:         v_sel ← null
16:         H_sel ← −∞
17:         for v' ∈ U do:
18:             U' ← U \ {v'}
19:             H' ← Entropy(U')
20:             if H' > H_sel then
21:                 v_sel ← v'
22:                 H_sel ← H'
23:         U ← U \ {v_sel}
24:         O ← Append(O, {v_sel ↦ i})
25:     return O
```

Finally, it then tries to insert the identifier variable v_n in all the positions, returning the order O^* that minimizes the final DD size.

The function EntropyHeuLabels is the core heuristic algorithm, performing the greedy search. It starts by defining an empty variable order O and by taking into account the full set of label variables $\mathcal{U}$. At each outer iteration (lines 17-28), a variable $v_{\text{sel}} \in \mathcal{U}$ is removed and assigned to position i in the order O. The variable v_{sel} is chosen to be the one that maximizes the entropy given by the remaining set of variables $\mathcal{U} \setminus \{v_{\text{sel}}\}$, namely:

$$v_{sel} = \arg\max_{v \in U} H(U \setminus v) \tag{6.5}$$

We assume that BuildDD(O) generates the MTMDD for the projected

variables subset with order O, and ENTROPY($\mathcal{U}$) computes (6.2) on the projected multiset $X/\mathcal{U}$.

Research question R2 The function ENTROPYHEU selects reasonably good variable orders, comparable with the theoretical-optimal order derived by the *SOE* metric.

Part III

Applications and tool implementation

Chapter 7

FeatSEE

In this chapter, we described in detail FeatSEE, which stands for *FEATure Selection, Explanation and Evaluation* a novel automated machine learning framework for multi-omics integration that we developed during my Ph.D. In particular, FeatSEE's novelties and strengths can be summarized as follows: (i) the implementation of stand-alone modules to perform high-level tasks without the need for programming and/or advanced computational skills; (ii) the containerization, through Docker technology, of all the implemented analysis techniques to improve the framework portability and reproducibility; (iii) the implementation of a Python module to provide the opportunity for expert users to write new software components using an object-oriented paradigm; (iv) the specification of a well-defined schema and related infrastructure to allow users to integrate their own analysis workflows in the framework.

Thus, Section 7.1 is focused on the architecture of the framework. Section 7.2 presented the components (i.e. classes) within the FeatSEE Python module and, then, Section 7.3 illustrates the end-to-end high-level modules implemented so far.

The effectiveness of FeatSEE framework is shown through the following three case studies described in the next chapter:

- Colorectal cancer (CRC) via miRNA. This work was published in [123] and it was exploited to identify a biomarker signature to accurately

distinguish between CRC and healthy samples.

- In-vitro fertilization (IVF). In this work, FeatSEE was exploited to develop a workflow whose primary goal was to enhance the explainability of results.
- Functional data integration. In this work, we combined theory-based and data-driven approaches by simulating omics data from transcriptomics through Flux Balance Analysis, and then, by applying a two-phase feature selection based on fluxomics data.

7.1 The Framework

FeatSEE is intended to be a framework oriented to the definition of user-defined workflows for multi-omics integration. It is written in Python 3 and mainly based on scikit-learn [124], the de-facto standard machine learning library in Python, which provides peer-reviewed implementations of a variety of common data preprocessing methods, supervised and unsupervised machine learning algorithms, feature engineering and selection methods, hyperparameter optimization procedures, and more.

FeatSEE is composed of a Python package and a series of ready-to-use tools, called end-to-end modules as follows.

1. The Python package provides the implementations of the fundamental entities for machine learning and data analysis, such as datasets, ML models, score functions, which are internally used to implement end-to-end modules. Advanced users have also the opportunity to further extend the package by creating news end-to-end modules, or customising the existing ones.
2. The end-to-end modules (ETEMs) provide a set of command-line tools for solving general tasks (e.g. feature selection) starting from raw files and a bunch of parameters.

Both the Python module and the ETEMs are containerized in a Docker image, which allows ETEMS to be run in a containerized way by a command-line interface. A Python 3 script with no external dependencies has been realized to allow users to execute analyses from everywhere there is an installed Python 3 interpreter and Docker versions.

7.2 Python module

Firstly, we introduce classes for representing features and datasets. Then, we go through the implementation of the feature graph formalized in Section 5.3, and subsequently with the machine learning algorithms available in FEATSEE. This section concludes with the presentation of the ETEM class, which allows the extension of the framework by implementing new end-to-end functionalities.

7.2.1 Data representation

As introduced in Section 2.2.1, tabular datasets are made up of a $n \times m$ feature matrix $\mathcal{X} \in \mathbb{R}^{n \times m}$ reporting observations, namely m-length feature vector, for each of the n examples, and a collection of metadata $\mathcal{M}$ that includes the information regarding the set of features described in the dataset. However, there are different ways to represent both the features and the datasets. Here, we propose two implementations of features and three datasets

Features

As introduced in Chapter 2, features represent specific pieces of information (attributes) within the data. The individual feature F is represented as an instance of the Feature class, whose attributes describe the name and the type (i.e. quantitative, ordinal, categorical and binary) of the feature itself. Let $type(f)$ and $dom(F)$ be the type (e.g. quantitative, categorical) and the domain (e.g. $\mathbb{R}$, interval, set of values) of the feature F, respectively. Then, a named collection of features is represented by the FeatureSet class.

Unlabeled dataset

The simplest representation of a set of data is given by the unlabeled dataset $\mathcal{D}_U = (\mathcal{X}, \mathcal{M})$, which represents a set of n examples described by means of m features. The UnlabeledDataset class is composed of a feature matrix $\mathcal{X}$ that reports the feature vectors for all the samples, and of the metadata collection $\mathcal{M}$, including the feature set $\mathcal{F} = \{f_1, \ldots, f_m\}$. The individual feature types are inferred by looking at the individual columns of the feature matrix, namely the set of values of a specific feature assumed by all the samples contained in the dataset. Feature types are exploited during missing values imputation, so

that the imputed value is the feature's mode, median or mean, for categorical, ordinal and quantitative features, respectively.

Labeled dataset

Starting from an unlabeled dataset $\mathcal{D}_U = (\mathcal{X}, \mathcal{M})$ composed of n examples and m features, a straightforward method allows obtaining its binary labeled version $\mathcal{D}_L = (\mathcal{X}, \mathcal{Y}, \mathcal{M})$, which is described in Algorithm 1.

Beyond the unlabeled dataset $\mathcal{D}_U = (\mathcal{X}, \mathcal{M})$ composed of n examples and m features, the building process of the labeled dataset is also parameterized by (i) the target feature $f_t \in \mathcal{F}_\mathcal{M}$, which can be of any type except quantitative, and by (ii) two sets of values, s_+ and s_- such that $s_+ \cap s_- = \emptyset$, containing the values of feature f_t forming the positive and negative classes, respectively. Given an $n \times m$ unlabeled dataset $\mathcal{D}_U$ and the other parameters, the output labeled dataset $\mathcal{D}_L$ will be composed of $n' \leq n$ examples and $m - 1$ features. The possibly smaller number of examples n' with respect to the initial number n is due to the fact that could exist examples x_i such that $x_i^t \notin s_+ \cup s_-$, which are not included in the final labeled dataset. Then, at line 3, the binary encoding is applied on the values of the target feature f_t of the remaining examples x_i to form the $\mathcal{Y}$ vector: $y_i = 1$ when $x_i^t \in s_+$, and $y_i = 0$ if $x_i^t \in s_-$.

Algorithm 1 Building process of a binary labeled dataset

1: **function** CREATELABELEDDATASET($\mathcal{D}, f_t, s_+, s_-$)
2: $\quad\mathcal{D}' \leftarrow$ REMOVEEXAMPLES($\mathcal{D}, \{i \mid x_i^t \notin s_+ \cup s_-\}$)
3: $\quad\mathcal{Y} \leftarrow$ GETBINARYENCODEDFEATURE($\mathcal{D}', f_t, s_+, s_-$)
4: $\quad\mathcal{D}' \leftarrow$ REMOVEFEATURE($\mathcal{D}', f_t$)
5: $\quad\mathcal{D}_L = (\mathcal{X}_{\mathcal{D}'}, \mathcal{Y}, \mathcal{M}_{\mathcal{D}'})$
6: $\quad$**return** $\mathcal{D}_L$

Explainable features

Moreover, for explainability purposes, we extend the concept of features by allowing the definition of high-level boolean features in the form of logical expressions, which are defined on a pre-existing FeatureSet $\mathcal{F} = \{F_1, \ldots, F_m\}$. Let define a logical expression e as a boolean function $e : \mathcal{X} \to \mathbb{B}$ that takes as input a feature vector $x = \langle x_1, \ldots, x_m \rangle$, which is defined as an inequality $e(x) = f_i(x) \gtrless t$ for any feature $F_i \in \mathcal{F}$, comparison symbol and threshold value $t \in dom(F_i)$.

A *logical rules* $r : \mathcal{X} \to \mathbb{B}$ is defined either as (i) a single expression e or (ii) a logical conjunctions of $m > 1$ expressions $e_1, \dots e_m$. Hence, the semantic of a rule r composed by $m > 0$ expressions is true whether all the logical rules are true, or false otherwise.

$$r(x) = \begin{cases} 1, & \text{if } e_i(x) = 1, \forall i = 1, \dots, m \\ 0, & \text{otherwise} \end{cases}$$

The RuleSet class allows us to group multiple logical rules that are defined on the same feature domain.

Rule-based dataset

The third kind of dataset provides a descriptive representation of a binary labeled dataset $\mathcal{D}_L = (\mathcal{X}, \mathcal{Y}, \mathcal{M})$ by means of a new set of m' dichotomous features (i.e. boolean domain) $\mathcal{R} = \{r_1, \dots, r_{m'}\}$ in the form of logical rules (e.g. $r_1 = f_1 < v_1 \wedge f_2 > v_2$) defined on the original features of $\mathcal{D}_L$, so that the new feature matrix is $\dots \mathbb{B}^{n \times, m'}$. Each rule $r_i \in \mathcal{R}$ is evaluated against each example x_j resulting in a new boolean feature vector $b_j \in \mathbb{B}^{m'}$ such that b_j^k corresponds to the evaluation of rule r_i against the example x_j.

7.2.2 Feature graph

The feature graph formalized in Section 5.3 has been implemented using graph-tool module [125], an efficient Python module for manipulation and statistical analysis of graphs, whose internals are mostly written in C++ for performance, using the Boost Graph Library [126]. Graph-tool module provides several algorithms that operate on the Graph class that is provided by module itself. The Graph class represents either a directed or undirected multigraph, with optional internal edge, vertex or graph properties, which are identified by a univocal name. They provide a mapping from vertices, edges or whole graph to arbitrary values (i.e. strings or numeric values), which may act as labels or weights, for instance.

The feature graph is parameterized by two sets of named functions, that are used to estimate feature relevance (vertex weights) and pairwise redundancies (edge weights), respectively.

- A set of n_s named functions $F_1^v, \dots, F_{n_s}^v$, which are used to create as many sets of vertex weights $w_v^1, \dots, w_v^{n_s}$;

- A set of n_p named functions $F_1^e, \ldots, F_{n_p}^e$, which are used to create as many sets of edge weights $w_e^1, \ldots, w_e^{n_p}$.

Moreover, vertex properties are exploited to store feature metadata, such as the names of the features, the kind (e.g. quantitative, categorical, etc), and the omics of belonging of each feature represented by the corresponding vertex.

Compute graph weights

Both vertex and edge weights are computed by estimating some statistical measures based on pairs of feature vectors. Ideally, these statistical measures have to be chosen according to the intrinsic types of features under consideration. Unfortunately, pursuing this is not so easy due to the frequent high heterogeneity of features within multi-omics contexts.

- **Pearson's Correlation Coefficient (PCC)** measures the linear relationship between two quantitative variables [127].
- **Point-Biserial Correlation Coefficient (PBCC)** measures the association between a binary and a continuous variable, an extension of PCC for one binary variable [128].
- **Cramer's V (CV)** measures the association between two categorical variables in a contingency table [129].
- **Spearman Rank Correlation Coefficient (SRCC)** measures the monotonic relationship between variables, suitable for ordinal and quantitative features [130].
- **Somer's D (SD)** measures the strength and direction of monotonic relationships, commonly used for ordinal and categorical features [131].
- **ANOVA F-Test (ANOVA)** analyzes differences among group means in a dataset, suitable for quantitative features with categorical groups [132].
- **Mutual Information (MI)** measures mutual dependence between variables, suitable for quantitative, categorical, binary, and ordinal features [133].

- **T-Test (TT):** Compares means of two groups to determine if there is a significant difference, suitable for quantitative features with binary groups [134].
- **Matthews Correlation Coefficient (MCC)**, also known as *phi coefficient* (ϕ) or *mean square contingency coefficient* in statistics, is a measure of association between two binary features that estimates the quality of binary classifications, incorporating true positives, true negatives, false positives, and false negatives [135]. It is similar to the PCC in its interpretation, and a PCC estimated for two binary variables will return the MCC [136].
- **Logistic Regression (LR):** Models the probability of an event as a function of predictor variables, commonly used for binary classification with quantitative, categorical, binary, and ordinal features [137, 138].

Table 7.1: Statistical measures for the estimation of relevance and redundancy among two features

Feature Types	Quantitative	Categorical	Binary	Ordinal
Quantitative	PCC			
Categorical	ANOVA	χ^2, Cramer's V		
Binary	PBCC, t-test, ANOVA, LR	χ^2, Cramer's V, LR	χ^2, MCC, Cramer's V	
Ordinal	SRCC	SD	PBCC	SRCC

7.2.3 Machine-learning models

Models are the result of a learning algorithm applied to a dataset. The scikit-learn library provides a wide variety of learning algorithms for different tasks (e.g. classification, regression, clustering, dimensionality reduction, etc.), as well as different tools for data pre-processing (e.g. feature encoding, data scaling, feature selection, etc).

Scikit-learn makes available the `estimator` class to implement objects that are able to take advantage of the provided data to achieve a certain goal. In particular, estimators are characterized by the fit method to learn from data, whose learning can be either supervised or unsupervised: the former takes as

input two parameters, namely the data matrix X and the target vector Y, whereas the latter takes as input only the data X. Two major classes extend the estimator one:

- predictors consists of supervised estimators, which implement the `predict` method, that
- transformers, consists either of supervised or unsupervised estimators that implement the `transform` method, which allows to modify or filter the input matrix X and return it in a new form X' (e.g. scaler, encoder, PCA, ...).

Moreover, scikit-learn provides an easy way to create new estimators by combining the available ones. A `pipeline` object allows the creation of ordered sequences of transformers with an optional final predictor, which are automatically chained and handled by the library itself.

In the large plethora of available learning algorithms provided by scikit-learn, FeatSEE allows us the usage of:

- k-Nearest Neighbors (kNN) is a geometric model whose prediction is based on the majority class of their k nearest neighbors in the feature space;
- Naive Bayes (NB) is a probabilistic model that estimates the probability of an example's class by naively assuming that the features are independent;
- Logistic Regression (LR) is a probabilistic model for binary classification based on fitting the logistic function;
- Support Vector Machines (SVM) is a geometric model able to find the optimal hyperplane to separate data into different classes while maximising the margin, which is the distance between the decision boundary and the nearest examples from each class;
- Decision Tree (DT) is a rule-based model represented as a tree-like structure obtained by recursively splitting data based on the most significant features, ultimately leading to the leaves which are labeled with a decision.

- Random Forest (RF) is a bagging ensemble model that combines a number of decision trees that are trained on bootstrap replicas of the original dataset, and whose prediction results from majority voting of individual trees;
- Gradient Boosting (GB) is a boosting ensemble model that combines a number of decision trees, which are sequentially trained to correct errors made by the previous ones.

Almost all such learning algorithms own a set of hyper-parameters that have to be defined prior to using the algorithm itself, whose optimal values have to be identified through hyperparameter optimization.

7.3 FeatSEE end-to-end modules

An end-to-end module (ETEM) is an abstract representation of the flow of a stand-alone piece of analysis for achieving a specific task. It takes as input a series of data and parameters, and a well-defined series of output data are returned as output. The ETEM's interface specifies the mandatory input data required by every module, which are listed as follows.

1. The output folder where output results will be persisted.
2. The main dataset $\mathcal{D}$, whose main purpose is to act as the training set to build ML models.
3. A list of n_t test sets $\mathcal{T} = \{D_i\}$, which are exclusively used to evaluate the performance of ML models on unseen data.
4. A list of n feature sets $\mathcal{F} = \{F_i\}$ that are exploited to execute different analyses by exploiting each of these sets as starting features.

Besides these default parameters, ETEMs can be characterized by a set of additional parameters. For instance, the ETEM that solves a binary classification task may accept the names of which learning algorithms the ETEM have to use to build the learning models, as well as the parameters to build the labeled dataset, namely the name of the target features and the feature's values to define the positive and negative classes.

The working schema of ETEM is represented in Algorithm 2. Initially, the module is initialized using the input parameters, namely the output folder,

Algorithm 2 High-level workflow of a end-to-end module

1: **function** EndToEndModule($\mathcal{D}_{tr}, \mathcal{T}, \mathcal{F}, \mathcal{A}, out, params$)
2: Initialize E2EM internal state.
3: **for** $F \in \mathcal{F}$ **do**
4: $\mathcal{D}' \leftarrow$ featureReduction($\mathcal{D}_{tr}, F$)
5: Let $\mathcal{T}' = \{\text{featureReduction}(\mathcal{T}_i, F) \,\forall\, \mathcal{T}_i \in \mathcal{T}\}$
6: runAnalysis($\mathcal{D}', \mathcal{T}', \mathcal{A}, params$)
7: Write the produced results in *out* folder.

the training data, the n_t test data and the n_f feature sets. Once initialization has been completed, the provided feature sets are exploited to choose which features of the training and test data have to be used in the analysis implemented by the ETEM itself. Therefore, given n_f feature sets, as many analyses will be run by the ETEM by building different copies of the training and test data, which are defined over the same samples but are described by different features. Once all the analyses have been completed, results are post-processed and persisted on the file system in the specified output folder.

7.3.1 Data preparation module

The end-to-end module for data preparation is devoted to the validation of the input parameters provided by the user, the loading of input files and the instantiating of the FeatSEE objects representing the entities, such as datasets and feature sets.

Parameters

It takes as input data different series of files:

1. a list of one or more files is used to build the training set. Generally, the first file contains clinical variables, whereas others are relative to omics data. The idea is to allow the user to provide the system with different files for different omics and then, to ease the realisation of different multi-omics integration strategies.

2. an optionally empty list of files, each one representing an independent dataset, namely composed of different samples from the ones appearing in the training set;

3. an optionally empty list of files, each one representing a different feature set identified by a unique name.

Moreover, different sets of parameters are required for the labeling procedure and to perform some other operations:

4. labeling parameters for binary classification: (a) the target feature f_t, and (b) two mutually exclusive sets of values C_{pos} and C_{neg} of f_t defining the positive and negative classes.

5. feature graph construction parameters: the *complete* feature graph is built based on (a) the vertex score functions and (b) the edge score function;

6. the test set proportion size, which is a value $0 \leq s < 1$. Whether it is non-zero, it is used to perform a train/test split, so that the existent training set is reduced in size and a new test set is formed by the extracted samples. Let N be the number of samples in the whole training set: $N \cdot s$ samples are used to build a new test set, and the remaining $N \cdot (1 - s)$ samples are actually used to build the new training set. The split is randomly performed in a stratified fashion to ensure that the class proportion of the original training set is maintained in both datasets.

Behavior description

Initially, feature sets and unlabeled dataset objects are built starting from raw files provided as arguments. If there are no feature sets provided, all the features of the main dataset will be considered. Otherwise, the overall feature set, namely the union of the input feature sets, is built and the features that are not required are removed from the provided datasets.

Then, given the target feature f_t and both the positive and negative classes C_{pos} and C_{neg}, labeled datasets are built from the unlabeled ones according to the labeling parameters.

The main dataset is split into a training and a test set whether the test set proportion size $s > 0$, otherwise the whole main dataset will be considered as the training set.

Then, standardization is applied through *z-scores* to ensure the same scale for all the features by centring the data around the mean with one standard

deviation. The training set is used to compute the mean and the standard deviation vectors, μ and σ, respectively. Given feature i, μ_i and σ_i are the mean and standard deviation of feature i in the training set. Then, given a feature vector x, the standardized feature vector z is computed as $\mathbf{z} = \frac{\mathbf{x}-\mu}{\sigma}$.

Lastly, the training set is used to build the feature graph, given the list of n score functions to estimate feature importance, which results in a feature graph having n named sets of vertex weights.

7.3.2 Evaluation module

The end-to-end module for evaluation estimates a wide series of metrics performance for a given binary classification task, by using multiple learning algorithms and feature sets.

Parameters

Besides the input data parameters concerning datasets (training and tests) and feature sets, the evaluation module requires to know

1. which learning algorithms have to be used;
2. the learning settings, namely whether to use LOOCV or repeated stratified k-fold CV. LOOCV does not require additional parameters, whereas the k-fold CV requires (a) k, the number of folds, and (b) t, the number of repeats of the k-fold CV procedure;
3. the number of processes to be used to parallelize the CV procedure.

Behavior description

The behaviour of the evaluation ETEM is described in Algorithm 3. It consists of three main functions, which are explained as follows.

The function performanceEvaluation is simply devoted to split the training set into folds based on the chosen CV technique; the split is done once, so that all the feature evaluations are done following the same split schema, namely using exactly the same samples for training and testing. For each feature set provided, the function evaluateFeatureSet is called to estimate the classification performances using the provided set of learning algorithms.

Algorithm 3 Evaluation module

```
1:  function registerPredictionData(model, test_data )
2:      y_pred ← predict(model, test_data)
3:      calcola performance tra pred and actual labels
4:      salva queste cose da qualche parte

5:  procedure evaluateFeatureSet(TrSet, cvSplits, TestData, AlgoSet)
6:      for (tr_split, test_split) ∈ cvSplits do
7:          TrSplit ← TrSet[tr_split]
8:          TeSplit ← TrSet[test_split]
9:          for a ∈ AlgoSet do
10:             model ← Fit(a, TrSplit)
11:             registerPredictionData( model, TeSplit )
12:             for t_data ∈ TestData do
13:                 registerPredictionData(model, t_data )
14:     Process performance estimation data

15: function performanceEvaluation(TrSet, TestData, FSets, AlgoSet)
16:     cvSplits ← CrossValidation(TrSet)
17:     for fset ∈ FSets do
18:         TODO: create tr e test sets con features in fset!!
19:         cose ← evaluateFeatureSet(TrSet, cvSplits, TestData, AlgoSet)
20:     return scores
```

Internally, the function evaluateFeatureSet iterates over training/test split provided by the CV, which corresponds to different sub-training and -test datasets extracted from the original training set provided as input to the function (lines 7-). For a fixed split, all the provided algorithms $A = \{a_i\}$ are trained on the training portion in order to derive a model (line 10), that is evaluated (i) on the test portion (line 11 and (ii) on the provided independent test sets (line 13).

Once the model has been built through the Fit(p)rimitive, it can be used for prediction which are handled by the function registerPredictionData, which takes as input (i) the fitted model and (ii) a named set of labeled data on which perform a prediction and evaluate it. At line 2, the model is asked to predict the labels of the data provided as input to the function, in order to obtain a prediction vector. This is compared against the actual labels of the data in order to compute the set of performance metrics presented in Section 2.2.1, which are computed in a dataset-specific fashion.

7.3.3 Ensemble feature selection module

The end-to-end module for ensemble feature selection employs data and function perturbation techniques to identify robust and predictive feature sets.

Parameters

In addition to the input data parameters, the ensemble FS module takes as input

1. nf_{min} and nf_{max}, which define the range $[nf_{min}, nf_{max}]$ of the number of features to take into consideration,
2. the data perturbation parameters, namely k and t, which are used as arguments for a repeated stratified k-fold CV;
3. the function perturbation parameters, which is a non-empty list of feature importance rankers, called *selectors* hereinafter, to predict salient features from a given set of data;

Behavior description

Algorithm 4 ...

```
1: procedure storeFSet(candidates, selector, data, nf)
2:     fset ← fit(founds, selector, data, nf)
3:     if lookup(candidates, fset) then
4:         increment(candidates, fset)
5:     else
6:         insert(candidates, fset )
7: function DataPerturbation(data, k, n)
8:     splits ← RepKFoldCV(k, n)
9:     Build the samplings of data
10:    return The list of sampled datasets
11: procedure ensembleFeatureSelection(TrSet, TestData, selectors, AlgoSet)
12:    candidates ← hashtable
13:    for data ∈ DataPerturbation(TrSet, ...) do
14:        for sel ∈ selectors do
15:            for nf ∈ [nf_min, nf_max] do
16:                storeFeatureSet(candidates, sel, data, nf)
17:    stats ← performanceEvaluation(TrSet, TestData, candidates, AlgoSet)
18:    Postprocessing evaluation results
```

The pseudocode of the ensemble feature selection is provided in Algorithm 4. It consists of a series of nested loops to combine the data perturbation and function perturbation strategies, as well as varying the number of features to be considered.

Feature selection phase The most-nested loop explores the feature set search space by varying the feature set length in the interval from nf_{min} to nf_{max}. Given a portion of data, a selector s and the number of features to select n, the function storeFeatureSet provides to fit the selector algorithm using the available data and exploit the selector to predict the n most important features.

Whenever a feature set is identified, the number of times it has been identified is incremented, in order to take into account the number of times each sets has been identified to evaluate the stability of each set.

Performance evaluation phase Once the feature selection is terminated, the candidate set is evaluated in a cross-validation setting (line 17 in order to obtain a set of metrics for each feature set. Results of the evaluation are exploited to rank the candidate feature sets and to provide as output the ones showing the highest classification performances on the provided test sets.

7.3.4 Filter-based feature selection module

The end-to-end module for filter-based feature selection performs a supervised dimensionality reduction based on the mRMR principle by exploiting the feature graph formalism defined in Section 5.3.

Parameters

The parameters required by such a module have the purpose of setting the configuration of the genetic algorithm, which acts on the feature graph.

1. Edge threshold value e_t in the complete feature graph, which is used to cut all the edges which are below the threshold;

2. The maximum number of features to select, namely the length of the integer chromosome;

3. Population size ps, namely the number of candidate solutions to be simultaneously considered during each generation;

4. The minimum number of generations n_{gen} before considering convergence, and an integer value δ_{gen} to define the convergence behaviour;

5. The size of the mating pool mps, namely the fraction of the population that is selected for reproduction;

6. The number of elitism individuals n_e that ensures that the best n_e individuals of the current population are directly carried to the next generation;, to avoid premature convergence and the loss of good solutions;

7. The rates for probabilistic operators of crossover and mutation, $0 \leq p_c \leq 1$ and $0 \leq p_m \leq 1$.

Behavior description

The algorithm is provided in Algorithm 5 is composed of different stages, namely (i) the feature graph construction, (ii) the execution of the genetic algorithm over the feature graph, and (iii) the final evaluation of the selected features exploiting ML algorithms.

Graph building phase Initially, the feature graph is built based on the training set data and on the provided FI and FR functions to assign vertex and edge weights. Once the whole graph G is built, multiple weighted graphs $g_{i,j}$ where i and j indicate which FI and FR, respectively, are used to set vertex and edge weights.

Genetic algorithm phase The genetic algorithm takes as input a feature graph $g_{i,j}$ weighted with respect to the (i) feature importance function I_i and (ii) feature redundancy functions R_j, for vertices and edges, respectively.

At line 15, an empty list of candidate solutions is instantiated. It is used to keep track of the fittest chromosomes together with their fitness value and the generation at which that solution had been found. The algorithm starts by creating an initial population composed of ps chromosomes through the createPopulation function, which takes as input (i) the whole feature graph and (ii) the population size. Then, the population pop of n chromosomes are randomly initialized.

At this point, the algorithm starts by evolving the population until the termination criteria have not been satisfied.

1. Generations start by evaluating the individual chromosomes belonging to the current population through the evaluatePop function at line 2.

2. From a given chromosome, the corresponding feature graph is obtained (line 5) and it is subsequently used to compute the fitness of that chromosome (line 6). The function returns the array of fitness score functions corresponding to the chromosomes in the population.

3. The acquired fitness scores are possibly used to update the candidates list. The function updateBestSolution verifies whether the highest score among the available ones is superior to the last best solution found on a previous generation. If does, it is saved in the candidates.

4. Scores are further exploited by the selection operator to perform the k-tournament selection and build the mating pool, which is composed of chromosomes that pass to the reproduction stage.

5. The reproduction consists of sampling with replacement of n pairs of chromosomes to be mated from the mating pool. Then, the crossover is applied to each pair with probability p_c and eventually, the mutation operator is applied to the results of the crossover operator.

6. The mutation operator mutates each gene of the chromosome with probability p_m. Consider locus i is subject to mutation. If the corresponding gene encodes for a feature f_i, it is either swapped with one of v_i's neighbours, or the gene at locus i is set as non-coding. Conversely, if the gene to be mutated is a non-coding one, it is set to a random feature among the available in the whole graph.

7. The algorithm executes at least n_{gen} generation before considering convergence criteria. The algorithm converges whether there are performed δ_{gen} generations with no update solutions, and it returns the list of identified candidate solutions.

Algorithm 5 Genetic graph algorithm

```
1: Let ps be the population size
2: function evaluatePop(pop)
3:     scores ← [0, ..., 0]                          ▷ list of ps elements
4:     for i ← [0, ps] do
5:         subg ← gpm(g, pop[i])
6:         scores[i] ← computeObjFunction(subg)
7:     return scores
8: function reproduction(pop, p_c, p_m)
9:     newPop ← []
10:    for i ← [0, ps − 1] do
11:        [chr_1, chr_2] ← CrossoverMutation(pop, p_c, p_m)
12:        newPop ← insert(newPop, chr_1, chr_2)
13:    return new_pop
14: function GraphGA(Graph, ps, n_gen,)
15:    candidates ← new list
16:    bestSolution ← (0, 0)
17:    new_pop ← createPopulation(Graph, ps, n_f)
18:    Let i ← 0 be the current generation
19:    while terminationCriteria(i, n_gen, n_no_change) do
20:        scores ← evaluatePop(new_pop)
21:        updateBestSolution(pop, scores, i)
22:        selected ← selection(new_pop, scores, k)
23:        new_pop ← reproduction(selected, p_c, p_m)
24:    return candidates
```

Algorithm 6 Crossover and mutation operators

```
1: function mutation(chr, p_m)
2:     for i ← [0, |chr| − 1] do
3:         if random < p_m then
4:             if chr[i] encodes for a feature then
5:                 Swap feature with neighbor in G
6:             else
7:                 chr[i] ← randomFeature
8:     return chr
9: function CrossoverMutation(population, p_c, p_m)
10:    Sample two items [chr_1, chr_2] from population
11:    if random < p_c then
12:        [chr_1, chr_2] ← tpx(chr_1, chr_2)
13:    return [mutation(chr_1), mutation(chr_2)]
```

7.3.5 Explainable feature extraction module

The end-to-end module for explainable feature extraction is devoted to extracting a new set of binary features from a given training set, allowing us to describe that dataset by logical rules.

Parameters

The parameters required for the feature extraction phase are

1. the data perturbation parameters, namely the CV settings (number of folds and repeats, k and t respectively);

2. nr_{min} and nr_{max}, which define the range $[nr_{min}, nr_{max}]$ of the number of features (i.e. rules) to take into consideration.

Behavior description

Three different phases characterize the EFE module, namely rule extraction, rule selection and eventually rule evaluation.

Given the training set D described by feature set F, (i) the rule extraction phase aims to generate a new feature set, denoted as a ruleset as follows, composed of explainable features called rules that are defined on the original feature set F. Since the number of extracted rules could be high, (ii) the second phase is a feature selection approach to identify the most promising explainable features to shrink the size of the ruleset in a user-defined acceptable range. Finally, (iii) the optional third phase concerning the rule evaluation phase exploits the evaluation ETEM that solves the original binary classification task using the identified rulesets of incremental size.

Rule extraction phase Rule extraction phase has been implemented by exploiting the Skope-rules python package [139], which is built on top of scikit-learn. Skope-rules exploits ensembles of trees (i.e. decision tree classifiers and regressors) to extract rules from data. A tree ensemble (e.g. random forest) is fitted to learn rules from data, which are filtered based on recall and precision thresholds, to remove low-performing rules. Then, duplicate and similar rules are removed by performing a selection based on the diversity of logical terms (variable + larger/smaller operator) and performance (F1-score) of the rules [140].

Specifically, the Skope-rules functionality is used to identify two sets of rules, describing the positive and the negative class, respectively. The rule extraction has been performed in a cross-validation setting, in order to limit overfitting by extracting rules from sub-samples of the training set. Finally, the two sets of rules are merged into a single ruleset $\mathcal{R}$, which is used to build the corresponding rule-based datasets of training and test sets.

Rule selection phase In the next step, an ad-hoc feature selection for binary features is performed to identify a set of relevant and non-redundant rules by exploiting the MCC for the estimation of both rules' relevance and redundancies.

The feature graph $G_R = (V, E, L)$ is built from the rule-based training set using the MCC for assign vertex and edge weights. We will refer to G_R as rule graph as follows. Then, redundant rules are removed by a greedy algorithm that exploits the strong correlation edges of the graph G_R, namely, such edges whose weight is near to one in magnitude. A threshold value e_t is chosen and a second rule graph $G'_R = (V, E', L)$ is obtained by considering only the edges such that $|w_e| > e_t$. Then, for each connected component appearing G'_R, only the highest scored vertex (i.e. rule) is kept, whereas others are discarded. Finally, the rules discarded by the greedy algorithm are removed from both the rule-based datasets and the rule graph G_R.

Performance evaluation phase Finally, the rules are ordered with respect to their MCC against the target feature, and they are evaluated in the classification task by building rule sets of incremental lengths.

Algorithm 7 ...

```
1: function ruleMining(TrSet, k, n)
2:     Init empty rule list Ruleset
3:     for cv_split ∈ RepeatedKFold(k, n) do
4:         for class ∈ {Pos, Neg} do
5:             rules ← SkopeRules(TrSet, cv_split, class)
6:             insert(rules, Ruleset)
7:     return Ruleset
8: procedure featureExtraction(TrSet, TestData, Features, AlgoSet)
9:     Ruleset ← ruleMining(TrSet, k, n)
10:    TrSet_R ← RuleBasedDataset(TrSet, Ruleset)
11:    G_tr ← FeatureGraph(TrSet_R)
12:    G_tr ← RuleDeduplication(G_tr)
13:    Sort rules w.r.t. vertex weights of G_tr
14:    Build rulesets R = {R_2, R_3, ..., R_n}
15:    performanceEvaluation(Tr_R, Test_R, R, AlgoSet)
16:    Process data
```

Chapter 8

GRAPES-DD

In this chapter, we presented the implementation details about GrapesDD, a software based on GRAPES that exploits decision diagram data structures for the indexing of collections of graphs and for the efficient search of specific substructures within the indexed collections, as described in Chapter 6.

Thus, Section 8.1 provides a general overview of the tool. Then, Section 8.2 provides the algorithm for the construction of the index starting from a graph collection. Section 8.3 describes the algorithm for the efficient filtering of the index given a query graph. Finally, Section 8.4 showed the implementation details of the algorithm to optimize the memory efficiency of the index MTMDD through the reordering of its variables.

The effectiveness of GrapesDD is shown in Section 9.4 using both real and synthetic collections of graphs. In particular, the following case studies are described:

- Graph indexing and subgraph searching. This work was published in [121] and employes both real and synthetic collections of graphs, and the performance are compared against the state-of-the-art tools.
- Memory optimization via variable reordering. The work published in [141] is the application of the entropy-based heuristic on the MTMDDs

of biological graphs collections to answer the two research questions presented in Section 6.5.

8.1 Overview

In [121] we proposed a new version of GRAPES, called GRAPES-DD, which exploits the decision diagram (i.e MTMDD) to achieve a substantial reduction of the memory footprint of the index graphs. The goal was reached thanks to DD's ability to efficiently handle the presence of similar patterns in the indexed graph paths.

GRAPES-DD has been written in C++ and exploits the decision diagrams implementation provided by the Meddly Library. Moreover, the whole tool has been containerized in a Docker[29] image to ensure the experiments' functional and computational reproducibility. The Dockerfile to build the image is provided together with the source code, and it is available at `https://github.com/qBioTurin/grapes-dd` and at `https://github.com/InfOmics/grapes-dd`.

The GRAPES-DD workflow is composed of three main stages, namely indexing, filtering and verification, as in the original version of the GRAPES tool. During the indexing stage, the MTMDD storing all the labeled paths up to length l is built from the collection of target graphs. During the filtering stage, a query MTMDD is built from the query graph and it is exploited to extract the candidate set from the index, namely those subgraphs potentially containing the query graph. Finally, the verification stage is applied to the candidate set provided by the filtering phase: the GRAPES subgraph isomorphism algorithm, either VF2 or RI, can be executed to find all the occurrences of the query graph.

8.2 Indexing stage

The indexing stage introduced in Section 4.3 is shown in Algorithm 8.2. The indexing function takes two arguments: the graph collection $D = \{g_1, \ldots, g_n\}$ to be indexed, and the maximum feature length $l > 0$.

The algorithm is composed of two phases. In the former phase, the data structures required for the indexing process are properly initialized, whilst

the second phase consists of storing all the graph features, together with the additional information, in the index MTMDD.

8.2.1 Index preparation

As introduced in Section 4.3, to create a forest of decision diagrams, the Meddly library requires to know the domain bounds for the required variables, namely a k-length vector $x_{\text{bounds}} = \langle b_1, \ldots, b_k$ such that (i) k is the number of variables, and the i-th elements indicates that the admissible values for variable x_i ranges between 0 and $b_i - 1$.

GrapesDD is based on the MTMDD using the flattened location information introduced in Section 6.2. Thus, given the maximum feature length l, the MTMDD domain is composed of $l + 1$ variables: the location information variable x_{liv} and l location variables $x_{lbl_1}, \ldots, x_{lbl_l}$.

The default variable order is $\mathcal{O} = x_{liv} \succ x_{lbl_1} \succ \cdots \succ x_{lbl_l}$, meaning the root node is labeled with variable x_{liv}, the level below the root node is assigned to x_{lbl_1} and so on. The non-terminal level at the bottom of the DD is assigned to x_{lbl_l}.

The domain bounds for variables are estimated by looking at the graph collection D: let N_V and N_L be the total number of vertices and of labels within D, respectively. Two encoders are defined: (i) the label encoder $E_L : \Sigma \to \mathbb{N}_+$ and (ii) the vertex encoder $E_V : G \times V \to \mathbb{N}$.

The vertex encoder E_V maps specific vertices within the graph collection D in the range $[0, \ldots, N_V - 1]$. The label encoder E_L maps the labels in the range $[0, N_L]$; in particular, labels are encoded in the range $[1, N_L]$, whereas the null value encodes for no label.

Considering the default variable ordering defined above, the corresponding domain bounds is $\langle N_V, N_L + 1, \ldots, N_L + 1 \rangle$.

Once the domain bounds have been defined, the forest $\mathcal{F}$ is instantiated by specifying (i) the range of terminal nodes among boolean, integer and real, and (ii) which kind of decision diagram among *multi-terminal* and *edge-valued* types. To obtain an MTMDD forest we specified (i) integer as the range of terminals, and (ii) multi-terminal as the second option.

8.2.2 Index construction

Initially, an empty MTMDD $\mathcal{I}$ is created from forest $\mathcal{F}$.

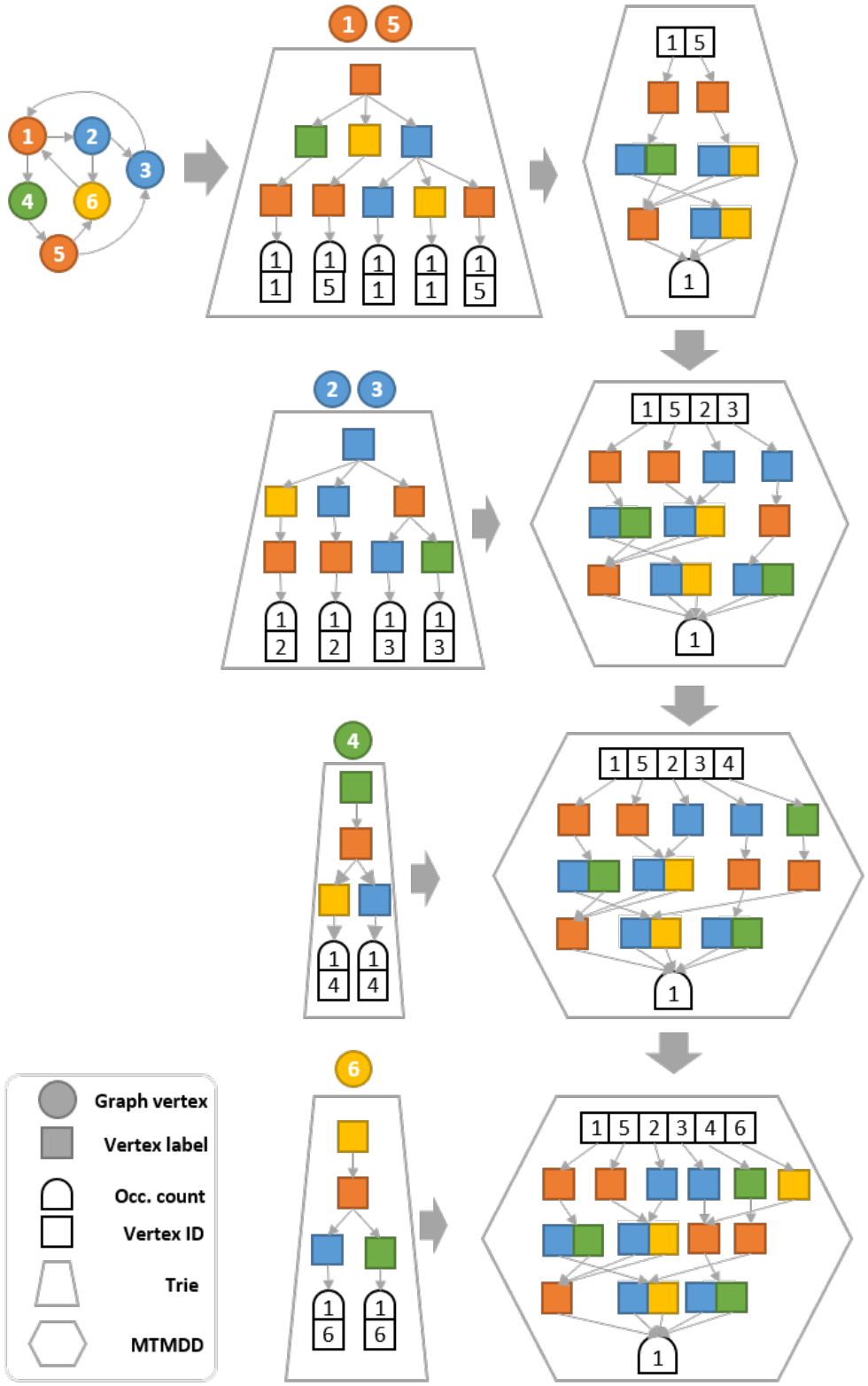


Figure 8.1: ...

Algorithm 8 Graph indexing algorithm

```
1: Let 𝓕 be the Meddly forest
2: Let 𝓑 = (B_x, B_y) be the buffer
3: Let E_V and E_L be the encoding functions
4: procedure flushToDD(𝓑, index)
5:     tmp ← createEdge(𝓕, B_x, B_y)                ▷ Get data from 𝓑
6:     apply(+, index, tmp, index)                  ▷ Merge tmp with the index
7:     setAsEmpty(𝓑)
8: procedure flushToBuffer(g, trie, index)
9:     for each p = (σ_1, ..., σ_l) ∈ trie do
10:        n_occ ← getFeatureCount(trie, p)
11:        for each v ∈ getFeatureVertices(trie, p) do
12:            tuple ← ⟨E_v(g, v), E_L(σ_1), ..., E_L(σ_l)⟩
13:            insert(𝓑, tuple, n_occ)
14:            if isFull(𝓑) then
15:                flushToDD(𝓑, index)
16: function createIndex(D, l)
17:    Create index MTMDD 𝓘 from forest 𝓕
18:    Initialize empty buffer 𝓑 = (b_x, b_y)
19:    for g ∈ D do
20:        VLabelSet ← groupByLabel(g)              ▷ A set of sets
21:        for VSet ∈ VLabelSet do
22:            Initialize empty trie
23:            for node ∈ VSet do
24:                DFS(g, node, trie)
25:            flushToBuffer(g, trie, 𝓑, 𝓘)
26:    if not isEmpty(𝓑) then
27:        flushToDD(𝓑, 𝓘)
28:    return 𝓘
```

Then, a buffer object $\mathcal{B}$ is initialized. Its purpose is to accumulate the tuples (and the corresponding return values) to be stored in the index $\mathcal{I}$. The buffer is used to have a large capacity to reduce the number of insertion operations. Such a buffer object is internally composed of two buffers, b_x and b_y, having the same length. The former is designed to store the variable assignments that will define the non-terminal nodes, whereas the latter is devoted to storing the terminal values corresponding to the tuples in b_x.

The actual indexing phase, which is depicted in Figure 8.1, is performed one graph at a time, by grouping its vertices with respect to their labels to

ease counting the occurrences of the extracted features within the whole graph. Then, each group of vertices are indexed separately by relying on auxiliary tries to accumulate path occurrences. Once all the vertices belonging to a group are indexed, the content of the trie is transferred to the buffer.

Whenever the buffer is full, it is flushed in the index MTMDD. The flushing procedure consists of creating a temporary MTMDD from $\mathcal{F}$, where the buffer's content is stored by calling CREATEEDGE(b_x, b_y), namely by stating a set of variable assignments and the corresponding return values. The MTMDD produced by such a process is then merged in the index by performing an addition operation between the index and the temporary MTMDDs, whose result became the new index MTMDD.

8.3 Filtering stage

The implementation of the filtering stage introduced in Section 6.4 is shown in Algorithm 9. The filtering function takes two arguments: the index MTMDD $\mathcal{I}$, and a query graph $G_Q = (V_Q, E_Q, L_Q)$ to be searched in the graph collection indexed by $\mathcal{I}$.

The algorithm starts by initializing a series of Meddly empty edges, Q, T and P, which are used for different purposes during the filtering stage. In particular, (i) Q is used to index the query graph using features up to the same length as those in I, (ii) T encodes the template query graph, and (iii) P encodes the pruned index.

8.3.1 Query preprocessing phase

Firstly, a similar indexing procedure to the one used to build I is applied to the query graph G_Q: all the features are extracted from G_Q and stored in the MTMDD Q. Query vertices are enumerated in $[0, \ldots, |V_Q|)$, and such values are used for the location information variable in the MTMDD Q.

8.3.2 Feature extraction phase

It is worth noting that the location information of the query MTMDD $\mathcal{Q}$ is incompatible with the one of the index $\mathcal{I}$, since they represent different vertices sets. In fact, the former refers to the vertices of the query graph, whereas the latter is about the vertices of the indexed graphs. To solve such compatibility

Algorithm 9 Filtering algorithm

```
1: Let 𝓕 be the Meddly forest
2: function queryFiltering(𝓘, G_Q)
3:     Let Q, T, P be new Meddly edges
4:     IndexQuery(G_Q, Q)
5:     BuildQueryTemplate(Q, T)
6:     Apply(×, 𝓘, T, P)
7:     cset ← QueryConstraintsVerification(P, Q)
8:     return cset
9: function checkQueryConstraints(P,Q)
10:    initGraphMatchDataStructure(Q)
11:    e ← getMeddlyEnumerator(P)
12:    repeat
13:        tuple ← getVarAssignment(e)
14:        loc ← getLocationInfoValue(tuple)
15:        q_supp ← getQuerySupportsInfo(e, loc)
16:        for (v_q, s) ∈ q_supp do
17:            vpaths ← getFeaturesFromQueryVertex(v_q)
18:            if s >= len(vpaths) then
19:                matched ← true
20:                for p ∈ vpaths do
21:                    occ_q ← evaluate(Q, ⟨v_q, p_1, ..., p_l⟩)
22:                    occ_v ← evaluate(𝓘, ⟨loc, p_1, ..., p_l⟩)
23:                    if occ_v < occ_q then
24:                        matched ← false
25:                        break
26:                if matched is true then
27:                    setGraphMatch(loc, v_q)
28:    until isValid(e)
29:    return getCompleteMatches()
```

issues, and especially to allow us to match a vertex query against vertices belonging to the index, at line 5 the buildQueryTemplate() function creates the template query MTMDD T from MTMDD Q by replacing the location information variable with a single DONT_CARE node.

Then, at line 6, the multiplication operator is applied between the index $\mathcal{I}$ and the template query T, whose product is stored in the pruned MTMDD P. All the features which appear in the query are extracted from the index $\mathcal{I}$, together with (i) the correct location information regarding the graphs in the collection D, and (ii) the occurrences counters. Practically, the pruned

MTMDD P encodes for a set of subgraphs of the original graphs indexed in $\mathcal{I}$. Each subgraph is composed of a subset of the vertices of one of the indexed graphs such that each vertex (i) has a label $l \in L_Q$ and (ii) from which starts at least one feature (i.e. labeled path) in common with the query Q.

8.3.3 Constraints verification phase

Subsequently, the queryConstraintsVerification() function identify which subgraphs (i.e. sets of vertices) encoded in the pruned index $\mathcal{P}$ satisfy the query constraints, and the candidate set, namely the set of subgraphs containing all the query features, is returned as output.

The constraints verification strategy is based on the enumerator class provided by the non-expert interface of the Meddly library, which allows us to iterate over the DD one variable assignment at a time.

Then, given a variable assignment $x = \langle x_1, \ldots, x_n \rangle$ describing a feature p starting from a certain graph vertex, the location information of the current x is decoded to get the graph and vertex identifiers, g_{id} and v_{id}, relative to the feature p. At line 15, since each graph vertex (g_{id}, v_{id}) can potentially match multiple query vertices, all the paths starting from the current vertex are retrieved from the query: for each of these paths, we obtain the query vertices from which they start.

At this point, the for loop in lines 16-27 verifies whether the current vertex is matchable against one or more query vertices, and in case, on which query vertices does it match. The conditional statement at line 18 filters out the current graph vertex if an insufficient number of features start from it. If the vertex (g, v) passes the filter, the actual query constraints are checked:

- for each feature p starting from a query vertex v_q, all other features p' starting from v_q must also start from (g, v);
- the number of occurrences of each feature in g must be at least equal to the number of occurrences of the same feature in the query graph

If both the above conditions hold, the current vertex is set as a candidate to match the current query vertex v_q. Once the whole pruned MTMDD P has been visited, the match vector is examined in order to retrieve the graphs for which there exists a complete match, namely a graph $g = (V, E, L)$ where every query vertex is fully matched against at least one vertex $v \in g$.

Finally, for each vertex of the query, the algorithm reports the list of the matchable vertices of the indexed graphs passing the filtering phase. The overall effect is that the algorithm extracts from the graph collection all the maximally connected components composed only by the vertices involved in the query graph. Over these components, a subgraph isomorphism algorithm can be executed to find all the occurrences of the query graph.

8.4 Entropy-based variable ordering

The variable ordering selection heuristic described in Algorithm 1 has been implemented in C++ using the Meddly library. The implementation details of EntropyHeu() are shown in Algorithm 2. The goal of the algorithm is to identify an ordering of the variables of the index MTMDD $\mathcal{I}$ that reduces the size of the index, which is measured as the total number of nodes and edges that compose the DD. Considering an index MTMDD $\mathcal{I}$ using features up to length l, the variable set of the index MTMDD is composed of $l+1$ items: $\mathcal{I}$ is $\{x_{liv}, x_{lbl_1}, x_{lbl_2}, \ldots, x_{lbl_l}\}$

The algorithm is composed of two steps. The first step is implemented by the EntropyHeuLabels() function, which builds a partial ordering considering label variables $\{x_{lbl_i}\}$ only, whereas the second step is provided by the entropyHeu() function that completes of the partial ordering by positioning the location information variable x_{liv}.

8.4.1 Label variables reordering

The creation of the partial order composed by the label variables is accomplished by the function EntropyHeuLabels(), a greedy algorithm to maximize the SOE metric introduced in Section 6.5.1. The metric computation is listed in Algorithm 3. It is based on the estimation of the entropy of a multiset, which is represented by an MTMDD obtained by projecting the index $\mathcal{I}$ on a specific set of variables.

The label variable reordering algorithm is based on constructing a series of decision diagrams composed of an incremental number of variables. For such a reason, at line 14 a pointer to a Meddly forest is declared. In the next two lines, the unbounded variable set $\mathcal{U}$ and the partial order vector $\mathcal{O}$ are defined. The former is initialized as the whole set of available label variables

Algorithm 2 Variable ordering selection heuristic

```
1:  function EntropyHeu(I)
2:      O ← EntropyHeuLabels(I)                          ▷ Build partial ordering
3:      O* ← getCurrentVariableOrdering(I)
4:      minSize ← getNodesEdgesSum(I)
5:      for i ← [1 ... n] do
6:          O' ← InsertAt(O, {v_n ↦ i})                  ▷ Insert x_liv at index i
7:          I' ← reorderDDVariables(I, O')
8:          size ← getNodesEdgesSum(I')
9:          if size < minSize then
10:             minSize ← size
11:             O* ← O'
12:     return O*
13: function EntropyHeuLabels(I)
14:     Let F ← null be a pointer to a Meddly forest
15:     U ← {v_1 ... v_{n-1}}
16:     O ← {}
17:     while |U| > 1 do
18:         n_v ← initForest(I, F, O)
19:         initState(null, −∞)
20:         for v' ∈ U do:
21:             I' ← projection(F, O + v', I)
22:             H' ← computeEntropy(I')
23:             if H' > H_sel then
24:                 updateState(v', H')
25:         v_sel ← getSelectedVariable()
26:         U ← U \ {v_sel}
27:         O ← append(O, v_sel)
28:         clearForest(F)
29:     Append the last variable of U in O
30:     return O
```

within the index MTMDD $\mathcal{I}$, whose number is equal to the maximum feature length used to build the index, whilst the latter is initialized as empty.

The loop in lines 17-28 chooses at each iteration the unbounded variable that, jointly with the current partial order maximises the entropy estimation. The iteration starts by initializing a Meddly forest defined on a domain of $n_v = |O| + 1$ variables. Then, the variables to perform the greedy choice are initialized and the inner loop in lines 20-24 iterates over the unbounded

variables in $v' \in \mathcal{U}$. At each inner iteration, the function computeMetric() estimates the entropy-based metric on the decision diagrams built by considering the variables currently in the partial order $\mathcal{O}$ and the current unbounded variable v', and such values (i.e. the entropy value and the selected variable v') are saved if the current entropy value greedily improves the current results. Once the inner loop terminates, the selected variable v_{sel} is removed from the set of unbounded variables and it is added to the partial order $\mathcal{O}$. Finally, the forest pointer is freed in the last line of the outer loop.

The computeMetric() function performs the entropy estimation for a given set of variables. It takes as input (i) the index MTMDD defined over $l + 1$ variables, which l are label variables, (ii) a Meddly forest defined on $k < l$ label variables, and (iii) a set of k label variables. To actually perform the entropy estimation, the index has to be projected over the provided set of k variables. To do so, at line 3, a new decision diagram dd is created from the current forest. Similarly to the indexing phase, a buffer $\mathcal{B}$ is used to accumulate the tuples pending to be inserted in dd. Tuples are represented by buffer slots, which consist of arrays of length k.

The projection of the index $\mathcal{I}$ over the provided variables is performed by traversing $\mathcal{I}$ using a Meddly enumerator, and by saving the assignments of the variables of interest in the buffer slots. At line 7, tuples are pushed in the buffer: the y value is set to one in order to count how many times each tuple occurs. Whenever the buffer is filled, it is flushed in the current DD by means of the flushToDD() function, which has been previously defined in Algorithm 8.2. Once the construction phase has been completed, the decision diagram obtained in such a way is exploited by the computeEntropy() function to compute the Equation 6.2. It is worth noting that the quantity returned by such a function is always negative, because of the missing minus sign, in order to treat the problem as a maximization one.

Placing the location information variable

Once the partial ordering of label variables has been identified by the entropy-guided heuristic, a simple brute-force method is applied to identify the best position for the location information variable. Given the ordered set of the l label variables $\mathcal{O}$, the EntropyHeu() function empirically evaluates the memory occupation of the index $\mathcal{I}$ for the $l + 1$ possible variables orderings.

Algorithm 3 Algorithm for entropy estimation of target MTMDD

```
1: function PROJECTION(I, F, V)
2:     Let B = (B_x, B_y) be a buffer
3:     D ← NEWEDGE(F)
4:     e ← GETMEDDLYENUMERATOR(I)
5:     repeat
6:         prjtuple ← GETVARASSIGNMENT(e) \ V
7:         INSERT(B, prjtuple, 1)
8:         if ISFULL(B) then
9:             FLUSHTODD(B, D)
10:    until ISVALID(e)
11:    FLUSHTODD(B, D)
12:    return D

13: function COMPUTEENTROPY(D)
14:    Let c and den be integer variables initialized to zero
15:    c ← APPLY(CARDINALITY, D)
16:    v : ARRAY(c)                          ▷ Allocate c-size array
17:    e ← GETMEDDLYENUMERATOR(D)
18:    for i ∈ [0, ..., c) do
19:        v[i] ← GETTERMINAL(e)
20:        den ← den + v[i]
21:        NEXTELEMENT(e)
22:    return Σ_{t∈v} (t/den) log_2(t/den)
```

The location information variable is inserted at each available position of the ordering $\mathcal{O}$.

Given a ordering $\mathcal{O}$, at line 7 the index's variables are reordered with respect to $\mathcal{O}$, and the size of $\mathcal{I}$ can be estimated as the sum of the number of nodes and edges composing that DD. Internally, the variables are reordered by calling the method $\mathcal{F}$.REORDERVARIABLES($\mathcal{O}$) imposes the ordering $\mathcal{O}$ on forest $\mathcal{F}$, and then, on all the decision diagrams within forest $\mathcal{F}$. At the end, the algorithm returns the variable ordering associated with minimum size, among the tested ones.

Chapter 9

Applications

In this chapter, we show different case studies successfully investigated using either FeatSEE or GrapesDD tools.

In detail, the first three sections describe different omics-related real-world case studies we faced exploiting personalized workflow of analysis composed of FeatSEE modules. Differently, the fourth section describes the experimental results obtained with the GrapesDD tool involving the indexing, filter-and-verification phases, and the variable reordering heuristic.

In detail, experiments involving the FeatSEE framework prove the

9.1 Biomarker discovery for colorectal cancer

Fecal tests currently used for colorectal cancer (CRC) screening show limited accuracy in detecting early tumors or precancerous lesions. In this respect, we comprehensively evaluated stool microRNA (miRNA) profiles as biomarkers for noninvasive CRC diagnosis. A total of 1273 small RNA sequencing experiments were performed in multiple biospecimens. In a cross-sectional study, miRNA profiles were investigated in fecal samples from an Italian and a Czech cohort (155 CRCs, 87 adenomas, 96 other intestinal diseases, 141 colonoscopy-negative controls). A predictive miRNA signature for cancer detection was defined by a machine learning strategy and tested in additional

fecal samples from 141 CRC patients and 80 healthy volunteers. miRNA profiles were compared with those of 132 tumors/adenomas paired with adjacent mucosa, 210 plasma extracellular vesicle samples, and 185 fecal immunochemical test leftover samples. Twenty-five miRNAs showed altered levels in the stool of CRC patients in both cohorts (adjusted $P < .05$). A 5-miRNA signature, including miR-149-3p, miR-607-5p, miR-1246, miR-4488, and miR-6777-5p, distinguished patients from control individuals (area under the curve [AUC], 0.86; 95% confidence interval [CI], 0.79–0.94) and was validated in an independent cohort (AUC, 0.96; 95% CI, 0.92–1.00). The signature classified control individuals from patients with low-/high-stage tumors and advanced adenomas (AUC, 0.82; 95% CI, 0.71–0.97). Tissue miRNA profiles mirrored those of stool samples, and fecal profiles of different gastrointestinal diseases highlighted miRNAs specifically dysregulated in CRC. miRNA profiles in fecal immunochemical test leftover samples showed good correlation with those of stool collected in preservative buffer, and their alterations could be detected in adenoma or CRC patients. Our comprehensive fecal miRNome analysis identified a signature accurately discriminating cancer aimed at improving noninvasive diagnosis and screening strategies.

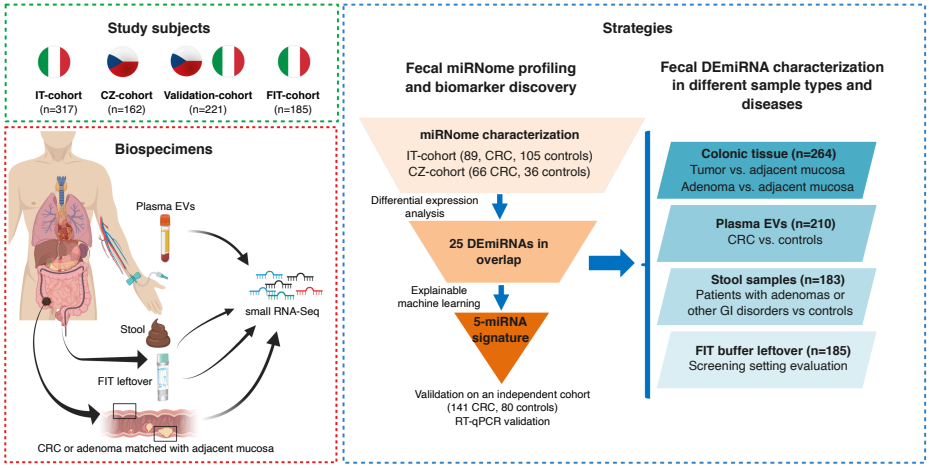


Figure 9.1: Representation of the study design.

9.1.1 Introduction

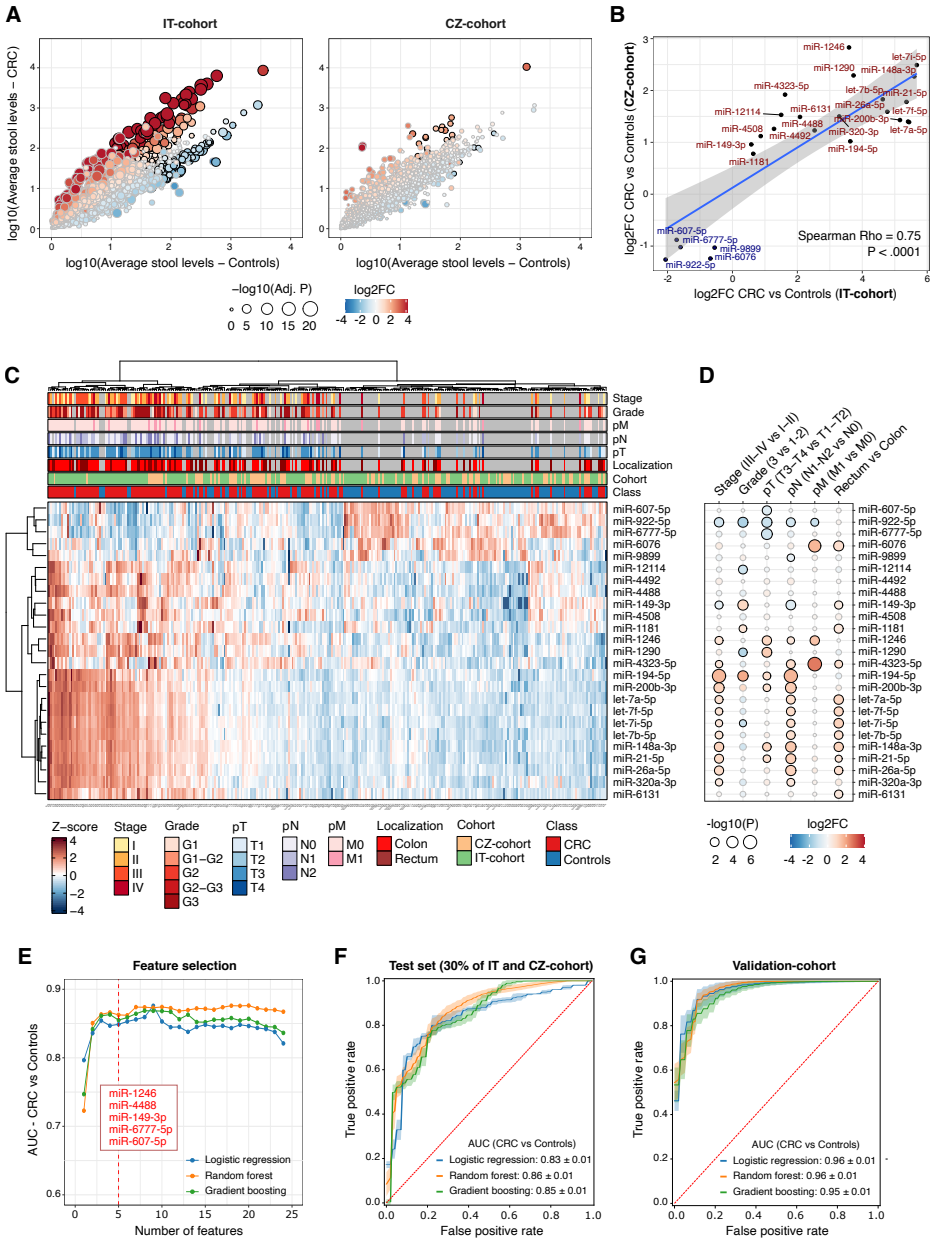
In the last 30 years, we have witnessed a dramatic increase in understanding the epidemiology, etiology, molecular biology, and various clinical aspects

of colorectal cancer (CRC) [142]. However, approximately 1.8 million new cases are annually diagnosed worldwide, posing CRC as the third most common incident cancer. Moreover, although early-stage tumors can be efficiently treated, CRC is still the second-leading cause of cancer-related death, with 900,000 deaths in 2018 [143, 144]. Hence, the early detection of preclinical cancers or precursor lesions is a desirable objective, because it may strongly increase the chances for successful treatment and cure. Most European countries have implemented CRC screening programs based on noninvasive stool tests for detecting fecal occult blood, mainly the fecal immunochemical test (FIT) [145, 146]. FIT selects individuals showing a higher prevalence of CRC and advanced benign neoplasia but has limited sensitivity to recognize advanced colorectal adenomas (AAs) [147]. Colonoscopy is also used in an opportunistic screening setting and detects both cancer and premalignant lesions but is bothersome and invasive, as well as costly for the health system [148]. Despite the fact that FIT-based screening programs are undeniably efficient in detecting premalignant growths and providing an earlier diagnosis, successfully reducing CRC burden, only approximately 5% of individuals who receive a colonoscopy based on FIT results will end up with a significant lesion (CRC or AA). Stool tests show a relatively low specificity, resulting in a high number of false positives and a considerable number of unnecessary colonoscopies [149]. Complementing traditional screening stool tests with other noninvasively detectable fecal molecular biomarkers could improve the triage of individual for colonoscopy, reducing the costs for the health systems in terms of the number of examinations and decreasing the risks and discomfort for patients [150, 151]. Identifying reliable biomarkers is not trivial, given the ensemble of hidden interactions between molecules and patient-specific clinical/anamnestic characteristics. However, machine learning (ML) algorithms have been defined to reveal significant features able to accurately discriminate groups of individuals. In particular, explainable ML approaches allow the identification of novel molecular biomarker signatures to improve early CRC diagnosis, as recently demonstrated for fecal microbial species [152] and urinary proteins [153]. The analysis of small noncoding RNAs in fecal samples has attracted interest with an excellent biological and analytic rationale for its application in large-scale clinical investigations [154]. Tumor-secreted small noncoding RNAs are directly and continuously released into the intestinal lumen, and their profiles may be altered in concomitance with the presence of

CRC and precancerous lesions. Moreover, small noncoding RNAs, such as microRNAs (miRNAs), are remarkably stable, enabling their accurate detection in stool without the need for special stabilization or logistic requirements [155]. miRNAs are suitable biomarkers in surrogate tissues and biofluids because their levels are altered in specific pathologic states [156], in the presence of precursor lesions [157], and in CRC development [158, 159, 160]. In addition, specific fecal miRNA alterations have been associated with the gut microbiome composition [161] and proposed as noninvasive CRC biomarkers [162]. So far, comprehensive miRNA profiling by small RNA sequencing (small RNA-seq) has been mainly performed on tumor tissues or plasma [162, 163]. In contrast, studies on fecal samples investigated few miRNAs in relation to CRC, typically in small cohorts and without taking into account their demographic characteristics [164]. In this respect, studies on the whole fecal miRNome showed that different lifestyles and dietary habits might critically affect specific miRNA levels [165, 166]. In addition, limited evidence is available on stool miRNA profiles in relation to patient clinicopathologic characteristics, such as specific CRC stages, precancerous lesions or other gastrointestinal (GI) diseases, except for the reported pleiotropic dysregulation of miR-21-5p in several diseases [167]. Therefore, a miRNA signature for CRC detection derived from a comprehensive fecal miRNome analysis across multiple populations is currently lacking. This multicenter study aimed to explore, by deep sequencing, the miRNA profiles in stool samples that best characterize CRC patients from control individuals and distinguish colorectal adenomas or other GI diseases. The analyses were performed in different independent cohorts adopting the same protocol for participant recruitment, sample collection, and small RNA-seq experiments/analyses. An integrated explainable ML strategy identified a fecal miRNA signature distinguishing CRC patients from control individuals, and the results were validated in an additional cohort. Finally, altered miRNAs in stool were also investigated in FIT-positive leftover samples collected within a population-based CRC screening program.

9.1.2 Material and methods

Stool Study Cohorts



(Caption on next page.)

Italian cohort Stool specimens and clinical and demographic data were collected from 317 subjects recruited in a hospital-based study at Vercelli, Italy.

Figure 9.2: (A) Scatterplot reporting the stool miRNA average levels in CRC patients (y-axis) or control individuals (x-axis) from the IT cohort (left) or CZ cohort (right). The dot color represents the log2 fold change (log2FC) from the differential expression analyses between CRC and healthy individuals, and the size is proportional to the age, sex, and multiple-testing adjusted P values. (B) Scatterplot reporting the correlations of log2FC of the 25 DEmiRNAs from the comparison between CRC and control individuals and in common between the IT cohort (x-axis) and the CZ cohort (y-axis). The up-regulated and downregulated miRNAs are reported in red and blue, respectively. (C) Heatmap of stool DEmiRNA levels in CRC and control individuals of both cohorts. For each participant, the CRC stage and grade based on the American Joint Committee on Cancer system, presence of metastasis, lymph node invasion status (pN), tumor size (pT), tumor localization, cohort of origin, and disease status (CRC or control) are reported. (D) DEmiRNA levels comparing CRC patients stratified for clinical data. The dot color represents the log2FC, and the dot size is proportional to the statistical significance. Black borders represent tests with $P < .05$. (E) Line plot reporting the ability of different combinations of feature selection methods and classifiers to perform the classification of CRC and control individuals. Each dot represents an AUC obtained using a different number of fecal DEmiRNAs in input. (F) Receiver operating characteristic curves obtained for the classification of CRC and control individuals using the identified miRNA signature. Data are reported for the 30% of participants excluded from the training set (left) and for the validation cohort (right). Adj., adjusted.

Based on results of complete colonoscopy examination, participants were classified into (i) 89 sporadic CRC patients; (ii) 74 polyp patients (6 hyperplastic polyps, 20 non-advanced adenomas (nAA), and 48 AA; serrated lesions were excluded since too few); (iii) 49 subjects with a GI disease (6 Crohn's disease, 9 ulcerative colitis, 14 diverticulitis, 7 diverticulosis, 13 hemorrhoidal disease); and (iv) 105 colonoscopy-negative control subjects. AAs were defined based on the presence of high-grade dysplasia, villous component, or lesion size > 1 cm as defined by [168]. Of this cohort, 93 stool samples (from 29 CRC patients, 27 polyps, 13 subjects with a GI disease, and 24 colonoscopy-negative controls) have been employed and described previously in other studies [169, 170, 171].

Czech cohort Stool specimens and clinical and demographic data were collected from 162 Czech individuals recruited from two hospitals in Prague and one in Plzen, Czech Republic.Based on colonoscopy results, subjects were divided in (i) 66 CRC patients, (ii) 28 polyp patients (9 hyperplastic polyps, 13

nAA, 6 AA; no serrated lesions were collected); (iii) 32 patients with other GI diseases (3 Crohn's disease, 11 ulcerative colitis, 17 diverticulosis, 1 unclassified inflammatory bowel disease, IBD); and (iv) 36 colonoscopy-negative subjects.

Validation cohort Stool specimens from 141 CRC patients recruited in a hospital in Brno, Czech Republic and 80 stool samples of healthy volunteers contributing to science were included. These subjects were previously described in other works: the CRC population is described in [172], but here sequenced for the first time for small RNA-seq; healthy volunteers are a part of the cohorts described and sequenced for small non-coding RNAs in [165, 166].

Fecal immunochemical test cohort FIT buffer leftover samples from 185 subjects with a positive test were collected within the CRC screening for the Piedmont Region (Italy). Based on colonoscopy results, subjects were classified as controls (n=53), AA (n=80), nAA (n=30), or CRC (n=22). Among them, 57 subjects also provided stool samples before undergoing colonoscopy.

Other Analyzed Biospecimens

For 132 patients having surgery at the Vercelli hospital, primary tissues (102 CRC and 30 adenomas) paired with adjacent colonic mucosa were collected. Blood samples were collected from 210 out of 317 Italian (IT) cohort participants, stratified into patients with CRC (n = 52), AAs (n = 19), nAAs (n = 15), hyperplastic polyps (n = 6), and other GI diseases (n = 39), and control individuals (n = 79).

Sample Collection

Naturally evacuated fecal samples were obtained from participants previously instructed to self-collect the specimen at home. Samples were collected in nucleic acid collection and transport tubes with RNA stabilizing solution (Norgen Biotek Corp). Stool aliquots (200 μL) were stored at –80C until RNA extraction [173]. For the validation cohort of CRC patients from Brno, stool samples were collected from untreated patients before the scheduled surgery with DNA-free swabs (Deltalab). Patients performed the collection at home and returned the samples to the hospital, where they were immediately frozen at –80C until further processing.

For the FIT cohort, leftovers from FIT tubes (w1.2 mL) used for automated tests (OC-sensor, Eiken Chemical Co) for hemoglobin quantification were stored at –80C until use. Plasma samples were obtained from 8 mL of blood centrifuged for 10 minutes at 1000 revolutions/minute, and aliquots were stored at –80C until use. Plasma extracellular vesicles (EVs) were precipitated from 200 mL of plasma using ExoQuick (System Biosciences) according to Sabo et al.32 Paired tumor/adenoma tissue and adjacent nonmalignant mucosa (at least 20 cm distant) were obtained from CRC and adenoma patients during surgical resection and immediately immersed in RNAlater solution (Ambion). All samples were stored at –80C until use.

Total RNA Extraction, Small RNA-Sequencing Library Preparation, and Quantitative Real-Time Polymerase Chain Reaction

Total RNA from stool and FIT leftover samples was extracted using the Stool Total RNA Purification Kit (Norgen Biotek Corp) as previously described.20 Total RNA from plasma EVs was extracted as described in Sabo et al.32 For tissue samples, total RNA was extracted using QIAzol (Qiagen) according to the manufacturer's instructions. Small RNAs were converted into barcoded complementary DNA libraries for Illumina single-end sequencing (75 cycles on HiSeq4000 or NextSeq500, Illumina Inc) as previously described.24 Candidate miRNA biomarkers were replicated in stool samples using the miRCURY LNA miRNA PCR Assays (Qiagen). Reverse transcription (RT) was performed using the miRCURY LNA RT kit (Qiagen) according to the manufacturer's instructions. All reactions were run on an ABI Prism 7900 Sequence Detection System (Applied Biosystems). Analyses were performed as described by Moisoiu et al [174].

Computational and Statistical Analysis

Small RNA-seq analyses were performed as described by Tarallo et al. [173], considering a curated miRNA reference based on miRBase v22 and including a characterization of novel miRNAs. Differential expression analyses were performed using DESeq2 v1.22.2 [175]. Functional enrichment analysis was performed with RBiomirGS v0.2.12 [176], considering the validated miRNA-target interactions. A generalized linear model was defined by considering the

miRNA levels as the dependent variable and participant age, sex, body mass index (BMI), smoking habit, and cohort as independent variables. An ML strategy was implemented to identify the optimal fecal miRNA signature to accurately classify CRC patients from control individuals. The ML approach is composed of 3 phases: data preparation, feature selection, and classification. The signature was determined by considering an increasing number of miRNAs prioritized by filter and classifier-embedded methods applied to the training set (70% of the IT/Czech [CZ] cohorts). The optimal set of miRNAs providing the highest area under the curve (AUC) was selected and further tested by 100 stratified 10-fold cross-validations, first on the remaining 30% of the IT/CZ cohorts excluded from the training set and then on the validation cohort. The training and test sets were defined by a stratified selection to maintain the same proportion of participants characterized by specific covariates (i.e., age, sex, cohort, disease status, and tumor staging). Other statistical tests were performed using the WilcoxonMann-Whitney and Kruskal-Wallis (continuous variables) or chi-square (categorical variables) methods. The BenjaminiHochberg method was used for multiple-testing correction. Results were considered significant at $P < .05$.

Study Design

This study was designed to define and characterize a fecal miRNA signature that accurately distinguishes CRC patients from control individuals (Figure 9.1). The applied analysis strategy included the following phases.

Fecal miRNome profiling and biomarker discovery.

- Detection of stool miRNAs with altered levels in CRC: miRNA profiles from small RNA-seq and metadata were used for a differential expression analysis between CRC patients and control individuals of both the IT cohort and CZ cohort, independently. The overlapping differentially expressed miRNAs (DEmiRNAs) from both cohorts were the input of the next step.

- Feature selection and definition of an miRNA predictive signature: An ML strategy identified an miRNA signature composed of the minimal set of DEmiRNAs that better distinguished CRC patients from control individuals by a stratified cross-validation procedure.

Analysis Details							Precision		F1-score	
Comparison	**Test Set**	**AUC (mean±SD)**	**95% CI**	**Accuracy**	**Sensitivity**	**Specificity**	**Disease**	**Healthy**	**Disease**	**Healthy**
CRC vs Healthy	Discovery	0.86 ± 0.01	0.79–0.94	0.78	0.78	0.78	0.82	0.74	0.80	0.76
CRC vs Healthy	Validation	0.96 ± 0.01	0.92–1.00	0.89	0.90	0.88	0.93	0.83	0.91	0.85
CRC I-II vs Healthy	Discovery	0.86 ± 0.01	0.76–0.96	0.81	0.65	0.90	0.79	0.82	0.71	0.86
CRC I-II vs Healthy	Validation	0.95 ± 0.01	0.90–1.00	0.86	0.82	0.91	0.90	0.83	0.86	0.87
CRC III-IV vs Healthy	Discovery	0.88 ± 0.01	0.78–0.98	0.83	0.66	0.92	0.82	0.83	0.73	0.88
CRC III-IV vs Healthy	Validation	0.96 ± 0.01	0.91–1.00	0.85	0.75	0.94	0.91	0.82	0.82	0.88
CRC+AA vs Healthy	Discovery	0.84 ± 0.01	0.77–0.91	0.77	0.83	0.67	0.81	0.70	0.81	0.69
AA vs Healthy	Discovery	0.82 ± 0.01	0.71–0.97	0.79	0.61	0.86	0.62	0.85	0.62	0.85
AA+nAA vs Healthy	Discovery	0.77 ± 0.02	0.65–0.89	0.73	0.62	0.81	0.67	0.77	0.64	0.79
nAA vs Healthy	Discovery	0.80 ± 0.01	0.63–0.97	0.82	0.13	0.99	0.79	0.82	0.22	0.90
CRC vs AA	Discovery	0.68 ± 0.02	0.54–0.82	0.76	0.92	0.25	0.80	0.49	0.85	0.33

Table 9.1: Performance of the 5-miRNA Predictive Signature in the Different Comparisons. The analysis includes age and sex covariates. Thirty percent of samples were excluded from the training and matched by age, sex, cohort, and CRC stage.

- Validation of the miRNA predictive signature. The signature performance was estimated in the validation cohort by a stratified cross-validation procedure.

Fecal differentially expressed microRNA characterization in different sample types and diseases.

- Assessment of DEmiRNA profiles in different biospecimens and clinical situations: DEmiRNA levels were evaluated in (1) tumor/adenoma tissue and adjacent mucosa, (2) plasma EVs of CRC patients and control individual, and (3) fecal samples from patients with a GI disease or precancerous lesions to identify CRC-specific or commonly altered miRNAs. In particular, the miRNA signature from (1) was also tested in the discrimination of patients with precancerous lesions (AA or nAA), alone or in combination with CRC, from control individuals.
- Testing the DEmiRNA levels in samples from a CRC screening program: DEmiRNA profiles were explored in parallel in FIT buffer leftovers and in stool collected in tubes with RNA stabilizing solution. Subsequently, stool DEmiRNA levels were analyzed in the leftover samples of the FIT cohort by stratifying participants based on the colonoscopy results.

9.1.3 Results

Stool miRNA profiles are altered in CRC patients: Evidence From Two European Populations

In agreement with previous studies [161, 165, 172], an average of 479 (range, 86–1516) miRNAs were detected in each stool sample by small RNA-seq.

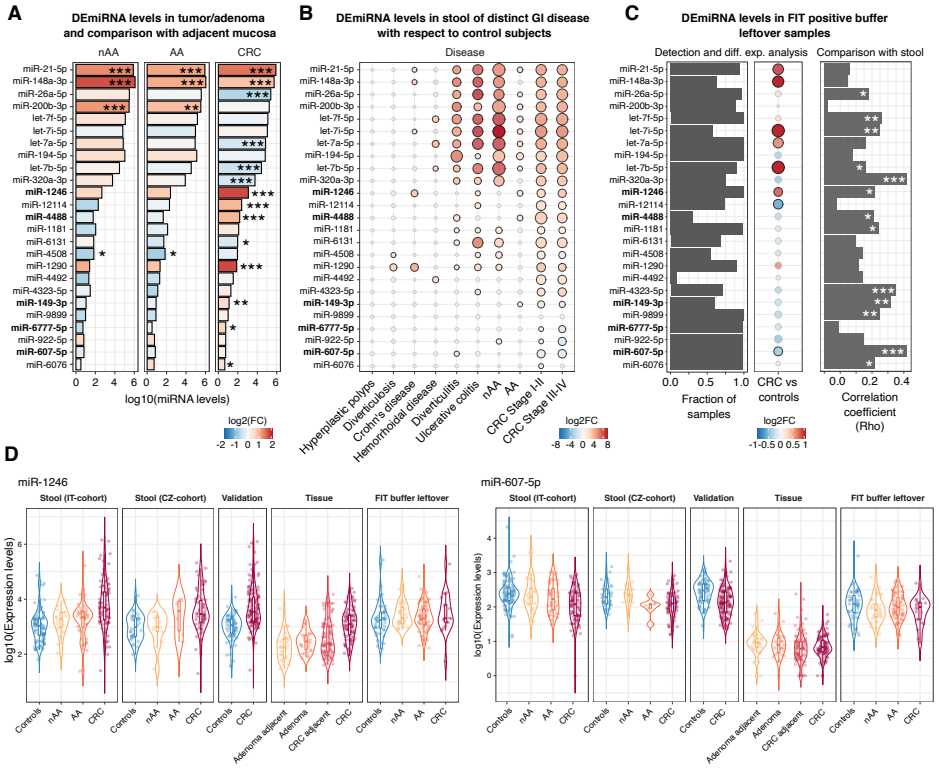


Figure 9.3: Characterization of the 25 fecal DEmiRNAs in different sample types. (A) Bar plot reporting the median levels in tumor, AA, and nAA tissues. The color code represents the log2 fold change (log2FC) from the paired differential expression analysis between CRC/adenoma tissues and matched adjacent mucosa. $***adjustedP < .001$, $**adjustedP < .01$, $*adjustedP < .05$. (B) Comparison of miRNA levels in the stool of patients with CRC, colorectal adenomas, hyperplastic polyps, or other GI disorders with respect to control individuals. The dot color represents the log2FC, and the dot size is proportional to the analysis significance. Black borders represent results with an adjusted $P < .05$. (C) DEmiRNA analysis in FIT leftover samples from CRC screening. (Left) The fraction of FIT cohort samples in which each miRNA was detected and (center) results of the differential expression analysis between FIT-positive patients with CRC diagnosis based on colonoscopy outcome and those with a negative one. The dot color represents the log2FC, and the dot size is proportional to the analysis significance. Black borders represent a DESeq2 Benjamini-Hochberg $adjustedP < .05$. (Right) Correlation coefficients between miRNA levels in stool and FIT buffer leftover samples from the same individuals ($***P < .001$, $*P < .05$). (D) Box plots reporting miR-1246 and miR-607-5p levels in all study cohorts and biospecimens.

The age- and sex-adjusted differential expression analysis between CRC patients and control individuals was performed independently on both the IT cohort and CZ cohort identifying, respectively, 250 and 29 DEmiRNAs (median expression, $> 20 reads$; adjusted $P < .05$) (Figure 9.1.2A). Twenty-five stool DEmiRNAs were in common between both cohorts (Figure 9.1.2B), all with a coherent expression trend (20 up- regulated and 5 down-regulated; $rho = 0.75$; $P < 0.001$) (Figure 9.1.2B). The alteration of these fecal miRNA levels in relation to CRC was further supported by a generalized linear model analysis adjusted for cohort, age, sex, BMI, and smoking habits: 22 out of the 25 DEmiRNAs remained significantly associated ($P < .05$). DEmiRNA profiles were further explored in relation to CRC patient clinical data (Figure 9.1.2C and D). The levels of 3 down-regulated miRNAs (miR-607-5p, miR-677-5p, and miR-922-5p) significantly decreased with increasing tumor size (Figure 9.1.2D). miR-922-5p also significantly decreased in patients with advanced disease stages or lymph node invasion (Figure 9.1.2D). Conversely, increasing levels of 19 out of the 20 up- regulated miRNAs in CRC were observed along with tumor size, with miR-1246, miR-1290, miR-148-3p, and miR-194-5p significantly related to this parameter. The levels of 11 CRC–up-regulated miRNAs significantly increased in patients with lymph node invasion. In addition, the levels of 11 miRNAs were significantly higher in samples from patients with rectal compared to colon cancers (Figure 9.1.2D). Functional analysis of DEmiRNA target genes showed their involvement in cancer-related processes, including cell cycle regulation and DNA repair, particularly for up-regulated miRNA targets.

A Fecal MicroRNA Signature Distinguishes Colorectal Cancer Patients From Control Individuals

An explainable ML strategy was implemented to identify the minimal set of miRNAs as a signature for CRC detection (Figure 9.4. The pipeline was applied on the 25 DEmiRNA profiles and considering 70% of the IT cohort and CZ cohort as the training set. The best miRNA signature distinguishing CRC patients from control individuals included miR-607-5p, miR-6777-5p, miR-4488, miR-149-3p, and miR-1246 (AUC, 0.87 ± 0.01) (Figure 9.1.2E). This set of 5 miRNAs represented the best combination of noncorrelated molecules with the highest discriminative power. Moreover, they showed a good performance in the classification of the 30% of participants excluded from the

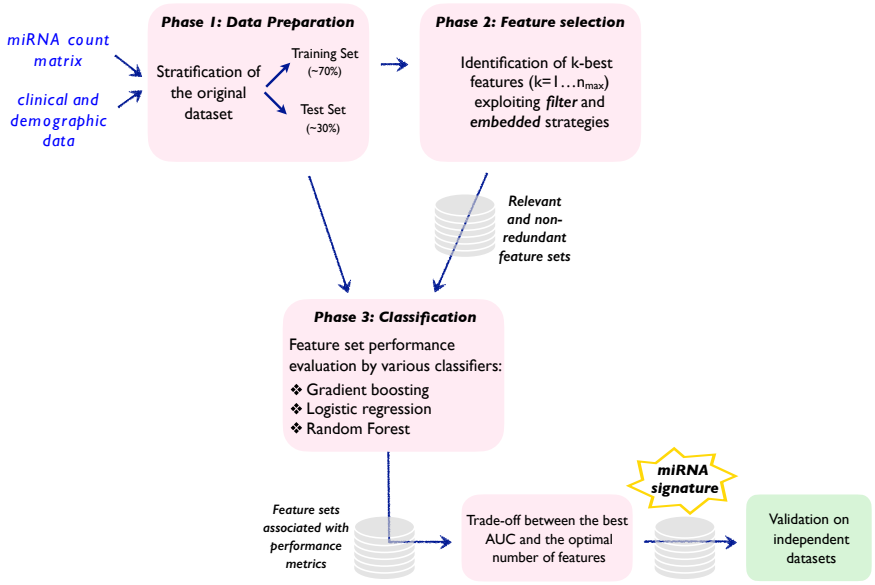


Figure 9.4: Schematic representation of the 3-phase explainable ML approach. An miRNA count matrix and the clinical/demographic data are the input data, and the best-performing miRNA signature is the output.

training set (AUC 0.81 ± 0.01) (Figure 9.1.2F). The classification improved after the inclusion of sex and age in the model (AUC 0.86 ± 0.01) (Table 9.1). The performance of the signature was again tested in the validation cohort, where it remained fairly similar, irrespective (AUC 0.91 ± 0.01) or not (AUC 0.96 ± 0.01) of age and sex (Figure 9.1.2F and Table 9.1). By stratifying patients for CRC stage, the same 5-miRNA signature accurately distinguished patients with stages III–IV CRC (validation cohort: AUC, 0.96 ± 0.01 and 0.94 ± 0.01, respectively, including or not age and sex), or CRC stages I–II from control individuals (validation cohort: AUC, 0.95 ± 0.01 and 0.87 ± 0.01, respectively, including or not age and sex) (Table 9.1). The panel of 5 miRNAs of the signature identified by sequencing was tested by RT quantitative polymerase chain reaction (qPCR) in RNA isolated from a subset of 96 stool samples equally distributed among IT and CZ cohort participants, with a balanced number of CRC patients and control individuals (Figure 9.5A). The 5 miRNAs were detected in all samples, also using this second method. The normalized levels from RT-qPCR showed patterns comparable to those provided by sequencing, except for miR-4488 (Figure 9.5A). In particular, miR-1246 and miR-149-3p levels were significantly increased in patient samples. The

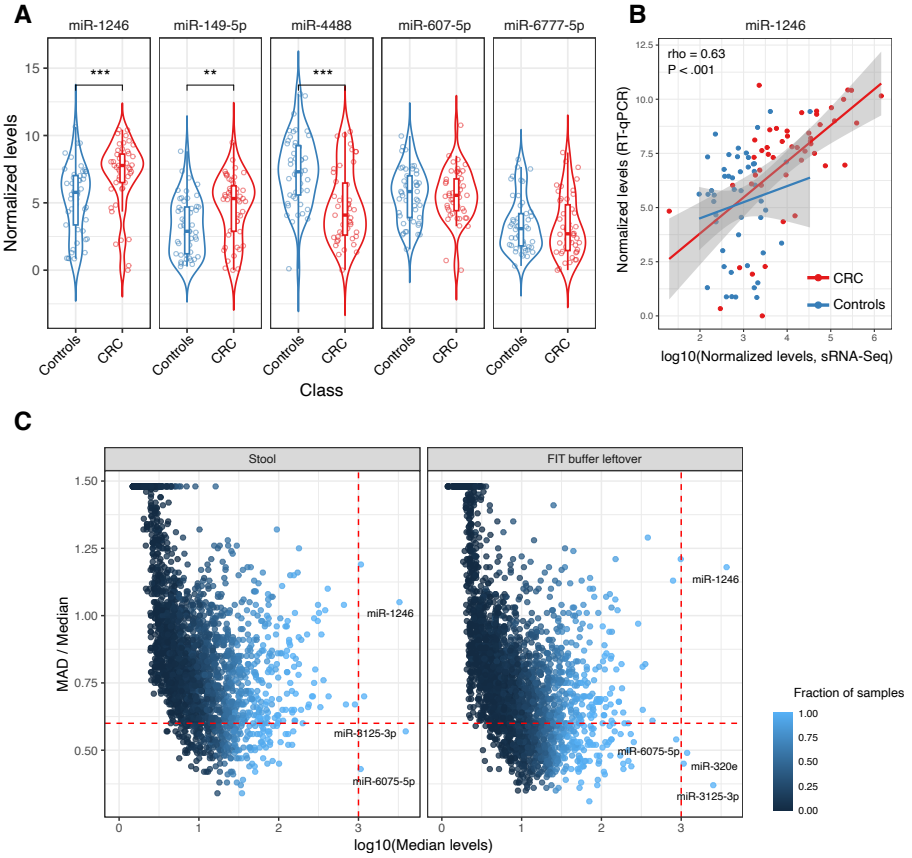


Figure 9.5: (A) Box plot showing the RT-qPCR normalized levels of the 5 miRNAs of the stool signature. P value by Wilcoxon rank sum test. $***P < .001$, $**P < .01$. (B) Scatterplot comparing the stool levels of miR-1246 measured by small RNA-seq (x-axis) and RT-qPCR (y-axis). The coefficient and significance of the Spearman correlation analysis is also reported. (C) Scatterplot reporting the median levels (x-axis) and the expression variability (as the ratio between median absolute deviation [MAD] and median, y-axis) of miRNAs measured in stool samples (left plot) or FIT buffer leftover (right plot) from the same subjects.

same method was used to test the 5 miRNA levels in RNA from 8 FIT left-over samples of participants with a positive FIT result at the CRC screening: all miRNAs were also detected in this biospecimen (data not shown). For 4 signature miRNAs, a concordant expression pattern was observed between small RNA-seq and RT-qPCR normalized levels, particularly for miR-1246 (rho = 0.63, $P < .001$) and miR-149-3p (rho = 0.26, $P < .05$) (Figure (Figure 9.5B).

Only the levels of miR-4488 were characterized by a negative correlation (rho = –0.48, $P < .001$) in CRC patients only.

Stool Differentially Expressed MicroRNA Profiles Mirror Those of Primary Colorectal Cancer and Adenoma Tissues

A paired differential expression analysis was performed between tumor tissues and matched adjacent mucosa collected from 102 CRC patients. Among the 25 stool DEmiRNAs, 14 were differentially expressed (adjusted $P < .05$) in this comparison (Figure 9.3A), with 7 miRNAs (miR-21-5p, miR-1246, miR1290, miR-148a-3p, miR-4488, miR-149-3p, miR-12114) up-regulated in tumor tissues coherently with their increase in CRC patient stool. Among them, 3 (miR-1246, miR-4488, miR-149-3p) were included in our miRNA signature. The 5 miRNAs significantly down-regulated in CRC patient stool (miR-607-5p, miR-6777-5p, included in the 5-miRNA signature; miR-6076; miR-922-5p; and miR-9899) were poorly expressed (normalized reads, < 20) in both tumor and adjacent tissues.The differential analysis performed on 30 adenoma tissues matched with adjacent mucosa showed miR-21-5p, miR-1290, miR-148a-3p, and miR-200b-3p as significantly up-regulated in adenoma tissues (adjusted $P < .001$), whereas let-7i-5p and miR-4508 were down-regulated (Figure 9.3A).

Few MicroRNA Levels Are Dysregulated in Circulating Extracellular Vesicles of Colorectal Cancer Patients

Small RNA-seq was performed on RNA isolated from plasma EVs collected from 210 participants in the IT cohort, detecting an average of 309 (range, 252–1213) miRNAs in these samples.Among the 25 DEmiRNAs identified in stool samples of CRC patients, both miR-1246 and miR-4488 emerged as coherently significantly dysregulated in plasma EVs, although the latter was associated with low levels (normalized reads < 20).Another miRNA (miR-150-5p) was differentially expressed between CRC patients and control individuals.

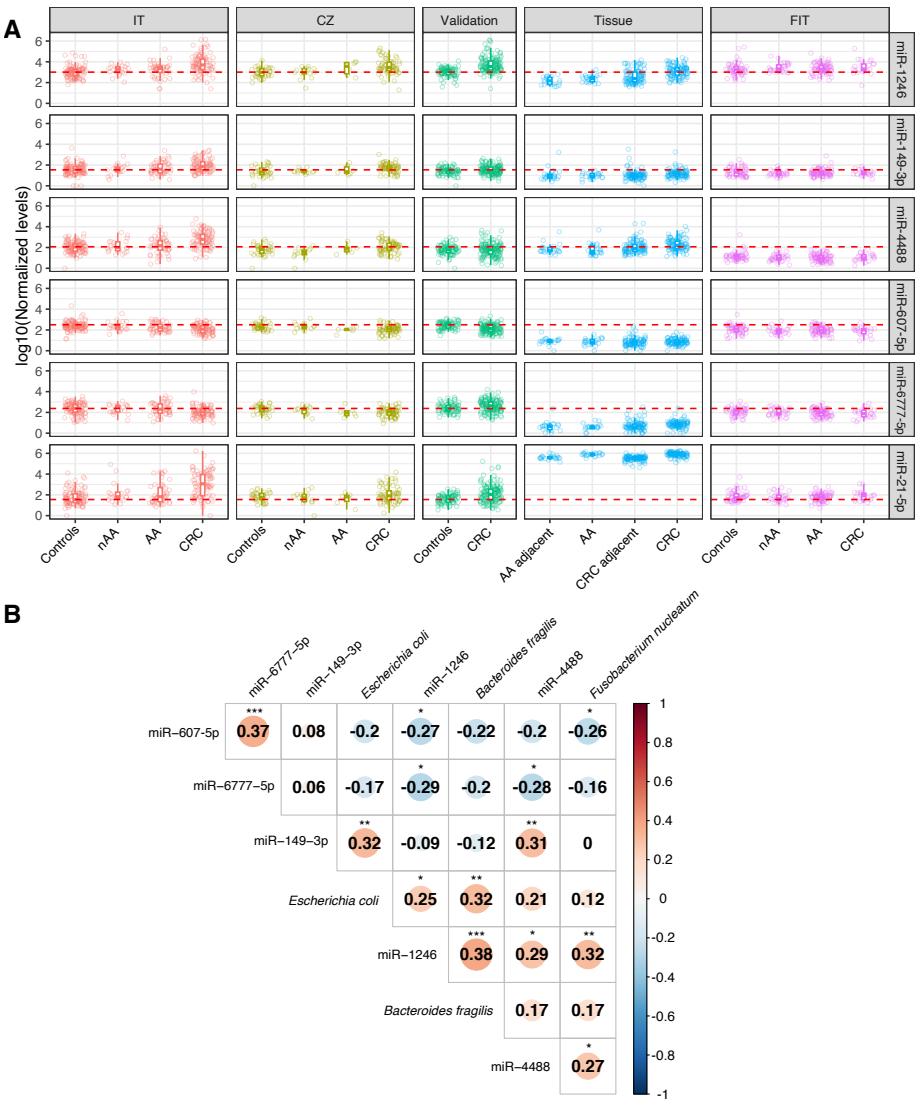


Figure 9.6: (A) Box plots reporting, for each study cohort, the normalized levels of the 5 stool miRNAs belonging to our CRC-predictive signature. At the bottom, the levels of miR-21-5p are also reported. The red dashed lines refer to the median miRNA level measured in control individuals of the IT cohort. (B) Correlation plot representing the results of the Spearman correlation analysis between the levels of the 5 fecal miRNAs and *F nucleatum*, *E coli*, and *B fragilis* abundances by the reanalysis of data from [169]. The size of the dot is proportional to the absolute correlation coefficient. $***P < .001$; $**P < .01$; $*P < .05$.

A Subset of Stool Differentially Expressed MicroRNAs Is Specifically Dysregulated in Colorectal Cancer Patients but Not in Those With Other GI Diseases

The CRC DEmiRNAs were further compared with those from patients with GI disorders and other precancerous lesions in both the IT and CZ cohorts. The age-, sex-, and cohort-adjusted differential expression analysis between each disease category and control individuals showed that the levels of 21 out of the 25 CRC DEmiRNAs were significantly altered in at least another GI disease (Figure 9.3B). Notably, in patients with ulcerative colitis, diverticulitis, nAA, or AA, 60% of the CRC DEmiRNAs were also dysregulated (Figure 9.3B). The lowest number of dysregulated miRNAs was observed in patients with Crohn's disease (2 miRNAs) or diverticulosis (5 miRNAs), whereas no DEmiRNAs were found in patients with hyperplastic polyps. Considering the 5 miRNAs constituting our predictive signature to distinguish CRC patients from control individuals, miR-6777-5p was not differentially expressed (compared to control individuals) in any other GI disease, miR-149-3p was significantly up-regulated only in patients with AA, and miR-607-5p was significantly down-regulated in patients with AA or ulcerative colitis compared to control individuals (Figure 9.3B).Conversely, miR-4488 and miR-1246 stool levels significantly increased in patients with diverticulosis, ulcerative colitis, diverticulitis, or AA, with the latter miRNA also increased in Crohn's disease patients. The identified signature was also used to classify AA and nAA patients from control individuals. Specifically, the miRNA signature was able to distinguish AA from control participants, both including (AUC, 0.82 ± 0.01) or not (AUC, 0.77 ± 0.02) age and sex in the analysis, as well as nAA (AUC, 0.80 ± 0.03 and 0.77 ± 0.02, respectively, including or not age and sex). Finally, patients with either CRC or AA were accurately distinguished from control individuals (including or not age and sex: AUC, 0.84 ± 0.01 and 0.81 ± 0.01, respectively) but not between them (CRC vs AA: AUC, 0.68 ± 0.02) (Table 9.1).

MicroRNAs Are Detectable in Fecal Immunochemical Test Leftover Samples by Small RNA Sequencing

The sequencing analysis was extended to 185 available leftover samples of the FIT cohort, still detecting an average of 618 miRNAs in each sample.All of the 25 stool DEmiRNAs were detected in this type of sample. Considering the

threshold adopted by our pipeline (i.e., a minimum of 20 reads), 4 (miR-607-5p, miR-1246, let7a-3p, miR-922) were detected in all samples, and 18 were detected in more than half (Figure 9.3C).Three miRNAs included in our signature (miR607-5p, miR-1246, miR-6777-5p) were detected in more than 95% of samples (Figure 9.3C), whereas miR-149-3p and miR-4488 were detected in 112 (57.4%) and 57 (30.8%) samples, respectively. Then, miRNA levels in FIT cohort samples were explored by stratifying participants according to the colonoscopy results. Comparing the levels of the 25 stool DEmiRNAs between 46 participants with a negative colonoscopy result (excluding 7 participants with high hemoglobin levels) and 22 patients with CRC, 8 (let-7a-5p, let-7i-5p, miR-148a-3p, let-7b-5p, miR-320a-3p, miR-12114, miR-21-5p, miR-607-5p) were significantly different (adjusted $P < .05$) (Figure 9.3C). Correlating the miRNA levels in FIT leftovers with the hemoglobin levels, only let-7b-5p showed a significant but limited correlation (rho = 0.16, P ¡ .05).Interestingly, miR-1246 and miR-607-5p were characterized, respectively, by increasing and decreasing levels, from colonoscopy-negative participants to CRC patients, as observed in the stool of the 3 case-control cohorts initially investigated for the miRNA signature identification (Figure 9.3D). Comparable miRNA expression levels and variability were observed between paired FIT leftover/stool samples from 57 individuals analyzed by small RNA-seq (rho = 0.70, Figure 9.3 and Figure 9.5C). Considering the levels of 468 miRNAs detected in at least half of FIT leftover samples, 99.6% were coherent with those in stool, with 282 miRNAs significantly correlated (average rho = 0.39, $P < .05$) (Figure 9.3C, Figure 9.5C)In both sample types, miR-3125-3p, miR-6075-5p, and miR1246 were characterized by the highest levels, and miR3125-3p was detected in all samples and associated with the lowest expression variability, in agreement with our previous findings in stool samples of 335 control individuals [166] (Figure 9.3A).The levels of all 25 stool DEmiRNAs positively correlated between the 2 specimens, with 13 of them reaching statistical significance (including miR-607-5p, miR1246, miR-149-3p, and miR-4488 from the 5-miRNA signature; $P < .05$) (Figure 9.3C). The 5-miRNA signature analyzed in FIT buffer leftovers was finally tested for the classification of patients with CRC from control individuals considering the signature alone or in combination with patient age, sex, and FIT hemoglobin levels. The 5-miRNA signature alone showed comparable classification performance (AUC, 0.85) as using age, sex, and hemoglobin levels (AUC, 0.87), and the combination of

both data provided the best classification results (AUC, 0.93).

9.1.4 Discussion

In the present study, to our knowledge, we performed the first large-scale profiling of the stool miRNome by deep sequencing of samples from patients with CRC, colorectal polyps, or other GI diseases and control individuals. Given the pervasive detection across multiple cohorts, we confirmed previous findings about fecal miRNA potential use as noninvasive molecular biomarkers [164] (Figure 9.3A). We also reported novel evidence on specific markers across different disease conditions. Notably, a fecal miRNA signature was able to accurately distinguish CRC patients from control individuals: both its ability to distinguish AA and its detection in FIT leftovers support future investigations for a use in CRC screening implementation.

In CRC patients, 25 fecal miRNAs emerged coherently altered in 2 independent cohorts. The profile of these miRNAs in stool reflected their altered expression in tumor tissue or adjacent colonic mucosa. More than half of such DEmiRNAs were already reported as altered in CRC, either in tissue or in various biofluids, including the up-regulated miR-21-5p, miR-148a-3p, miR-149-3p, miR-194-5p, miR200b-3p, and miR-320a-3p [164, 177]. Other miRNAs were associated with a disease for the first time by us; thus, further in vitro studies are needed to characterize the functional activity of these molecules and their involvement in CRC. Moreover, 3 DEmiRNAs identified in our study (miR-4323-5p, miR-607-5p, and miR-922-5p) are not currently annotated in the miRbase but were quantified based on the read mapping position within the miRNA hairpin. This is consistent with the need for continuous refinement of miRBase annotations [178] and with evidence of new miRNAs reported by different groups [179, 180].

Consistent with their overall higher/lower levels in the stool of CRC patients with respect to that of control individuals, the 25 DEmiRNA levels also increased/decreased with tumor size and stage. On the other hand, they were characterized by coherent altered levels when patients were stratified by tumor localization (proximal, distal, rectum). This further supports the importance of these miRNAs in relationship with the disease, as confirmed by the overrepresentation of cancer-related processes involving their validated target genes.

Based on this initial evidence, we implemented an integrated explainable ML strategy to explore, among the 25 DEmiRNAs, the minimal set of stool miRNAs able to accurately discriminate CRC patients from control individuals. Our approach generated a signature composed of 5 miRNAs (namely, miR-1246, miR-607-5p, miR-6777-5p, miR-4488, miR-149-3p) that was clinically validated in an additional independent cohort of cases compared to healthy volunteers and technically validated by another methodology (i.e., RTqPCR). The accurate discrimination of both participants in early and late cancer stages from control individuals confirmed the robustness of these 5 miRNAs for CRC detection. Although based on a small sample set, the signature could also accurately discriminate participants with AA from control individuals (AUC, 0.86), and in all analyses, high performances were obtained, irrespectively, by adjusting or not for sex and age, 2 relevant risk factors for this cancer [181]. To the best of our knowledge, this is the first signature based on fecal miRNAs whose efficiency was proven in populations from 2 countries characterized by different lifestyle and dietary habits [182] and CRC incidence [183]. Notably, such populations also show different trends in early-onset CRC [184], the incidence of which is linked to unhealthy individual habits, such as a sedentary lifestyle [185].

Similar to the functional analysis of all 25 DEmiRNAs, focused research on the 5-signature miRNA target genes evidenced a prevalence of genes involved in cancer-related processes, including regulation of the cell cycle, programmed cell death, and DNA damage response. Interestingly, functional analysis of predicted target genes of miR607-5p highlighted terms/processes related to nuclear cell cycle DNA replication and showed TRIM66, HIPK2, GRIN2B, and WTIP as the targets with the highest number of miR607-5p binding sites.

Among all the miRNAs of the signature, miR-1246 has been previously widely studied in CRC. Altered levels of this miRNA have been found in circulating exosomes in relation to cancer metastasis and prognosis [186, 187]. Exosomal miR-1246 levels were induced by *Fusobacterium nucleatum* in in vitro and in vivo CRC models with an increase of tumor cell metastatic potential [188]. These results align with more recent observations on the relationship between intratumor levels of *F nucleatum* and the aggressiveness of colon and breast cancers [189]. An intratumor increase in this well-known CRC-related bacteria might induce the release of exosomal miR-1246 in the

gut lumen, with the subsequent detection of this miRNA in stool samples. Similar considerations could be drawn from another study investigating a model of enterotoxigenic *Bacteroides fragilis* that induced up-regulation of exosomal miR-1246 in CRC cell lines [190]. Interestingly, in the same study, this microbial species reduced the exosomal levels of another fecal miRNA included in our signature, miR-149-3p, that was demonstrated to regulate tumor-infiltrating CD4 T-helper type 17 differentiation [190].

Similar findings were observed when analyzing the fecal miRNome and gut metagenome data from a previous study by our group in which we investigated the miRNAmicrobiota relationships in stool samples [173]. Specifically, by re-analyzing the data from that study, miR-1246 levels emerged as significantly related to both *F nucleatum* and *B fragilis* abundances, whereas miR-149-3p was inversely related to *B fragilis* abundances (Figure 9.6B).This pervasive relationship between in vitro exosomal miRNA levels and microbial infections suggests that the most informative stool biomarkers for CRC might reflect the dysregulated interactions between colonic tissue and the gut microbiota. Interestingly, in the miRNA-microbiota correlation analysis, 2 down-regulated fecal miRNAs (miR-607- 5p and miR-6777-5p), included in the predictive signature and so far scantly investigated in the literature, were inversely related not only to F nucleatum and B fragilis abundances but also to Escherichia coli, another species related to CRC onset [191] (Figure 9.6B)

To further explore the stool results, we tested DEmiRNA patterns in tumor and adenoma tissues paired with nonmalignant adjacent mucosa from patients of the IT cohort. Stool generally mirrored the altered miRNA expression levels of these tissues. Only the levels of miR-21-5p and miR-148a-3p increased in both CRC and adenoma compared to matched adjacent mucosa, whereas the other DEmiRNAs (including miR1246, miR-4488, and miR-149-3p of the signature) showed a CRC-specific dysregulation. miR-607-5p and miR-6777-5p, decreasing in patients' fecal samples, were characterized by low expression levels in both tumor/adenoma and adjacent mucosa, suggesting their deletion or epigenetic silencing. In The Cancer Genome Atlas [192], both miRNAs are frequently deleted in CRC,supporting the down-regulation in stool and tumor tissues observed by us. In agreement with our findings, previous studies have demonstrated that the down-regulation of miRNAs seems to be a premature step in the development of several cancers [193, 194]. Surprisingly, miR320a, let-7b-5p, and let-7a-3p, more abundant in stool of CRC patients,

were more expressed in adjacent mucosa than in tumor tissue. miR-320a has been widely reported as downregulated in CRC [195], whereas its circulating levels increased in relation to gut inflammation in IBD patients [196], coherent with our data in stool samples. Interestingly, miR-320a has been described as a key regulator of intestinal barrier formation [197]. Similarly, the expression of let-7 family members has been observed in the healthy gut epithelium, whereas their genetic depletion induced tumorigenesis in CRC mouse models [198]. Thus, the analysis of stool miRNAs is relevant to identify not only markers of the tumor small noncoding transcriptome but may also unveil an intestinal response of the stromal component to the presence of a tumor mass.

We also explored the miRNome of plasma EVs from a subset of the study population using the same experimental approach as in stool and tissue samples. However, in this circulating biospecimen, only a few miRNAs showed similar trends as in feces. For instance, among the miRNAs of the signature, miR-1246 and miR-4488 levels significantly increased in plasma EVs of CRC patients compared with control individuals. These results are consistent with previous findings reported by us, supporting stool miRNAs as more sensitive than plasma miRNAs in reflecting intestinal changes driven by a long-term dietary pattern [165]. Although more data are needed to compare the stool and plasma EV miRNome, given the reported relationships between miR-1246 levels in EVs and CRC metastasis [186], these circulating molecules may be more informative for advanced stages of the disease, which is beyond the scope of our investigation.

In this study, we sought to compare the stool DEmiRNA profiles of CRC patients with those of patients with other bowel inflammatory diseases of different severity confirmed by colonoscopy. Besides different polyp types, we included samples from several GI diseases, like different types of IBDs and diverticulitis. Notably, although the CRC-specific miRNAs were down-regulated, most of the altered miRNAs in common with adenomas and inflammatory diseases were up-regulated: miR-21-5p was the clearest example, confirming the literature [167]. As an exception, miR-607-5p was down-regulated in the stool miRNA profiles of patients with AA and ulcerative colitis. Accordingly, recent studies showed altered miRNA profiles in the fecal samples of patients with inflammation [199, 200], even in relation to microbiota [201]. We can therefore conclude that altered stool miRNA profiles reflect either the intestinal response to an inflammatory process or the transcriptional alterations related

specifically to CRC development. Importantly, we clearly demonstrated that well-known CRC-related miRNAs, such as miR-21-5p, show dysregulated fecal levels in several disease contexts, suggesting that other miRNAs, such as miR-6777-5p and miR-149-3p, should be investigated to design CRC-specific molecular signatures. This is the first evidence from a large-scale analysis of individuals with different gastrointestinal diseases of stool miRNAs specifically altered in CRC. It also highlights an extensive reflection of the gut inflammation on the fecal miRNA levels.

The fact that specific dysregulated fecal miRNAs could distinguish individuals with CRC or precursor lesions from control individuals and that, at least for cancer, data were confirmed in different cohorts, encouraging their use to complement the existing noninvasive screening tests. In this respect, we also investigated whether miRNAs can be detected in buffer-diluted stool leftovers from FIT tubes used in a context of a population-based screening program, and we found a remarkable similarity between the profiles detected in the stool samples collected in nucleic acid preservative medium tubes from the same participants. Despite data on a larger cohort being needed, this pilot small RNA-seq–based quantification of miRNAs in FIT buffer leftovers is consistent with previous evidence measuring miRNAs in this sample type by RT-qPCR [163], as well as by us. By exploring miRNA profiles within FIT-positive patients, we observed a subset of miRNAs differentially expressed between individuals with a positive or a negative colonoscopy outcome. In addition, miR-1246 and miR-607-5p from the 5-miRNA signature deserve further investigation because they were detected in most of the samples, and their levels respectively increased and decreased progressively, going from individuals with negative colonoscopy results, to those with adenomas of different severity, to CRC patients. Although these data confirm that miRNAs can be widely detected in FIT leftovers, the comparative results between individuals must be carefully considered given the small group size analyzed so far; the lack of samples from FIT-negative individuals; and the fact that we cannot rule out the role of confounding factors, including subclinical diseases in the colonoscopy-negative patients.

Most likely, by including hemoglobin levels evaluated by FIT, the discrimination capability of the present stool miRNA predictive signature would be further improved, as already reported in the past (FIT/FOBT + microbiome [152, 202], FIT + miRNAs [162], and FIT + methylation markers [203]).

The sensitivity and specificity of our 5-miRNA signature suggests that it could show a similar diagnostic performance as the multitarget stool DNA test [204] when used as a screening test in average-risk populations. Duran-Sanchon et al [162] proposed a 2-stool miRNA-based classification signature (namely, miR-27a-3p and miR421) combined with hemoglobin levels, age, and sex of FIT-positive individuals. The signature accurately classified CRC (AUC, 0.93) from control individuals but was less efficient when AA patients were included (AUC, 0.70) [203]. Different from us, the researchers initially selected miRNAs based on their differential expression between tumor tissue and adjacent mucosa and included in all models sex and age, 2 important risk factors for CRC. Hereby, we demonstrated the robustness of our signature because its performance remained similar even without the inclusion of age and sex covariates. In addition, despite the study not being designed for identifying stool biomarkers for adenomas, the 5-miRNA signature was able to accurately distinguish AA alone or in combination with CRC (AUC, 0.84), suggesting its use to detect precancer lesions at risk. In our study, miR-27a-3p and miR-421 were detected in tissue samples but not in stool, where only the former miRNA was measurable. In search of reproducible fecal molecular biomarkers for the noninvasive diagnosis of CRC and adenomas [152], a hypothesis-free miRNome-wide approach, such as the small RNA-seq analysis in stool performed in multiple independent populations, overcomes these issues.

The present study has several strengths: (1) the inclusion of independent cohorts from 2 countries with different diet and lifestyle habits as well as CRC rates; (2) the fact that the cohorts were different for CRC clinical characteristics, allowing the identification of accurate biomarkers independent of the disease stage; (3) the adoption of the same protocol for the collection of stool in both training cohorts; (4) the validation of the signature on a cohort with a different stool collection protocol, showing its robustness; (5) the miRNome-wide approach in different biospecimens and different GI disease contexts, which has allowed us to discriminate miRNAs specifically dysregulated in the stool of CRC patients; 6) the implementation of an explainable ML approach able to provide an unbiased method for identifying the minimal set of predictive biomarkers.

However, we are also aware of several limitations. Although there was a similar study design for recruitment, the 2 cohorts were heterogeneous for individual cancer categories. This heterogeneity could be responsible for the

observed differences in the median stool miRNA levels and expression differences between the 2 cohorts. Given the difference in the clinical characteristics of CRC patients, the main driver of such a difference may be the higher proportion of low-grade and low-stage tumors in the CZ cohort. However, the fact that the results are reproducible between cohorts further supports the robustness of the signature identified in this study.

Despite the large number of analyzed samples, the variegated spectrum of CRC, adenomas, and other precancerous lesions needs to be more exhaustively represented and deserves further investigation. For example, we did not investigate serrated lesions or deeply explore the alterations in CRC stratified based on molecular or clinical data. In addition, even though the observed DEmiRNAs were not reported to be modulated by dietary habits [165], the lack of dietary/lifestyle information of analyzed individuals may represent a limitation of the study. Follow-up studies with additional cohorts representing patients with different ethnicities, dietary patterns, and lifestyle habits are required, but this is beyond the scope of this study, which, to our knowledge, represents the largest sequencing-based analysis of stool miRNAs so far. In conclusion, this multicenter and international study based on small RNA-seq allowed us to comprehensively detect in stool several miRNAs differentially expressed in CRC. Furthermore, the implemented ML approach identified a minimal number of miRNAs whose combined profiles showed a good discriminating power for the presence of a tumor or AA, independent of age and sex. This may represent a fecal signature for improving the effectiveness of current noninvasive screening programs, potentially increasing sensitivity and maintaining high specificity, and applicable on a large scale, with a reasonable cost/time required.

In this respect, for FIT implementation, in the near future miRNA profiles will be investigated in additional cohorts, possibly from different countries, increasing the number/ types of precancer lesions and also including FIT-negative samples, with the chance to explore the role of diet and lifestyle habits on an adequate scale. Furthermore, the inclusion of FIT-negative samples will allow the possibility to prospectively test miRNA profiles in subsequent rounds of CRC screening, collecting multiple samples per individual. In parallel, the analysis of the microbiome composition of stool/ leftover FIT samples will help deepen the research on guthost crosstalk with small noncoding RNAs. Finally, even if small RNA-seq and RT-qPCR currently represent the most

commonly used approaches for miRNA analyses, we must consider that more rapid, practical, but reliable approaches, such as biosensors, may provide an alternative for testing the miRNA signature in a large clinical setting.

9.2 An application combining mechanistic and data-driven approaches

Biological phenomena are based on the precise and accurate cooperation of a non-random combination of molecules implicated in several pathways and networks. In the view of precision medicine, the plethora of omics data accrued sheds light on the comprehension of molecules cooperation. However, these data bring noise and redundancy that it is necessary to take into consideration during the data analysis. A combination of a mechanistic model on omics data and a multiphase feature selection is proposed. With this approach, we are able to select the relevant features as well as the potential functional role of them, and also the underlying biological mechanisms at the basis of the system under study.

9.2.1 Introduction

In recent years, multi-omics datasets are becoming increasingly common in biological research leading to obtaining an overview of the complex interactions beyond the biological systems. As a consequence, computational approaches for data integration become popular in order to extract hidden relationships insight from these several layers of information. We aim to deal with data analysis strategies to integrate diverse measures of biological information. Among the available strategies, the methodology should be chosen with respect to the biological question and the computational context used to translate enormous amounts of data and information into clinical and biological understanding. One limitation of these strategies is the explainability of the results obtained. Specifically, even though state-of-the-art algorithms are used in order to perform the data integration, a deep understanding from the biological functional point of view is not easy to obtain. To pursue this direction, we would like to integrate fluxomics and transcriptomics providing a more comprehensive understanding of cellular processes under clinical conditions.

Indeed, fluxomics corresponds to the global analysis of fluxes in the metabolic network of a cell, which can measure and evaluate the rates of reaction (i.e. fluxes) for a metabolic network. It represents an innovative omics research field, where the fluxome (the total set of fluxes in the metabolic network) depicts the integrative information on several cellular processes.

Its integration with transcriptomics can help identify the genes responsible for changes in metabolic fluxes and can be implemented by FBA, which is a mathematical modeling technique [205] that uses stoichiometric models of metabolic networks to predict metabolic fluxes (i.e., velocities at which metabolites are produced and consumed) under different environmental conditions by minimizing/maximizing an objective function (e.g., biomass growth). FBA can complement a novel workflow to integrate multi-omics data simultaneously or incorporate it with other FBA-derived tools [206].

In this way, integrating fluxomics data simulated from transcriptomic data can provide a more comprehensive picture of cellular metabolism and help identify the genes and pathways distinctive for a given condition.

In this work, we propose a new approach to characterize metabolic phenotypes from gene expression data. Specifically, we integrate the gene expression profiles in the FBA network and then the most influenced fluxes correlated to the increase of the cell biomass were identified by multiphase feature selection techniques. The application of our functional data integration approach was applied to the identification of glycolysis-associated clusters in colorectal cancer profiles [207]. The analysis supports cancer sample classification by metabolic phenotype, and results aimed to identify multi-omic profiles across cancer types with estimated variably in metabolic fluxes.

9.2.2 Material and methods

The data

Gene Expression profiles from 429 colorectal cancer (CRC) tissues were obtained from COAD cohort of The Cancer Genome Atlas. Data were retrieved using the TCGAbiolinks R package v2.28.1 considering the STAR-counts as the GDC workflow type. The CRC metabolic subtypes were obtained from [207]. We used the most comprehensive human Genome-scale metabolic model (GEM) to date; the community-curated Recon3D human metabolic reconstruction (8399 metabolites, 13543 reactions, 3698 genes) [208], to build tissue-

specific models.

Flux Balance Analysis

The first works [209] regarding the FBA date back to the early 1980s, and showed the possibility of deriving from a system of metabolic reactions the stoichiometric equations describing the relations among different products and biomass, and how to exploit linear programming for deriving the fluxes of such relations.
The principal aim of the FBA models is to simulate genome-scale reconstructions of metabolic networks that are characterized by a large number of reactions and metabolites by assuming the steady state of the system, which supposes that there is no net surplus or deficit of any metabolite. In particular, the fluxes distribution of the model is estimated considering inferior and superior boundaries for each flux, and an objective function to maximize, which plays a role as a surrogate for the most plausible physiological state among the states of the system. Mathematically, the FBA modeling m metabolites and r reactions can be translated as a linear programming problem (LPP) as follow:

$$\begin{aligned} &\text{maximize} && f(v) \\ &\text{subject to} && S \cdot v = 0 \\ &\text{and} && v_i^{Low} \leq v_i \leq v_i^{Up} \qquad \forall i = 1, \ldots, \ r \end{aligned} \tag{9.1}$$

where $v \in \mathbb{R}^r$ is the flux vector describing the activity of all the r reactions, $S \in \mathbb{Z}^{r \times m}$ is the stochiometric matrix, $f : \mathbb{R}^m \to \mathbb{R}$ is the objective function to maximize, $v_i^{Low} \in (-\infty, 0]$ and $v_i^{Up} \in [0, \infty)$ are respectively the known upper and lower bound of the rates for the $i - th$ flux.
Hereinafter, the integration of gene expression data can contribute to obtaining a biologically relevant and context-specific flux distribution so that the resulting model can predict active metabolic pathways from RNA-seq data and reconstruct a tissue-specific metabolic network. Specifically, RNA-Seq data were used to specify the activity of genes in each sample, and accordingly, reaction rates were quantitatively constrained to obtain flux distributions consistent with the available data and ascribe the impact of changes in gene expression to phenotypes according to [210]. This approach allows mapping the RNA-seq data for each reaction into a specific condition of the

model. First of all, each model's reaction i depends on the expression of a set of genes which can be represented by a logical formalism including gene names and boolean operators AND and OR. The formulation is then converted into expression level for a gene set of reaction i from the expression of individual genes following two scenarios:

- When two genes are connected with an AND operator, the gene set expression i for reaction i is the minimum of the individual gene making the gene set.
- When two genes are connected by an OR operator, the gene set expression i for reaction i is the sum of the individual gene making the gene set.

This gene-reaction map provides compatibility with a boolean on-off approach, as it approaches zero when the expression level approaches zero. Moreover, we defined the multiplicative factor for the flux bounds given the gene set expression GSE_i as follows:

$$v_i^{Low} h(GSE_i) \leq v_i \leq v_i^{Up} h(GSE_i), \tag{9.2}$$

where

$$h(GSE_i) = \begin{cases} (1 + |log(GSE_i)|)^{\frac{GSE_i - 1}{|GSE_i - 1|}} \text{if } GSE_i \in \mathbb{R}^+ \setminus \{1\} \\ 1 \text{ if } GSE_i = 1 \end{cases} \tag{9.3}$$

Based on the direct integration of the gene expression information into the flux bounds, we set fluxes to zero if the expression of their associated genes was low and the maximum allowable flux value as a function of measured gene expression. Finally, the metabolic model fluxes are units of $mmol/gDW * h$, where gDW is the dry weight of cell mass in grams, and h is the reaction time in hours. Human metabolism reconstruction Recon3D can be queried and downloaded from `http://bigg.ucsd.edu/` or `http://vmh.life`. Several models, including Recon3D, were initially deposited with default bounds set. Then, constraints were applied based on the recalculation of default bounds.

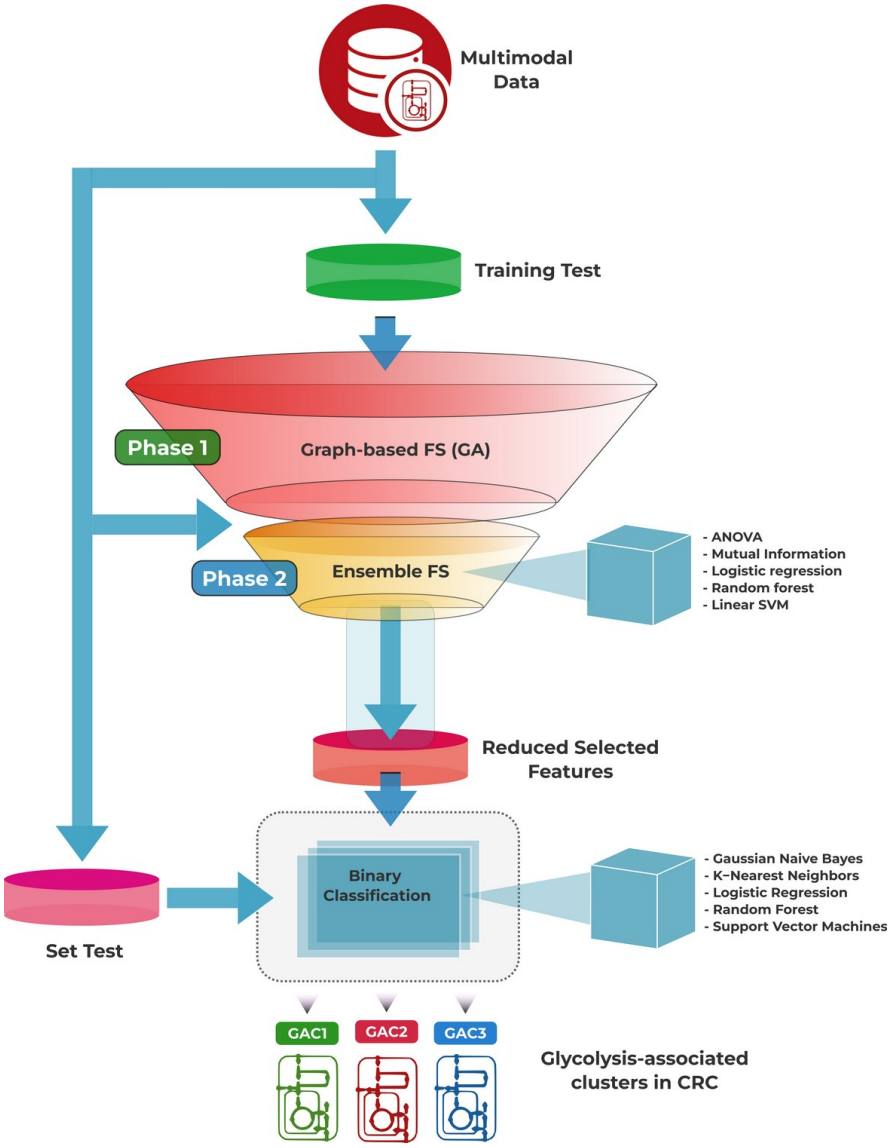


Figure 9.7: The workflow of analysis built exploiting FeatSEE end-to-end modules.

Data-driven approach

A graphical representation of the workflow exploited for the analysis is represented in Figure 9.7, and it is composed of four FeatSEE modules, namely data preparation, filter-based feature selection, ensemble feature selection and finally, model evaluation.

The first phase is devoted to data preparation, the fluxomics dataset is loaded

from files, and the labeled dataset is built given the labeling parameters. Then, it is split into training and test sets with a 70/30 proportion, and the resulting training test is used to build the feature graph, given the list of score functions to assign vertex weights, whereas the PCC is used to compute edge weights.
The second phase consists of a supervised dimensionality reduction exploiting the feature graph, to shrink the number of useful features by removing redundant and irrelevant ones.
The third phase consists of the ensemble feature selection exploiting multiple selectors, namely filter and embedded methods for feature ranking based on their relative importance.
The fourth phase evaluates the feature sets in a binary classification task by exploiting multiple learning models, which are fitted over the training set in a CV procedure. Performance metrics are estimated by evaluating classifiers' predictions both on (i) the CV fold not used for training and (ii) the independent test set.

9.2.3 Results

By combing the transcriptomics data and the FBA model, we would like to offer a functional explanation of the three glycolysis profiles identified in [211]. In detail, the glycolysis-associated clusters (GAC) identified distinct clinical genomic and tumor environment properties. Here, we focus our attention on the expansion of the description of the glycolysis profiles adding a new layer of stratification and classification. Then, for each CRC subject, was generated the corresponding FBA model mapping the transcription profile on the set of reaction boundaries (Eq. 9.2) leading to a personalization of the metabolic model. The reliability of the model was increased, and a fluxes distribution for each CRC subject by the maximization of biomass was obtained.

The flux profiles of each CRC subject were given as input for the multi-phase features selection methodology. Specifically, three binary classification tasks have been designed: GAC1 vs GAC2, GAC1 vs GAC3, and GAC2 vs GAC3, denoted as GAC12, GAC13, and GAC23 hereinafter. The resulting datasets have been split into training and test sets with a 70/30 proportion. For each comparison group, the GA is run three times using three different feature score functions (i.e. Pearson correlation coefficient, ANOVA F-test, and mutual information), and imposing 30 as the maximum number of selected features. Features sets selected by the GA are evaluated against the

test set exploiting a leave-one-out setting and six classification algorithms. For each comparison group, the sets associated with the highest Area Under the Curve (AUC) values in at least five out of six classifiers are merged into a single set that became the feature pool for the ensemble FS: 57, 84, and 87 features for GAC12, GAC13 and GAC23, respectively. The ensemble FS is applied to such pools to identify the proper trade-off between the number of features and the achieved performance (Figure 9.8). The resulting sets, which have no overlap among the features, contain 11 (GAC12), 7 (GAC13), and 9 (GAC23) GAC-related reactions that achieve an AUC performance on the test set of 0.98, 0.93, and 0.98, respectively.

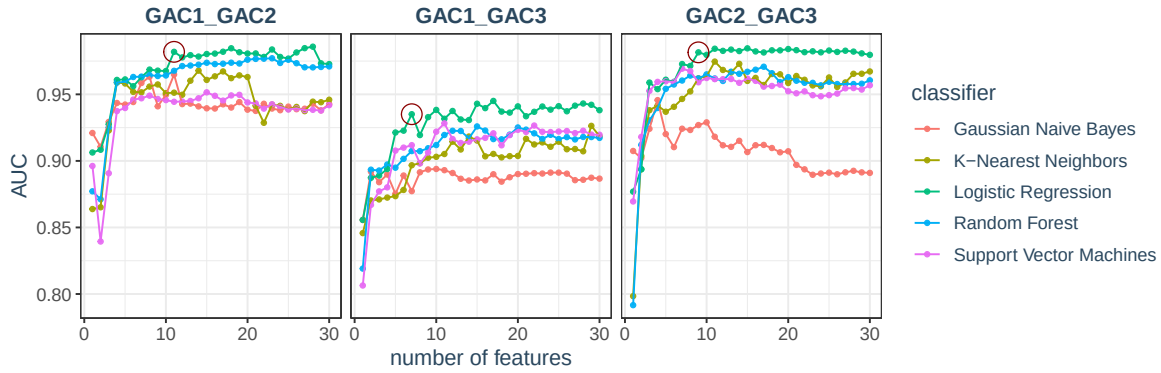


Figure 9.8: Performances on the test set in terms of AUC using the lists of features identified by the ensemble FS. For each comparison group (namely GAC1_GAC2 for GAC1 vs GAC2 and so on), we show the list arising the maximum AUC given the number of features desired and a classifier. The final sets are selected considering those with the minimum number of features associated with an $AUC_{set-selected}$ where $AUC_{max} - 0.01 < AUC_{set-selected} < AUC_{max}$

These GAC-related reactions are linked to 9 GAC1-, 8 GAC2-, 12 GAC3-related genes retrieved from the gene-reaction annotations incorporated by Recon3D. Through the KEGG pathway representation, we found pathways characterized by the network of enzyme–enzyme relations associated with the selected GAC-related genes. GAC-related metabolic pathways (Fig. 9.9A) were presented using AnnotationDbi which provides an interface for connecting and querying the KEGG annotation database. Additionally, for GAC-related genes signature we found their involvement in various metabolic subsystems (Fig. 9.9B). A metabolic subsystem corresponds to a set of reactions sharing a similar metabolic function. GAC1 attributes to pathways connected

Figure 9.9: Metabolic characterization of genes linked to the signatures identified by the ensemble FS within each comparison GACs groups (namely GAC1_GAC2 for GAC1 vs GAC2 and so on): (A) relative abundance of pathways annotated with KEGG modules, (B) relative abundance of reactions in various Recon3D metabolic subsystems.

to nucleotide metabolism. This characteristic was revealed by both annotation processes. GAC2 associates with the transport across the endoplasmic reticulum, the compartment reserved for the biogenesis of secretory proteins. Also, intermediate products of protein biogenesis as amino acids are connected to GAC2. GAC3 tended to be enriched with pathways related to steroid hormones and fatty acid metabolism.

9.2.4 Discussion

Functional data integration could be a new breakout in the bioinformatics approaches.

This work represents a first step in demonstrating how combining mechanistic models integrating omic-data with multiphase feature selection methodologies can deciphering the complexity at the basis of the biological systems. As proof of concept, we used our approach to integrate transcriptomics into human genome-scale mechanistic models comparing glycolysis-related molecular subtypes and identify the different metabolic markers of GACs through feature selection of fluxes followed by annotation.

Our workflow aims to provide sample-specific maps of cancer metabolism for enhanced data analysis planning. Notably, the ensemble feature selection

characterizes heterogeneity in the metabolism of GAC subtypes, returning non-overlapping feature sets and achieving an AUC performance on the test set of 0.98, 0.93, and 0.98 for (GAC12), (GAC13), and (GAC23). The importance of the hybrid approach is highlighted through our metabolic process characterization, which revealed that different GAC groups are associated with different metabolic pathways.

There are limitations to the results presented in this work. First and foremost, while Recon3D is the most updated human metabolic reconstruction, it has default bounds of (-1000, 1000). These default bounds could be problematic for reactions that were not constrained by gene expression profile integration, such as constraints representing the availability of metabolic sources in the environment. Indeed, there is a type of reaction in metabolic modelling called boundary reactions, which usually describe the exchange of metabolites between the internal cell and the external environment. Recon3D contains 13,543 reactions, 1,892 of which are devoid of gene-to-reaction associations because they are boundary reactions. If one uses default bounds to constrain reactions, there must be a check on the solution flux distribution to see if any of the fluxes are 1000 or -1000, and a warning should be given in such cases. We note that while the uniform model for metabolic sources in the environment was not as accurate, it could provide an initial prediction for metabolic characterization methods. Thus, the metabolic modelling framework was used as an initial prediction for metabolic fluxes, but more data is certainly needed to supply higher-quality data for feature selection. Sample-specific FBA best accuracy will be achieved through multi-omics integration FBA models. FBA has significant potential for embracing increasingly available metabolomic data [212]. Hence, advances in metabolomics and transcriptomics hold promise for synergistic outcomes by incorporating such data into the FBA framework. In future work, we also intend to study frameworks for estimating the sensitivity coefficients of reaction boundaries in constraint-based models.

9.3 Explaining early embryonic development via time-lapse features

9.3.1 Context

To date, an objective method to evaluate human embryo competence is still lacking, as conventional morphology is highly subjective, being performed using a wide range of grading systems affected by inter and intra-observer variability [213]. Indeed consistency and reproducibility of embryo evaluation are still not guaranteed among different in-vitro fertilization (IVF) laboratories [214]. In addition, standard grading only provides a brief view of embryonic morphology at a specific time point, while developmental changes over multiple time points may provide a more robust impression of embryo potential [215].

Quite recently, Time-Lapse Technology (TLT) has been introduced as a non-invasive method for real-time, dynamic observation of pre-implantation embryo development. Morphokinetic events can be monitored at the exact time of occurrence, providing new insights into several steps of in vitro growth and, ultimately, suggesting objective data with potential clinical relevance [216]. Some authors claimed time-lapse technology to improve embryo selection and IVF outcomes [215, 217, 218]. However, its efficacy is still a matter of debate [219, 220, 221, 222], being mainly ascribed to unperturbed culture conditions (temperature, atmosphere), rather than to the identification of reliable morphokinetic biomarkers of embryo competence [220, 223]. Indeed, the real-time observation of crucial events occurring during in-vitro growth has revealed a number of new parameters that have been associated to embryo development potential [216]. Looking forward, significant improvements of the technologies associated to TLT will drive a more detailed knowledge and understanding of the early developmental kinetics of human embryos [224]. In recent years, Artificial Intelligence (AI) has rapidly developed in various fields, including human embryology [225, 226]. Artificial intelligence (AI) may be used as a tool to assist embryologists in daily activities (e.g. morphological selection of embryos to transfer or cryopreserve), as it is able to analyse a huge number of heterogeneous data, such as those relating to embryo development in TLT systems [227, 228]. Machine Learning (ML) algorithms allow the identification of the most relevant variables included in a bulk set

of data, and may be used to develop complex prediction models of embryo growth [229, 230]. Among the above-mentioned models, no one was able to identify clear and clinically relevant cut-offs for morphodynamic variables to provide an objective tool to assist embryologists during embryo development assessment.

9.3.2 Material and methods

Data

The study was carried out in accordance with the Declaration of Helsinki and was authorized as a retrospective observational study by the local Ethical Committee (authorization number: 0056908). A signed informed consent was obtained from all patients.

The analysis included 575 embryos obtained from 80 women undergoing IVF and receiving the transfer in uterus of a single fresh blastocyst on day 5 between March 2018 and March 2020. These patients had mean age 35.3±3.5 years (range 25-42), body mass index 24.2±4.8 (range 18–25), ovarian reserve markers suggesting normal responsiveness to FSH stimulation (serum day 3 FSH ¡12 IU/l, antral follicle count 8-18, anti-mullerian hormone 2.5-4 ng/ml), and received the transfer in uterus of a single, fresh blastocyst on day 5. Patients with polycystic ovary syndrome (PCOS), ovarian endometriosis and/or unfavourable biomarkers leading to an expected poor/sub-optimal responsiveness to FSH [231] were excluded. Patients' clinical characteristics and variables related to IVF cycle were recorded, including the total dose of exogenous gonadotropins, the number of retrieved COCs (cumulus-oocyte complexes), the ovarian sensitivity index (OSI = retrieved COCs × 1000 /total gonadotropin dose) [232], the fertilization, cleavage, and blastocyst formation rates, the clinical pregnancy rate per embryo transfer (CPR/ET).

Embryos were cultured in the Geri plus ®TLS (Genea Biomed, Germany), that is equipped with an integrated embryo monitoring system to observe one zygote/microwell, as previously described [233]. The dish format allowed the observation of each embryo individually, even if all embryos shared a common $80 - \mu l$ medium drop. Up to day 3, embryos were cultured in pre-equilibrated Cleavage medium (Cook, Ireland) overlaid with mineral oil; then, a change of medium was performed, and the new one (Blastocyst medium, Cook, Ireland) was kept until the blastocyst stage. Bright-field images were captured by Geri

plus ®system every 5 minutes from the time of fertilization until the time of embryo transfer (ET) (day 5), cryopreservation (day 5 or 6), or discharge (day 5 or 6). Embryo morphological evaluation was first performed on day 2 using the Integrated Morphology Cleavage Score (IMCS) [234], and then repeated on day 5 and 6 according to standardized criteria [213]. All videos collected by Geri plus ®were analyzed by the same senior embryologist, and following ESHRE recommendations [216] the following morphokinetic parameters (times) were considered manually annotated: pronuclear appearance (tPNa), pronuclear fading (tPNf), completion of cleavage to two, three, four, and eight cells (t2, t3, t4, and t8, respectively), time intervals tPNf-tPNa, t2-tPNf, t3-t2, t4-t3, t4-t2, and t8-t4. Blastocyst formation was assessed at the same time interval (116±2 h post-insemination) for all embryos. Embryos reaching the expanded blastocyst stage on day 5 (score 3 according to [213] were included in the Blastocyst Group (BL Group, n = 210), whereas those arrested or progressed to a stage earlier than expanded blastocyst were included in the Not-expanded Blastocyst Group (nBL Group, n = 365), as previously described [235].

Features

A total of thirty variables among those currently recorded during clinical routine were considered for each embryo and divided into three categories: (i) woman-related (n=6): age, BMI, day 3 FSH, AMH, antral follicle count (AFC), years of infertility; (ii) COS-related (n=10): total exogenous FSH, peak E2, OSI, number of retrieved oocytes, number of mature oocytes, maturation rate, number of fertilized oocytes, fertilization rate, number of cleaved embryos, cleavage rate; (iii) embryo-related (n=14): insemination technique, IMCS score on day 2, tPNa (only for embryos obtained by ICSI), tPNf, t2, t3, t4, t8, tPNf-tPNa (only for embryos obtained by ICSI), t2-tPNf, t3-t2, t4-t3, t4-t2, t8-t4.

Machine learning workflow

The workflow of analysis is composed of four phases, namely (i) feature selection, (ii) rules extraction, (iii) rules selection and, (iv) rules evaluation, that have been implemented by applying the FeatSEE modules of feature selection and explainable feature extraction.

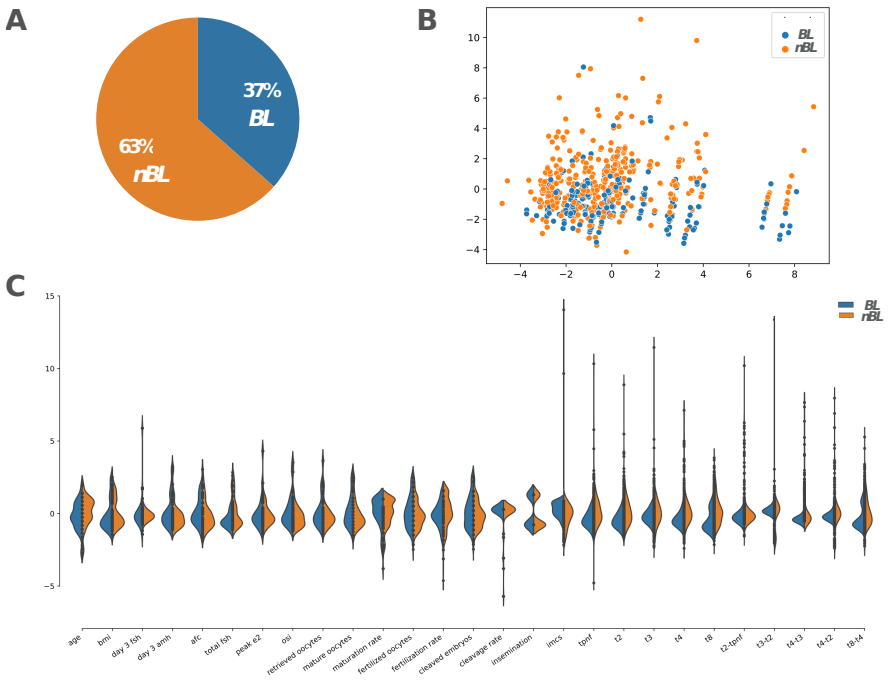


Figure 9.10: (A) Pie chart showing the distribution of the 575 embryos: embryos progressed to the expanded blastocyst stage on day 5 (BL, blue) or not (nBL, orange). (B) Scatter plot obtained from PCA considering all 575 embryos and all variables. The color of the embryos identifies those grown to the expanded blastocyst stage on day 5 (BL, blue) or not (nBL, orange). (C) Violin plots distribution of the z-score of the value of all features distinguishing embryos grown to the expanded blastocyst stage on day 5 (BL, blue) or not (nBL, orange).

9.3.3 Results

Correlation between the considered features

Table 9.2 summarizes the clinical characteristics of the 80 patients included into the Training dataset and the outcome of their IVF cycles.

Overall, 575 embryos were obtained, of which 210 (36.5%) progressed to the expanded blastocyst stage on day 5 (BL group) whereas 365 (63.5%) did not (nBL Group) (Figure 9.10A). At first, all embryos were considered together for a preliminary analysis applying the Principal Component Analysis (PCA) to generate a labelled scatter plot according to the clustering category of the embryos (BL or nBL). Although there was not a clear separation of the two embryo populations (Figure 9.10B), different density curves could

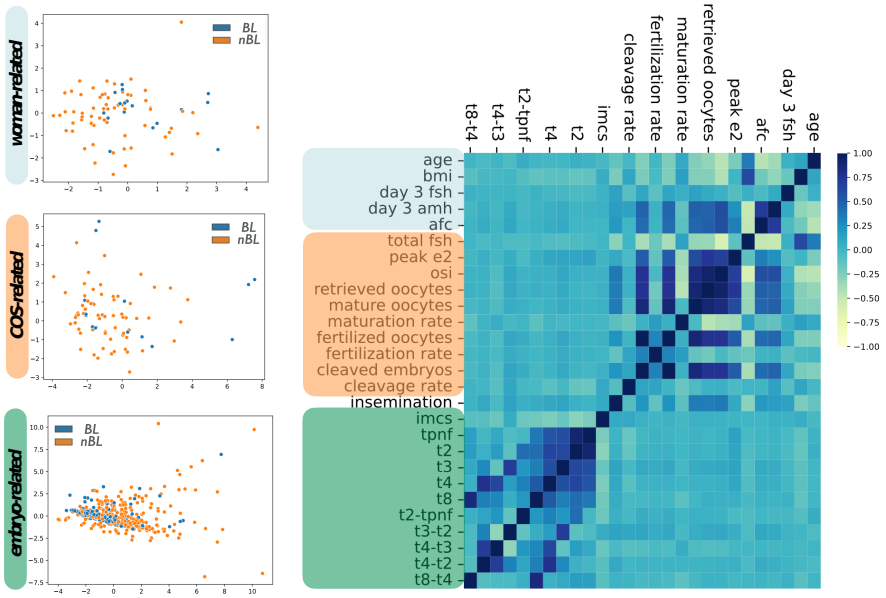


Figure 9.11: On the left: scatter plots computed through PCA for each set of variables, independently. On the right: heatmap plot showing the Pearson correlation value among all variables.

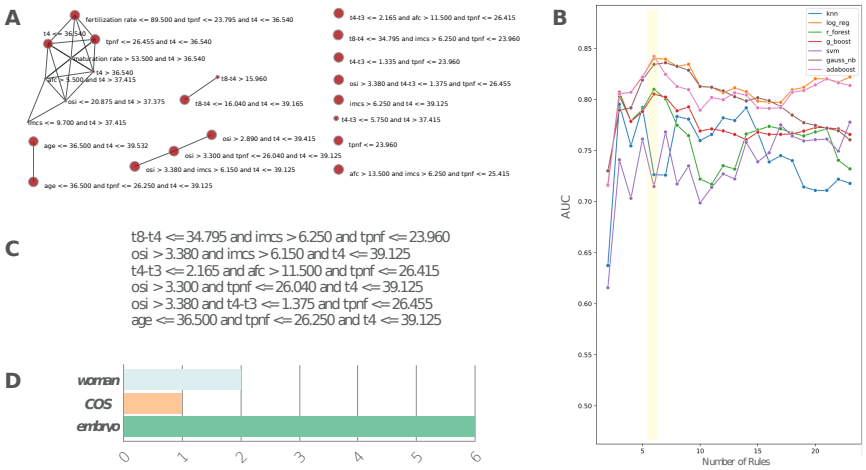


Figure 9.12: Correlation graph of the selected rules. Vertices represent the rules and arcs are reported only for a correlation value > 0.8 computed by Matthews Correlation Coefficient (MCC). (B) Line plot representing the ability of different combinations of classifiers to classify the expanded or not expanded blastocyst stage. Each dot corresponds to the AUC computed using a different number of rules in input. (C) Set of 6 rules of the embryo-signature with (D) their composition in terms of variable category.

Table 9.2: Patients' clinical characteristics and IVF outcome. The training and validation cohorts are composed of 80 and 10 women, respectively. Data are shown as mean ± standard deviation, or as a percentage.

Feature	Training cohort	Validation cohort
Age (years)	35.3 ± 3.5	35.5 ± 3.1
Duration of Infertility (years)	2.9 ± 1.5	2.4 ± 1
Body Mass Index (kg/m2)	24.2 ± 4.8	$24,6 \pm 2.6$
Day 3 FSH (IU/l)	7.3 ± 3.9	7.3 ± 2.4
AMH (ng/ml)	5.4 ± 4.2	5.2 ± 3.9
Antral follicle count (n)	17.5 ± 8.3	16.7 ± 9.4
Total FSH dose (IU)	2137.1 ± 900	2121.8 ± 930
Ovarian sensitivity index (OSI)	6.8 ± 4.7	6.4 ± 5.1
Retrieved COCs (n)	12.1 ± 5.3	12.0 ± 5.6
Mature (MII) oocytes (%)	85.9 ± 14.9	82.8 ± 11.7
Fertilization rate (%)	73.7 ± 17.7	75.5 ± 16.7
Cleavege rate (%)	98.6 ± 4.9	98.9 ± 5.3
Blastulation rate/2PN (%)	53.8 ± 4.7	55.4 ± 6.1

be noted after comparing the range of values of BL with those of nBL, as shown in Figure 9.10C. The inter-variables correlation analysis revealed two major clusters, the first between woman-related and COS-related variables, the second within embryo-related variables, as depicted in the heat map (Figure 9.11). Interestingly, the insemination technique showed either weak or no correlation with the other variables (Figure 9.11). In addition, a clear cluster was not detectable considering each set of variables to discriminate the BL and nBL embryo groups by scatter plots (Figure 9.11).

Selection of rules associated with blastocyst development

The first phase of EmbryoMLSelection framework application consisted in the identification of the most predictive features. This process exploits multiple selection algorithms (e.g. filter and embedded methods) to explore the ideal set composed of the balanced cut-off between the number of features and the power of their association with embryo development. The selection strategy was performed on the training set (70% of the total number of embryos), and the identified set of features was tested on a classification task using stratified 10-fold cross-validation repeated 100 times against the test set (the remaining 30% of the embryos). A total number of 12 variables composed the optimal set that provided the highest area under the curve (AUC) (data not shown). In a second phase, these variables were managed to define the set of rules (combination of features) able to discriminate embryos of BL and nBL groups. The Rules extraction module was applied to generate 71 rules from the training set.

To explore the association of the extracted rules within each other and with blastocyst development, a visualization module was displayed in which the rules were represented as nodes and the edge between two nodes represented a correlation value for those rules higher than 0.8 (computed by Matthews Correlation Coefficient (MCC). Applying the second and third modules of the EmbryoMLSelection framework, the extracted rules underwent a further selection process to remove those unpredictive or redundant, leading to a new set of 23 selected rules. The correlation graph drawn on the selected rules showed that 8 rules were isolated vertices (Figure 9.12A, red nodes on the right) highlighting their low (i.e. minor than 0.8) functional dependency from the others but, at the same time, their high relevance with the primary outcome (blastocyst development). By inspecting the plot reporting the AUC value (Figure 9.12B), a final set of 6 rules involving 9 variables (70% of which were embryo-related) was obtained (Figure 9.12C and 9.12D). This set of rules defined the embryo-signature and represented the combination of rules with the highest discrimination power.

Validation of the selected rules on an independent dataset

The performance of the 6 selected rules was tested by five classification algorithms on an independent validation cohort of 81 embryos obtained from other 10 patients with clinical characteristics comparable to the women included in the study (Supplementary Table 1). Overall, the rules composting the embryo-signature showed a predictive performance with AUC =0.842 and accuracy =0.81.

9.3.4 Discussion

Artificial Intelligence is one of the most promising, objective methodologies aimed at standardizing embryo assessment in human IVF. An intriguing application of AI within the IVF laboratory is providing new knowledge on cellular profiles regulating embryo in vitro growth and embryo competence [226]. The rather recent introduction of TLT offers the possibility to obtain a vast bulk of data on the kinetic of human embryo growth, producing a much more detailed timeline of dynamic events as well as showing previously unrecognizable phenomena [236]. In the present study, we describe the performance of the novel EmbryoMLSelection framework in identifying a set of rules associated to

timely embryo development to the expanded blastocyst stage on day 5, called embryo-signature. The rationale behind a two-step process of features selection, rule extraction and rule selection from a large number of variables/embryo (n=30) was the identification of a set of rules (from an initial number of 71 to the final 6) identifying an embryo-signature composed of relevant features (n=9), describing cleavage stage embryos able to timely (within day 5) progress to the expanded blastocyst stage. As lower implantation and clinical pregnancy rates were reported in case of transfer of slow-growing blastocysts vs. fully expanded day 5 blastocysts [237, 238], probably due both to poor embryo competence and the loss of embryonic-endometrial synchrony [239], we considered the fully expanded blastocyst on day 5 as the optimal development stage, that confers the highest probability of embryo implantation. Embryo selection models developed using morphokinetic parameters were previously shown to predict blastocyst development [240, 241]. Furthermore, the application of machine-learning technology provided an algorithm able to predict clinical pregnancy and live birth rate by analysing embryo morphokinetics [242]. Giscard d'Estaing [230] used a machine-learning system in order to build up a score for blastocyst formation with a prediction power having AUC =0.634. In another study, the prediction accuracy of embryo

assessment performed by experienced embryologists with morphokinetic grading methods added to conventional static morphology was shown to range between 60% and 70%, with AUC 0.63 - 0.70 [243]. Herein, we provide evidence that the novel EmbryoMLSelection framework allowed to perform a more precise evaluation of embryo dynamic growth with a performance described by AUC =0.84 and accuracy 81%. Notably, the Rules Selection step ensured such an increased performance providing a concomitant reduction of the rules and variables used (n=6 and 9, respectively). Importantly, the EmbryoMLSelection framework developed here was registered in the Docker image and therefore its application is globally accessible online. Of note, the rules associated with the ability of reaching the stage of expanded blastocyst on day 5 include early embryo-related variables, such as embryo morphological score on day 2, and some cytokinesis times occurring in the first three days of development (tPNf, t4, t4-t3, t8-t4). So far, only one study coupled TLT annotation with morphological embryo assessment performed with the evidence-based score named IMCS [244]. IMCS was based on the evidence of implantation and clinical pregnancy after double ET on day 2, and was incor-

porated into a complex prediction model for IVF outcome, recently shown to predict live birth with a remarkably good precision [245]. According to our results, good quality embryos having static morphological score > 6.0 on day 2 are more likely to reach the expanded blastocyst stage on day 5. In addition, the relevance of timings describing early embryo development is confirmed by previous studies reporting that a timely blastocyst development on day 5 can be predicted looking at the first three days of development [235, 246]. Moreover, morphokinetic data of cleavage stage embryos were found to be associated to both blastulation rate and blastocyst quality [247]. Indeed, embryos with quicker cleavage time from the 2-cells to the 8-cells stage have the highest potential to timely become blastocysts with good morphological score, and with the ability to expand and implant [248, 249]. In this context, the pivotal clinical significance of our framework would be to indicate on day 3 which embryos are more likely to develop into viable blastocysts, giving the potential advantage to select the most competent embryos on day 3 without the need to extend culture till day 5, thus saving time and resources. For the sake of convenience, the EmbryoMLSelection framework was also applied in a setting in which the variables were not previously selected. Specifically, all variables were used together in order to obtain at first 131 extracted rules, showing a high functional dependency highlighted by the high number of arcs (Supplementary Figure 2A). From a subset of 30 selected rules, 17 were finally derived based on the AUC in order to choose the best combination of rules with the highest discrimination power (Supplementary Figure 2B). However, these rules reached lower values in terms of both AUC (0.79) and accuracy (0.70) in the validation cohort with respect to the previously identified embryo signature. Finally, when considering only the embryo-related variables, 146 rules were extracted with high functional dependency (Supplementary Figure 3A). A subset of 30 selected rules allowed to identify in the validation cohort a final number of 21 rules reaching an AUC value =0.74 and an accuracy value =0.68 (Supplementary Figure 3B). This performance suggests that embryo-related variables alone are not enough to accurately describe blastocyst development, and that it is necessary to consider the overall set of variables while designing the framework. In fact, other clinical variables, such as age, AFC and OSI, were associated to the timely progression to the blastocysts stage. Indeed, female age defined as advanced (AMA ¿ 35 year) was extensively associated with a decline in oocyte yield, fertilization, and overall oocyte/embryo devel-

opmental competence, mainly due to an increased incidence of aneuploidies and a decreased mitochondrial activity [250, 251]. Studies reporting embryo morphokinetics from the fertilization to the pre-implantation period in women of AMA remain limited; our findings suggest a link between morphokinetic patterns and maternal age. Maternal age seems to have a relevant impact on the regulation of cell polarity during compaction, as well as on blastocoel cavity expansion, suggesting that AMA may affect embryo competence irrespective of the well-known consequences of oocyte meiotic errors [252]. On the other hand, AFC and OSI are markers of ovarian reserve and responsiveness to COS, and are associated not only with female age, but also with circulating AMH levels, oocyte yield and, ultimately, clinical pregnancy [253, 254]. Interestingly enough, the insemination technique (conventional IVF or ICSI) was not included as relevant variable in the selected rules, confirming previous evidence showing only minor morphokinetic differences between the two procedures [255]. The present study has the following limitations: (i) only couples undergoing single blastocyst transfer were considered in this study; (ii) the overall number of considered embryos was limited (n=575) but it constituted the entire time-lapse database available in our centre; (ii) embryo developmental timings were manually annotated, with unavoidable intra- and inter-operator variability [256]; (iv) a timely blastocyst formation has a limited association coefficient with embryo ploidy and implantation chance [257].

9.4 Index-driven subgraph search exploiting decision diagrams

9.4.1 Datasets description

For this study, we considered six different types of graphs. Four of them are real graphs widely used as a benchmarks in the fields of bioinformatics and computational chemistry, the others are synthetically generated by means of the Barabasi-Albert's and the Forest-Fire models. The choice of such two synthetic models has been taken according to their properties of the topologies to be similar the graphs used in biological databases. Differently from collections of real graphs, synthetic topologies allow us to investigate the performance of compared methods in relation to the parameters of such models, and thus to the properties of the produced topologies [258].

Biochemical structures

The collection of biochemical graphs was initially used for evaluating the performance of one-to-one subgraph isomorphism algorithms [259], and, nowadays, it is a well-established benchmark for graph theory problems linked to the subgraph isomorphism [260]. These four datasets that compose the collection are now described.

AIDS is the standard database for Antiviral Screen [261], and it consists of 40k chemical structures representing small molecules. Vertices are atoms and edge are the chemical bounds linking them. Vertices are labelled by the atomic element they represent, and there are a total of 62 distinct elements. The average number of vertices per graph is 44.98, and the average degree is 4.17.

PDBS is a benchmark composed of 600 target graphs representing the topological structure of proteins [262, 263]. Vertices are the atoms and edges are chemio-physical bounds between them. These graphs have up to 16,431 vertices and 33,562 edges, with an average degree over the whole dataset equal to 4.27. There are a total of 10 unique labels, corresponding to the atomic types.

PCM is composed of three-dimensional structures of protein, called protein contact maps [264]. Vertices represent the amino acids of a protein and edges informs about the spatial proximity of amino acids. The dataset contains 200 target graphs having up to 883 vertices and 18,832 edges, with an average of 376 vertices per graph and 44.78 edges per vertex. There are a total of 21 labels of which 18 appears on average in each graph.

PPI is a dataset of 20 protein-protein interaction target graphs of 5 different species: *Caenorhabditis elegants, Drosophila melanogaster, Mus musculus, Saccaromyces cerevisae* and *Homo sapiens* [265]. Protein interactoms are a widely used approach for investigation biological phenomena, since they reports the known physical interactions between proteins of a given living species. They are often embedded in graph-based multi-omics knowledge bases, and contribute to a form a predominant part of such graphs. For this reason, we decided to focus the attention on PPI network, with the future perspective of analysing more complete multi-omics graphs. Vertices are proteins and edges are predicted physical interactions between them. For each species, different thresholds on the accurateness of the prediction were applied, ranging from 0.4, 0.5, 0.6 to 0.7. Vertices are labelled according to their functional

category, for a total of 45 distinct categories. The dataset contains graphs up to 10,186 vertices and 179,348 edges, an average degree of 18.46 and an average number of distinct labels per graph equal to 28.45.

For all of the biochemical datasets, queries were extracted from the target graphs by fixing the desired number of edges, from 4, 8, 18 to 32, and such that the topological structure of the extracted graph reflects the properties of the graph of origin.

Synthetic graphs

The Barabasi-Albert's model is able to reproduce a graph with an observed stationary scale-free distribution, which reflects many of the structures that can be encountered in nature [266]. Starting from an initial set of vertices, m_0, the model inserts one vertex at time to the graph. At each insertion, new edges are added in order to connect the new vertex with existing ones. The probability of an edge with vertex i is $p_i = k_i^{\alpha}$, where k is the vertex degree and α is a user defined parameter. The benchmark contains 384 target graphs which were generated by fixing a desired number of vertices and average degree. Generated graphs have 200, 500, 1k, 5k, 10k and 20k vertices. In addition, three copies of each generated network are made in order to provide a labelled version of the initial structure with three different percentages of distinct labels, 0.1%, 1% and 10%. Labels are assigned randomly to vertices according to a uniform distribution.

The second type of synthetic graphs were generated according to the Forest-Fire model [267], that is inspired by forest growing behaviours, and which imposes a geometric distribution with mean $p/(1-p)$ which is used for randomly extract links between two distinct vertices. This benchmark contains 160 target graphs having the same number of vertices of the Barabasi-Albert benchmark, and they were labelled in the same way of the previous model. Moreover, the graphs were generated by varying the value of the model parameter p as 0.1, 0.3, 0.5, 0.7 and 0.9.

For both synthetic benchmarks, query graphs were extracted from the generated target graphs. The extraction was performed by fixing the number of desired vertices, ranging from 4, 8, 24, 32 to 64, and by extracting all edges among the selected vertices.

9.4.2 Experimental setup and output

We evaluated the performance of GRAPES-DD, with respect to its predecessor GRAPES, by taking into account both space and time requirements. In particular, we focused on the amount of primary memory that the two approaches require during the execution, reported as *memory peak*, as well as the space needed to store the built index in the hard disk, reported as *index size*. In addition, we compared the running time required by the two approaches for building the index. The analysis was mainly focused on the index construction phase because it is the main difference between the two approaches. They share the same methodology for what concerns the matching phase. In addition, the filtering time can be considered negligible with respect to the total querying time.

Both GRAPES-DD and GRAPES have been containerized in a Docker[29] image in order to ensure both functional and computational reproducibility of the experiments. The Dockerfile to build the image is provided together with the source code, and it is available at `https://github.com/qBioTurin/grapes-dd` or at `https://github.com/InfOmics/grapes-dd`. Both the tools were implemented in C++ and compiled with gcc 6.3.0. Then, the experiments have been carried out on a server equipped with four processors AMD Opteron 6167 2.20 GHz and 502 GB of RAM. Since GRAPES is a natively parallel software while GRAPES-DD is sequential, the experiments were executed using GRAPES with a single-thread.

Indexing

Figures 9.13 and 9.14 show memory peak and index size on the synthetic datasets obtained by indexing one target graph at time. Values are calculated taking into account three different grouping strategies that reflect the way in which the datasets are generated. Plots were generated via the *Pandas* framework available for Python [1]. In details, datasets were grouped by (i) percentage of distinct labels with respect to the total number of vertices of the graph, (ii) number of vertices and (iii) value of the Barabasi-Albert model parameter α or Forest-Fire parameter p.

[1] `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.boxplot.html`

Results show that, independently from the label percentage and model parameters, the performance of GRAPES-DD improves as the number of vertices of the indexed graph increases. In fact, for graphs having less than 5k vertices, the memory peak required by GRAPES-DD is higher than the peak of GRAPES, resulting in a ratio between the two values less than 1. The out-performance of GRAPES-DD can reach two/three orders of magnitude with respect to GRAPES, for graphs with 20k vertices, which means that the memory requirement of GRAPES-DD is one hundredth that of GRAPES.

Similar trends are observed for the size of the index when it is stored into the hard disk. In this case, the ratio can achieve five orders of magnitude as it is shown for the Forest-Fire graphs with 20k vertices. In general, best ratios are obtained for the Forest-Fire graphs with a high number of vertices, however, this behaviour is counterbalanced by the fact that on average Forest-Fire graphs with less than 5k vertices are also those with the lowest ratios.

For what concerns the memory peak, we can observe that the label percentage is a more crucial factor for the Barabasi-Albert model rather than for the Forest-fire model. More in general, a low label percentage is to the advantage of the trie structure of GRAPES because the extracted paths share and relatively high number of labels. Opposite trends are observed for what concerns the storing of the index.

As for the label percentage, model parameters produce less variation compared to the number of vertices. The Barabasi-Albert model produces scale-free networks where the distribution of the degrees of the vertices follows a power law. A value greater than 1 increases the skewness of the resultant distribution, while a value less than 1 flattens the distribution. Thus higher values trend to produce a more sparse graph. Results in Figure 9.13 show that GRAPES-DD performs better for dense graphs, namely for low values of the α parameter. The trend is confirmed by the results regarding the Forest-fire models (Figure 9.14), where higher values of the p parameters produce more dense graphs.

GRAPES-DD reaches an average indexing compression ratio of 11.16 with respect to GRAPES when Barabasi-Albert networks are indexed. Instead, an average ratio of 9.46 is reached over the Forest-Fire collection. The better ratio obtained by GRAPES-DD highlights that the application of MTMDDs is advantageous for any of the two types of random graphs, however, it is more suitable for Barabasi-Albert networks that are considered more similar

to biological networks.
Subsequently, we evaluated the performance of exploiting the MTMDD structure for indexing 14 collections of synthetic graphs (see Table 9.3). The first three collections are obtained by grouping Barabasi-Albert graphs by the label percentage, such that graphs having the same percentage are contained in the same collection. Similarly, Forest-Fire graphs were grouped into three further collections. The grouping procedure was also performed by taking into account the α and p model parameters. As for the previous analysis, the ratio is computed by dividing the values measured for the trie structure of GRAPES with those registered for the MTMDD of GRAPES-DD. As it has been shown for the single-graph analysis, the percentage of distinct labels with respect to the total number of vertices in the graphs) is a discriminant factor for the compression gain obtained by the MTMDD structure. Also the trends relative to the parameters of the models are confirmed. In general, the MTMDD structure is on average more convenient on the Forest-Fire graphs for what concerns the memory peak. Barabasi-Albert graphs with $\alpha = 0.5$ are an exception to this trend, since they reach the maximum registered ratio equal to 45. In contrast to the single-graph analysis, the space required for storing the index into the hard disk does not provide the same advantage to the MTMDD structure. In fact, in the single-graph analysis the ratio reaches a value of 10^5 that is two order of magnitude higher of the ratios obtained for the memory peak. On the contrary, these experiments show an inversion of the ratio such that the MTMDD structure reaches best results for the memory peak. It is notable to report that, while the trie structure requires a maximum of 94Gb of memory, the process for building the MTMDD-based index does not reaches the 9Gb of requirement, making it suitable for common personal computers.

Table 9.3 also shows the running time of the two approaches for building the index and for storing it. The MTMDD structure requires more time for its construction, the compression capability of the MTMDD must come with an unavoidable additional cost. However, the growth time is only a few minutes and the construction of the index is performed in a preprocessing phase, only once and reused for each query search.

Table 9.4 reports the complete set of experiments that were performed on the biochemical graphs. The experiments regard the indexing of the four different collections of real graphs. For this benchmark, ratios are less promi-

Table 9.3: Indexing comparison of GRAPES and GRAPES-DD of synthetic graphs in terms of RAM requirement, Storage requirement, and Building time.

			RAM req. (MB)			Storage req. (MB)			Build time (s)	
			DD	trie	ratio	DD	trie	ratio	DD	trie
Barabasi-A.	l	0.1%	3,649	7,935	2.2	305	3,493	11.5	470	109
		1%	8,229	66,838	8.1	1,543	28,772	18.6	646	214
		10%	7,876	81,552	10.4	10,071	34,368	3.4	668	265
	α	0.5	2,103	94,654	45.0	24,915	40,281	1.6	516	330
		1	8,351	58,519	7.0	16,602	25,145	1.5	766	213
		1.5	1,447	3,068	2.1	1,246	1,219	1.0	144	26
Forest-Fire	l	0.1%	834	3,929	4.7	121	1,689	13.9	63	29
		1%	1,308	21,255	16.2	738	9,229	12.5	77	66
		10%	1,167	24,936	21.4	5,351	10,650	2.0	62	70
	p	0.1	147	1,426	9.7	2,882	585	0.2	10	7
		0.3	188	2,451	13.1	3,358	1,024	0.3	14	9
		0.5	281	4,966	17.7	3,922	2,109	0.5	26	16
		0.7	487	11,694	24.0	4,535	5,020	1.1	57	37
		0.9	988	29,565	29.9	5,386	12,840	2.4	139	88

Table 9.4: Indexing comparison of GRAPES and GRAPES-DD of biochemical datasets in terms of RAM requirement, Storage requirement, and Building time

	RAM req. (MB)			Storage req. (MB)			Build time (s)	
	DD	trie	ratio	DD	trie	ratio	DD	trie
AIDS	5,304	1,064	0.20	164	39	0.24	170.12	16
PDBS	532	556	1.04	22	17	0.78	176.00	10.07
PCM	512	7,057	13.77	253	1,392	5.51	617.24	754.56
PPI	629	1,698	2.70	166	665	4.00	2,514.18	2,906.65

nent compared to synthetic graphs, however many of them are higher than 1, confirming a gain in using the MTMDD structure rather than the trie. The trend for which paths extracted from more dense and more uniform graphs are better compacted by the MTMDD structure is confirmed. In fact, the best ratio is obtained for the *PCM* collection that contains the most dense graphs. However, the *PCM* collection is also the one with the lowest number of labels and a relatively small number of vertices. Thus, it seems that the density of the graphs is the key factor for the good performance of GRAPES-DD in biochemical graphs. In addition, in contrast with the results on the synthetic graphs, the running time of GRAPES-DD for the construction of index is generally faster than the time required by GRAPES. In these cases, the compression capability of the MTMDD comes without additional cost.

Filtering

Collections of biochemical graphs were also used for evaluating the performance of GRAPES-DD during the querying phase in comparison with exiting

Table 9.5: Indexing comparison of GRAPES and GRAPES-DD of single PPI network in terms of RAM requirement, Storage requirement, and Building time

			RAM req. (MB)			Storage req. (MB)			Build time (s)	
Species	$\|V\|$	$\|E\|$	DD	trie	ratio	DD	trie	ratio	DD	trie
S. cerevisiae	4,709	40,284	38.1	91.1	2.39	10.5	26.2	2.49	60.93	63.49
	5,230	53,699	56.3	136.3	2.42	11.7	42.4	3.63	604.47	630.92
	5,762	76,482	61.1	150.9	2.47	12.7	47.5	3.73	846.24	879.90
	5,936	89,674	46.5	121.2	2.61	12.9	36.8	2.86	128.52	135.01
C. elegans	1,557	2,472	7.1	5.7	0.80	0.2	0.3	1.45	0.24	0.20
	2,421	3,981	7.8	7.9	1.02	0.5	1.0	2.06	0.52	0.44
	3,664	7,005	10.4	14.6	1.40	1.0	2.8	2.65	1.33	1.13
	6,173	26,184	25.1	58.5	2.33	4.3	16.0	3.75	34.16	34.19
D. melanogaster	1,185	2,008	8.0	12.1	1.51	0.6	1.5	2.30	0.31	0.26
	2,488	6,151	12.1	32.1	2.65	1.9	6.6	3.47	3.28	3.21
	2,729	7,235	13.3	36.2	2.73	2.3	7.9	3.37	4.30	4.22
	7,928	37,542	52.3	198.7	3.80	13.1	64.3	4.90	144.05	156.07
M. musculus	1,810	2,413	8.0	13.1	1.64	0.7	2.2	2.94	0.42	0.36
	3,255	5,424	11.0	31.0	2.81	1.9	7.1	3.68	2.52	2.50
	3,758	6,853	13.1	43.6	3.33	2.6	11.2	4.30	4.47	4.61
	6,875	23,779	41.1	193.6	4.71	12.4	62.1	5.02	76.64	81.56
H. sapiens	4,638	10,665	17.3	55.0	3.18	3.6	14.6	4.07	5.60	5.36
	8,728	31,164	53.5	215.4	4.02	13.1	70.8	5.42	65.14	68.26
	9,826	48,835	87.5	351.4	4.02	21.5	120.1	5.59	213.95	230.16
	10,186	51,484	89.2	391.8	4.39	22.0	134.1	6.10	191.63	209.63

approaches VF2 [78] and CT-Index [89]. VF2 is a non-indexed approach used by GRAPES and GRAPES-DD in the verification phase. The comparison with it allows us to evaluate the effectiveness of using indexing in graph searching applications. CT-Index is a index-based graph searching solution that uses paths as indexing features. Biochemical graphs have already been used for investigating the performance of GRAPES, VF2 and CT-Index [87, 106]. Here, we propose those comparisons by adding GRAPES-DD. GRAPES-DD is compared with GRAPES, two configurations of CT-Index and the pure subgraph isomorphism algorithm VF2. All the compared methods enumerate all the matches. CT-Index was run with default parameters (CT-index def), such that paths, cycles and trees are indexed. Moreover, a configuration (*CT-index 4*) which only includes paths of length 4 was taken into account. We were not able to run CT-Index on the PCM and PPI datasets due to excessive memory usage of the tool.

Figure 9.15 shows the cumulative time (in seconds) for executing 100 queries. Running times of GRAPES, GRAPES-DD and CT-index include the time to read graphs from the input files, filtering time and verification time. The time required by the methods for reading the pre-built index is considered only once and it is included in the running time of the first exe-

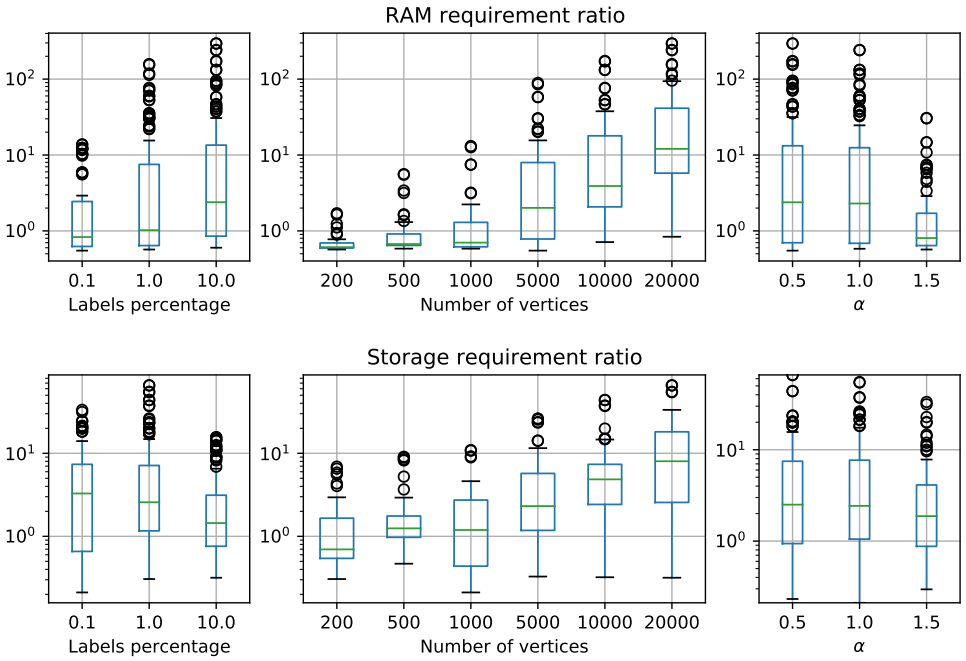


Figure 9.13: GRAPES/GRAPES-DD ratios of memory peak (as a RAM requirement) and index size (as a storage requirement), obtained by indexing Barabasi-Albert graphs. The chart was made by using the *boxplot* function of the Python3 Pandas module.

cuted query. Since no index is built by VF2, its total execution time is taken into account. CT-index takes 461 seconds for building the index of the AIDS datasets with default parameters and 82 seconds for indexing only paths of length 4. Moreover, it requires 4,400 for indexing the PDBS collection with default parameters and 40 seconds when only paths of length 4 are taken into account. In all experiments, CT-index is outperformed by the other three approaches.

On AIDS collection (see Figure 9.15 (a)), GRAPES-DD is not able to outperform GRAPES; its running time is close to the one of VF2. As shown in Table 9.4, this type of biochemical structures are too small and not suitable for being indexed and queried via MTMDDs. The overhead for reading the index and for extracting candidate graphs according to the query structure is not amortized during the verification phase, indeed, GRAPES-DD requires 15 seconds for reading the index and an average of 12 seconds for the filtering phase. On the contrary, GRAPES requires only 3.5 seconds for loading the index and an average of 3 seconds for the filtering.

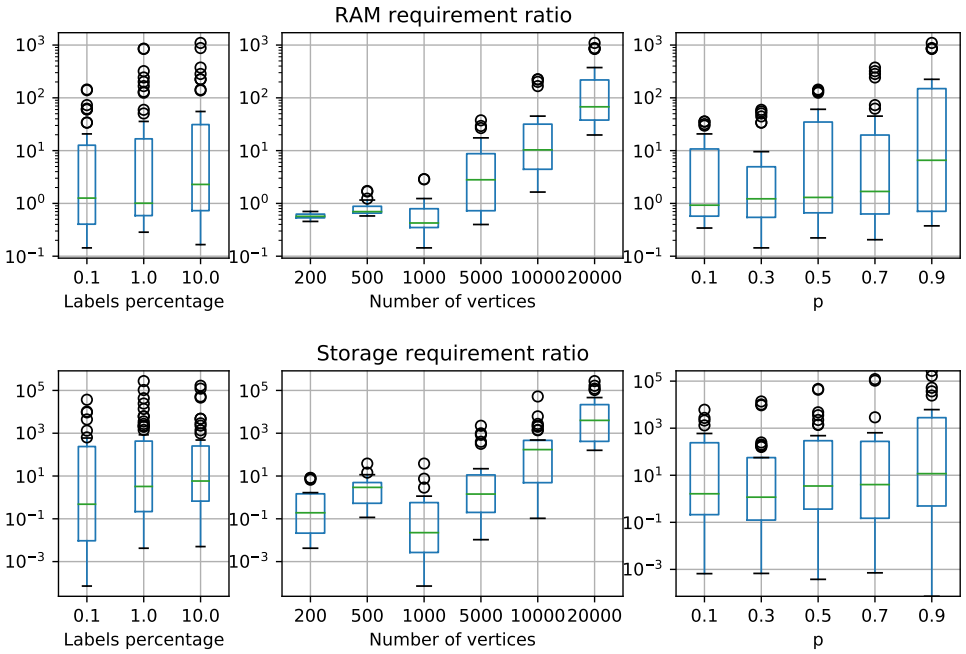


Figure 9.14: GRAPES/GRAPES-DD ratios of memory peak (as a RAM requirement) and index size (as a storage requirement), obtained by indexing Forest-Fire graphs. The chart was made by using the *boxplot* function of the Python3 Pandas module.

The VF2 algorithm is outperformed by GRAPES-DD in the PDBS, PCM and PPI collections (see Figure 9.15 (b), (c) and (d)). Moreover, VF2 is outperformed by GRAPES also in AIDS dataset. Thus, the index-based methodology used by GRAPES and GRAPES-DD is generally helpful in reducing the time required for the verification phase.

Regarding the PDBS collection (see Figure 9.15 (b)), GRAPES-DD requires 2.3 seconds for loading the index and an average of 6 seconds for filtering the collection. GRAPES requires 0.12 seconds for the index load and 20 seconds for the filtering phase. Since GRAPES-DD and GRAPES produce the same set of candidate graphs, GRAPES-DD outperforms GRAPES thank to its performance in the filtering phase.

Considering the PCM collection (see Figure 9.15 (c)), GRAPES-DD requires 20 seconds for loading the index and an average of 14 seconds for filtering the collection. GRAPES requires 30 seconds for the load and 2 seconds for the filtering. Thus, GRAPES-DD builds a more succinct index that allows a fast loading time, however it is not sufficient for outperforming GRAPES in

filtering time.

On the PPI collection (see Figure 9.15 (d)), GRAPES-DD and GRAPES have comparable running times. GRAPES-DD requires 13 seconds to load the index, in contrast to 2 seconds required by GRAPES. However, GRAPES-DD spends on average 0.05 seconds for the filtering phase, while GRAPES requires on average 11 seconds.

Lastly, Table 9.5 reports the results regarding the PPI networks obtained by indexing one PPI at time, since PPI networks are often analysed standalone. Similarly to the synthetic networks, the increase of the graph size, i.e. number of vertices $|V|$ and number of edges $|E|$, results in a better performance of GRAPES-DD with respect to GRAPES. However, comparing the ratios obtained for *M. musculus* and *H. sapiens* we can deduce that as expected there is not a fixed correlation between the graph size and the performance. Therefore, the intrinsic nature of the graph is also responsible for these results. PPI networks are also the targets for which running times of GRAPES-DD are comparable to those of GRAPES, and some times they are even better. The GRAPES-DD building approach includes the construction of partial tries but without merging them. The cost for traversing a single whole trie may limit GRAPES.

Variable ordering

We empirically tested our two research questions **R1** and **R2**, initially introduced in Section 6.5, on a subset of the well-known biological benchmarks previously described, namely the PPI (i.e. protein-protein interaction) networks of 5 different species, and the standard database for *Antiviral Screen*(AIDS) [261].

We conducted a set of experiments over the graph databases described above. We indexed each database using labeled paths up to length 4, so that each index MTMDD is defined over 5 variables. Then, for each collection, we obtain the size of the index MTMDD for all possible variable orders.

For each collection, we first obtain the measurements on the index MTMDD by testing all 120 possible variable orderings, namely the factorial of the number of variables.

Figure 9.16 reports the results for the **R1** question on the 6 benchmarks. Each dot represents one of the 120 possible variable orders. Dots are coloured by their respective stratification induced by the level of the identifier vari-

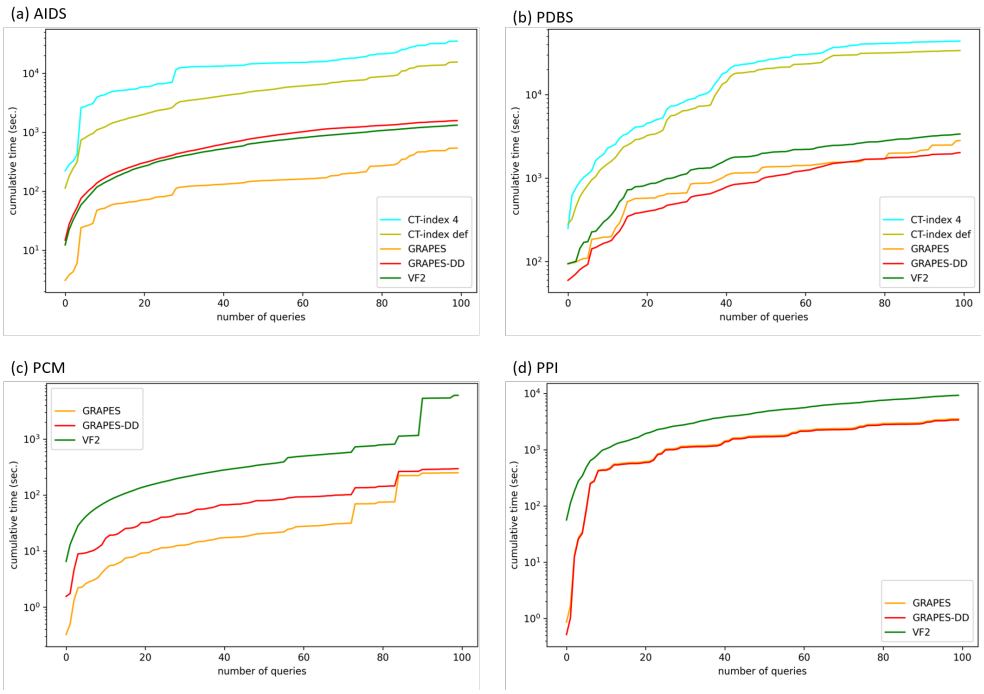


Figure 9.15: Cumulative time for running 100 queries over the four collections of biochemical graphs. The chart was made by using the *plot* function of the Python3 Pandas module.

able. In each stratum, a dashed line represents the trend of the relation between the *SOE* metric and the final DD size. The number indicates the value of Spearman's correlation coefficient. We can observe that the metric has medium-to-strong anticorrelation values in all strata except for the one where the identifier is positioned at the bottom, whose sizes are almost insensible to the reordering of the label variables. The figure shows a very positive result, because it shows that a heuristic that maximizes the *SOE* metric has a high chance of selecting a good order that minimizes the DD size. Moreover, the cross on each stratum identifies the ordering that would be selected by the proposed heuristic EntropyHeu when fixing the position of the identifier variable, while the star indicates the final order chosen by such heuristic without fixing the position of the identifier variable.

Figure 9.17 shows the results for the **R2** question on the effectiveness of the EntropyHeu heuristic on the 6 benchmarks. Relative DD sizes are shown on the y-axis, while the x-axis has no meaning (it is only used for visualization purposes to separate the dots). The green cross identifies the relative DD size

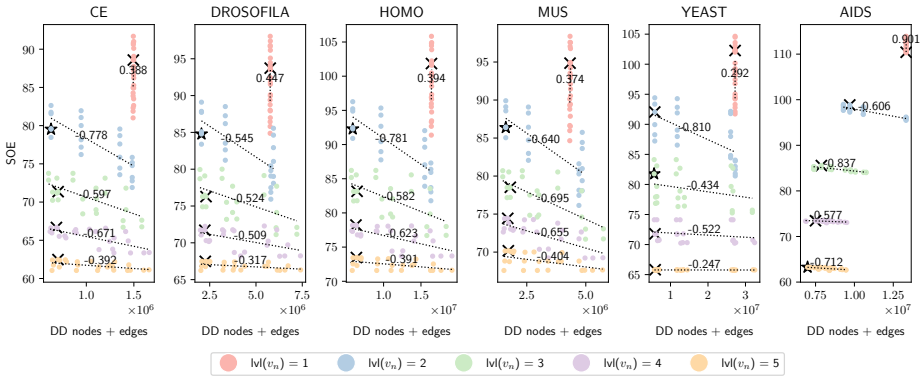


Figure 9.16: Spearman's correlation and trend lines of the *SOE* metric value w.r.t. the DD sizes, divided by sample strata. The black symbol on each strata identifies the order that is selected by EntropyHeu. In particular, the black star identifies the final order chosen by the heuristic.

of the final order selected by EntropyHeu, while the blue bar identifies the average size that would be obtained by randomly taking an order among the possible 120 orders. We can observe that the greedy heuristic that follows the metric *SOE* is actually capable of selecting almost-optimal orders in all the tested cases, showing the effectiveness of the proposed information-based strategy.

Looking at the star symbols in Figure 9.16 we can observe some characteristics of the final variable order chosen in each dataset. For CE, DROSOFILA, HOMO and MUS datasets, our heuristic EntropyHeu choose the same order $O_1 = \{v_1, v_5, v_4, v_2, v_3\}$ located in the second stratum. Differently, in the last PPI network (i.e. YEAST) the variable order $\{v_1, v_2, v_5, v_4, v_3\}$ located in the third stratum was selected. Despite it is different from O_1, we observe some similarities: they both start and end with v_1 and v_3 variables, respectively. Finally, the order found for AIDS dataset, which is $\{v_4, v_3, v_1, v_2, v_5\}$, places the identifier variable as the root of the MTMDD. In conclusion, as expected, we observed that similar graph characteristics have led to similar efficient variable order (see PPI results). Of course, future investigations will be needed to better identify such connections between graph features and the effectiveness of particular variable orders.

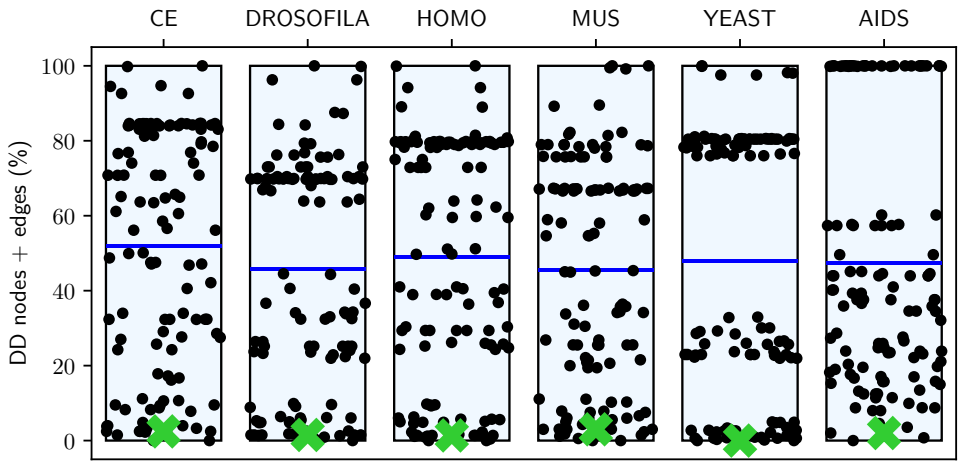


Figure 9.17: Relative position of the variable order selected by EntropyHeu among all the other possible orders.

9.4.3 Discussion

In this study, we deal with the problem of reducing the indexing size of biochemical and biological graph searching systems to make them effective with the increasing size of the structures. We show that the indexing of labelled graphs can take the advantages of newly adapted data structures based on decision diagrams. These techniques allow already existing methodologies to increase their compression power, in terms of memory consumption, without significantly increasing the searching time requirement.

We examined synthetic graphs because they offer a more systematic way of investigating performance of indexing using decision diagrams. Since the type of the generated graphs reflects the structures that are found in nature, their analysis can be exploited for inferring performance behaviour of real biochemical and biological structures. The results showed that relevant indexing compression ratio can be obtained in relation with the size and the topological structure of the graphs and the distribution of labels within them. Moreover, the larger are the indexed graphs, the higher is the advantage of using Decision Diagram data structure.

A well-established benchmark was also used for evaluating the performance on real graphs. The size of the considered graphs are relatively small, compared with the synthetically generated ones, however trends of gain ratio are confirmed. This must be considered in the perspective of future applications of

the proposed indexing technique, because the continuous development of new technologies for extraction biological information leads to the construction of biological relational systems that constantly increase in size. In addition to the gain in compression ratio, GRAPES-DD outperforms GRAPES in terms of build times while maintaining comparable query times. Furthermore, our analyse show that graph search approaches based on indexing, in graphs of some complexity, can amortize the overhead of building indexing data structures at query time.

Moreover, we extended the approach initially proposed and published in [121] by investigating how the MTMDD variable order may affect the performance of such an approach in terms of memory consumption. To achieve this task we first proposed the new metric *SOE* based on the Shannon entropy which experimentally showed a medium-to-strong anticorrelation with respect to the DD size encoding the graph indexing. Then we developed the sub-optimal heuristic EntropyHeu inspired to the information gain which is able to derive a variable order comparable with the theoretical-optimal order derived by the *SOE* metric. As a future extension, we will apply the EntropyHeu heuristic on a bigger set of benchmarks coming from different research fields and we will evaluate its performance by increasing the length of labeled paths. In addition, we will also investigate how the graph characteristics (e.g. graph sizes, number of labels, sparseness, communities, ...) could affect the internal structure of the index and the choice of the optimal variable order.

Chapter 10

Conclusion and future work

In this thesis, two original contributions were presented, concerning (i) the definition of new high-level formalisms for biological knowledge representation, and (ii) the development of general and efficient analysis techniques for dealing with massive and heterogeneous omics data. These theoretical results are then implemented into two software components, namely (i) a new general modular framework for multi-omics integration via machine learning, which can be exploited even by researchers without advanced mathematical and computational skills, and (ii) an index-driven subgraph searching algorithm exploiting decision diagrams as the indexing data structure.

In detail, the first contribution regards the definition of a novel multivariate filter-based feature selection method handling both high-dimensional datasets and small sample sizes. The feature graph formalism provides a compact search space for the feature selection approach by modeling both features' importances and redundancies, and it is introduced in Chapter 5. On top of this representation, (i) an mRMR-based metric on feature subgraph and (ii) a genetic algorithm optimizing the metric have been defined. We tested this approach on a fluxomics dataset composed of 430 samples and over 7000 features (i.e. rates of metabolic reactions) for biomarker discovery of glycolysis-associated clusters in colorectal cancer profiles. In particular, the shifting from the tabular dataset to a graph representation enables a concise but meaning-

ful representation of the salient relationships among features. Considering this formalism, in future works we will investigate (i) how to handle heterogeneous feature types (i.e. quantitative, categorical, ordinal, etc), and (ii) exploiting other bio-inspired optimization algorithms rather than genetic algorithms, such as swarm intelligence optimization methods (e.g. Ant-Colony, Particle-Swarm, ...). Moreover, we will apply this new approach for modeling a multi-omics real case study, such as CRC data exploiting miRNomics, metabolomics and metagenomics data on Italian and Czech cohorts.

The above theoretical result has been included in FeatSEE, a general and modular framework for multi-omics integration via machine learning that is presented in Chapter 7. The novelties and strengths of the proposed framework can be summarized as follows: (i) the high level of abstraction to compose user-defined workflows of analysis by combining end-to-end modules, (ii) a high parameterization of end-to-end modules,(iii) framework portability and reproducibility of the results granted by Docker technology, (iv) the possibility to be used by both expert and non-expert users: the former can implement new end-to-end modules, whilst the latter can exploit the high-level functionalities without worries about the underlying source code. The high-level implemented functionalities can be grouped into five classes: (i) data preprocessing building what is needed by the system from raw files, (ii) model evaluation estimating performance metrics in a given task (i.e. binary classification), (iii) feature selection identifying feature sets that are predictive with respect to the target, (iv) feature extraction creating new features exploiting the current datasets, and (v) model tuning (hyperparameter optimization), namely the identification of the best values of the hyperparameters of a given learning algorithm, which is then trained with the best parameters and saved on a file for future reuse.

The effectiveness of this framework and the theoretical results are shown through three different case studies in which we investigated i) biomarker discovery for the identification of a miRNA signature for colorectal cancer in multiple cohorts (Section 9.1), ii) a functional data integration approach on fluxomics obtained by gene expression data through Flux Balance Analysis (Section 9.2), and iii) an explainable feature extraction approach of in-vitro fertilization data to investigate when embryos reach expanded blastocyst on day 5 (Section 9.3). In particular, by using the FeatSEE functions, in all these case studies we are able 1) to easily define and execute the machine

learning workflow with all its underlying intermediate steps, 2) to analyse the given datasets for different aims, and 3) to easily simulate different scenarios. In future works, we will extend the framework to other supervised tasks, such as regression and multi-class classification. Moreover, we are working on a web-based GUI for ease of the definition of the workflow of analysis, the setting of modules' parameters, to control the execution of the modules and, in particular, for output visualization. Another future branch is the integration of a workflow management system (e.g. Nextflow) for (i) the automation of the execution of the workflow composed as a sequence of end-to-end modules, and in particular, (ii) for scaling performances on high-performance computing systems, since the WFMSs provide key capabilities enabling efficient resource allocation, job scheduling and monitoring.

The second contribution is a novel application of symbolic data structures in the context of index-driven subgraph searching, and it consists of three computational techniques based on decision diagrams. We consider the state-of-the-art tool GRAPES as the starting point, and then we relaxed the constraints of having a tree-based data structure by exploiting a family of graph-based data structures in place of the GRAPES' trie. The first technique is presented in Section 6.3 and describes how different decision diagrams can be exploited for computing and storing the graphs' features together with the additional information.

The second technique is described in Section 6.4, in which we showed how an MTMDD index can be easily manipulated for the definition of a filtering strategy. In future work, we will investigate the definition of an ad-hoc algorithm for performing a one-way filtering step, namely to verify the query constraints during index manipulation, instead of extracting a portion of the index that has to be further inspected.

Finally, in Section 6.5 we presented the third technique, which is a variable reordering heuristic algorithm to cope with the issue of decision diagrams whose memory efficiency is strictly dependent on the order imposed on its variables. Here, future work is to investigate other heuristics, such as conditional entropy for estimating the entropy of variable n-uples instead of single variables at each time.

These theoretical results were implemented in GrapesDD (Chapter 8), a novel index-driven subgraph searching tool derived from GRAPES of Giugno et al. by replacing the index trie with a MTMDD and in which the theoretical

results proposed on decision diagrams have been implemented. The novelties and strength of the proposed approach are the usage of a little-known symbolic data structure that allows efficient storage and advanced manipulation operators. The effectiveness of this methodology and the theoretical results are shown through different case studies involving both real and synthetic well-known datasets. Future works involve (i) designing an efficient thread-based indexing phase as the original GRAPES does, and (ii) removing the intermediate tries used to count feature occurrences. Another future direction consists of analyzing the internal structure of the index MTMDD on different graph collections, such as the number of nodes per layer, as well as studying how the characteristics of the graph collections (e.g. number of labels, density, etc.) impact to the memory occupation of the decision diagram. This could help in the design of a variable ordering heuristic algorithm, whose aim is the identification of a possibly good variable ordering before indexing. This will be done by exploiting the expert interface of the Meddly library that allows direct access to DD nodes, among other things.

Lastly, as a novel research direction, we will condense index-driven subgraph search and feature graph formalism for experiments with index-driven feature selection in multi-omics datasets, where the omics layers will define the label alphabet of the graph collection. A way to go is by exploiting the reverse labeled feature graph such that an edge means that the feature endpoints are independent, and where the label alphabet comprehends the omics layers appearing in the starting tabular dataset. Candidate biomarkers can be represented as relatively small graphs, such as labeled cliques, and exhaustively identified within the graph collection. Then, a hybrid-based feature selection can be easily defined by evaluating the feature sets matching the query constraints using one or more learning algorithms, optionally exploiting a relevance-based heuristic metric for filtering unpromising candidate feature sets. Eventually, bootstrap sampling can be exploited to build bootstrap replicas of the initial dataset and obtain a graph collection, which enables a sort of majority voting for the identification of stable feature sets.

Abbreviations

ACO Ant Colony Optimization. 31

BDD Boolean Decision Diagram. 49

CRC Colorectal Cancer. 6

DD Decision Diagram. 46, 47

DT Decision Tree. 86

EA Evolutionary Algorithm. 31

FBA Flux Balance Analysis. 13, 14, 138, 142

GA Genetic Algorithm. 31

GB Gradient Boosting. 87

GIGO Garbage-In Garbage-Out. 29

ICA Independent Component Analysis. 30

kNN k-Nearest Neighbors. 86

LR Logistic Regression. 86

LXC Linux Container project. 17, 18

MCC Matthews Correlation Coefficient. 85, 98

MDD Multi-way Decision Diagram. 49

MI Mutual Information. 84

mRMR minimum Redundancy Maximum Relevance. 31, 58

MTBDD Multi-Terminal Boolean Decision Diagram. 49

MTMDD Multi-Terminal Multi-way Decision Diagram. 49, 172

NB Naive Bayes. 86

NGS Next-Generation Sequencing. 1, 3, 11, 12, 14

NMF Non-negative Matrix Factorization. 30

PBCC Point-Biserial Correlation Coefficient. 84, 85

PCA Principal Component Analysis. 30

PCC Pearson's Correlation Coefficient. 84, 85, 142

PSO Particle Swarm Optimization. 31

RBP Reproducible Bioinformatics Project. 20

RF Random Forest. 87

SD Somer's D. 84, 85

SNP Single Nucleotide Polymorphism. 11

SRCC Spearman Rank Correlation Coefficient. 84, 85

SVM Support Vector Machines. 86

VM Virtual Machine. 16–19

WfMS Workflow Management Systems. 20

Bibliography

[1] Irun R Cohen and David Harel. Explaining a complex living system: dynamics, multi-scaling and emergence. *Journal of the Royal Society interface*, 4(13):175–182, 2007.

[2] Muhammad Afzaal, Farhan Saeed, Yasir Abbas Shah, Muzzamal Hussain, Roshina Rabail, Claudia Terezia Socol, Abdo Hassoun, Mirian Pateiro, José M Lorenzo, Alexandru Vasile Rusu, et al. Human gut microbiota in health and disease: Unveiling the relationship. *Frontiers in microbiology*, 13:999001, 2022.

[3] Gilbert S Omenn, Sharly J Nass, Christine M Micheel, et al. Evolution of translational omics: lessons learned and the path forward. 2012.

[4] Jason Y.H. Chang and Sylvain Ladame. Chapter 1.1 - diagnostic, prognostic, and predictive biomarkers for cancer. In Sylvain Ladame and Jason Y.H. Chang, editors, *Bioengineering Innovative Solutions for Cancer*, pages 3–21. Academic Press, 2020.

[5] F. Sanger, S. Nicklen, and A. R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proc. Natl. Acad. Sci. U.S.A.*, 74(12):5463–5467, Dec 1977.

[6] J. C. Venter et al. The Sequence of the Human Genome. *Science*, 291(5507):1304–1351, 2001.

[7] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, feb 2001.

[8] The 1000 Genomes Project Consortium et al. A global reference for human genetic variation. *Nature*, 526:68 EP –, Sep 2015. Article.

[9] Zhong Wang, Mark Gerstein, and Michael Snyder. Rna-seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, 10(1):57–63, 2009.

[10] Fatih Ozsolak and Patrice M Milos. Rna sequencing: advances, challenges and opportunities. *Nature reviews genetics*, 12(2):87–98, 2011.

[11] Wendy Weijia Soon, Manoj Hariharan, and Michael P Snyder. High-throughput sequencing for biology and medicine. *Molecular systems biology*, 9(1):640, 2013.

[12] Ali Mortazavi, Brian A Williams, Kenneth McCue, Lorian Schaeffer, and Barbara Wold. Mapping and quantifying mammalian transcriptomes by rna-seq. *Nature methods*, 5(7):621–628, 2008.

[13] A. D. Goldberg, C. D. Allis, and E. Bernstein. Epigenetics: A Landscape Takes Shape. *Cell*, 128(4):635–638, feb 2007.

[14] George M Weinstock. Genomic approaches to studying the human microbiota. *Nature*, 489(7415):250–256, 2012.

[15] Luke K Ursell, Jessica L Metcalf, Laura Wegener Parfrey, and Rob Knight. Defining the human microbiome. *Nutrition reviews*, 70(suppl_1):S38–S44, 2012.

[16] Jean-François Hocquette, Isabelle Cassar-Malek, Augustin Scalbert, and F Guillou. Contribution of genomics to the understanding of physiological functions. *J Physiol Pharmacol*, 60(Suppl 3):5–16, 2009.

[17] Christophe H Schilling, Jeremy S Edwards, David Letscher, and Bernhard Ø Palsson. Combining pathway analysis with flux balance analysis for the comprehensive study of metabolic systems. *Biotechnology and bioengineering*, 71(4):286–306, 2000.

[18] Pernice Simone, Follia Laura, Balbo Gianfranco, Milanesi Luciano, Sartini Giulia, Totis Niccoló, Lió Pietro, Merelli Ivan, Cordero Francesca, and Beccuti Marco. Integrating petri nets and flux balance methods in computational biology models: a methodological and computational practice. *Fundamenta Informaticae*, 171(1-4):367–392, 2020.

[19] Gal Winter and Jens O Krömer. Fluxomics–connecting 'omics analysis and phenotypes. *Environmental microbiology*, 15(7):1901–1916, 2013.

[20] Andrea Franceschini, Damian Szklarczyk, Sune Frankild, Michael Kuhn, Milan Simonovic, Alexander Roth, Jianyi Lin, Pablo Minguez, Peer Bork, Christian Von Mering, et al. String v9. 1: protein-protein interaction networks, with increased coverage and integration. *Nucleic acids research*, 41(D1):D808–D815, 2012.

[21] David Croft, Gavin O'kelly, Guanming Wu, Robin Haw, Marc Gillespie, Lisa Matthews, Michael Caudy, Phani Garapati, Gopal Gopinath, Bijay Jassal, et al. Reactome: a database of reactions, pathways and biological processes. *Nucleic acids research*, 39(suppl_1):D691–D697, 2010.

[22] Andrew Chatr-Aryamontri, Rose Oughtred, Lorrie Boucher, Jennifer Rust, Christie Chang, Nadine K Kolas, Lara O'Donnell, Sara Oster, Chandra Theesfeld, Adnane Sellam, et al. The biogrid interaction database: 2017 update. *Nucleic acids research*, 45(D1):D369–D379, 2017.

[23] Mark B Gerstein, Anshul Kundaje, Manoj Hariharan, Stephen G Landt, Koon-Kiu Yan, Chao Cheng, Xinmeng Jasmine Mu, Ekta Khurana, Joel Rozowsky, Roger Alexander, et al. Architecture of the human regulatory network derived from encode data. *Nature*, 489(7414):91–100, 2012.

[24] Aaron K Wong, Christopher Y Park, Casey S Greene, Lars A Bongo, Yuanfang Guan, and Olga G Troyanskaya. Imp: a multi-species functional genomics portal for integration, visualization and prediction of protein functions and networks. *Nucleic acids research*, 40(W1):W484–W490, 2012.

[25] Bonnie Berger, Jian Peng, and Mona Singh. Computational solutions for omics data. *Nature reviews genetics*, 14(5):333–346, 2013.

[26] Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604), 2016.

[27] Docker. What is a container? `https://www.docker.com/resources/what-container/`. Accessed: 2023-09-22.

[28] C. Boettiger. An introduction to Docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, January 2015.

[29] Dirk Merkel. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux J.*, 2014(239), March 2014.

[30] Gregory M Kurtzer, Vanessa Sochat, and Michael W Bauer. Singularity: Scientific containers for mobility of compute. *PloS one*, 12(5):e0177459, 2017.

[31] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten simple rules for reproducible computational research. *PLoS computational biology*, 9(10):e1003285, 2013.

[32] Neha Kulkarni, Luca Alessandrì, Riccardo Panero, Maddalena Arigoni, Martina Olivero, Giulio Ferrero, Francesca Cordero, Marco Beccuti, and Raffaele A Calogero. Reproducible bioinformatics project: a community for reproducible bioinformatics analysis pipelines. *BMC bioinformatics*, 19(10):5–13, 2018.

[33] Paolo Di Tommaso, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. Nextflow enables reproducible computational workflows. *Nature biotechnology*, 35(4):316–319, 2017.

[34] Tom M Mitchell. Does machine learning really work? *AI magazine*, 18(3):11–11, 1997.

[35] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[36] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[37] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[38] Robert E Schapire et al. A brief introduction to boosting. In *Ijcai*, volume 99, pages 1401–1406. Citeseer, 1999.

[39] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[40] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[41] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[42] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1-2):245–271, 1997.

[43] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[44] Nicholas Pudjihartono, Tayaza Fadason, Andreas W Kempa-Liehr, and Justin M O'Sullivan. A review of feature selection methods for machine learning-based disease risk prediction. *Frontiers in Bioinformatics*, 2:927312, 2022.

[45] Hui-Huang Hsu, Cheng-Wei Hsieh, and Ming-Da Lu. Hybrid feature selection by combining filters and wrappers. *Expert Systems with Applications*, 38(7):8144–8150, 2011.

[46] Parham Moradi and Mehrdad Rostami. Integration of graph clustering with ant colony optimization for feature selection. *Knowledge-Based Systems*, 84:144–161, 2015.

[47] Giorgio Roffo, Simone Melzi, Umberto Castellani, Alessandro Vinciarelli, and Marco Cristani. Infinite feature selection: a graph-based feature filtering approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12):4396–4410, 2020.

[48] Mehrdad Rostami, Kamal Berahmand, and Saman Forouzandeh. A novel community detection based genetic algorithm for feature selection. *Journal of Big Data*, 8(1):2, 2021.

[49] Milan Picard, Marie-Pier Scott-Boyer, Antoine Bodein, Olivier Périn, and Arnaud Droit. Integration strategies of multi-omics data for machine learning analysis. *Computational and Structural Biotechnology Journal*, 19:3735–3746, 2021.

[50] Gustavo E. A. P. A. Batista, Ronaldo C. Prati, and Maria Carolina Monard. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29, jun 2004.

[51] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[52] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005.

[53] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. Ieee, 2008.

[54] Michal Krassowski, Vivek Das, Sangram K Sahu, and Biswapriya B Misra. State of the field in multi-omics research: from computational needs to data mining and sharing. *Frontiers in Genetics*, 11:610798, 2020.

[55] Sina Tabakhi, Mohammod Naimul Islam Suvon, Pegah Ahadian, and Haiping Lu. Multimodal learning for multi-omics: A survey. *World Scientific Annual Review of Artificial Intelligence*, 1:2250004, 2023.

[56] Nenad Trinajstic. *Chemical graph theory*. Routledge, 2018.

[57] Luke Hakes, John W Pinney, David L Robertson, and Simon C Lovell. Protein-protein interaction networks and biology-what's the connection? *Nature biotechnology*, 26(1):69–72, 2008.

[58] Eric Davidson and Michael Levin. Gene regulatory networks. *Proceedings of the National Academy of Sciences*, 102(14):4935–4935, 2005.

[59] Vincenzo Bonnici, Giorgio De Caro, Giorgio Constantino, Sabino Liuni, Domenica D'Elia, Nicola Bombieri, Flavio Licciulli, and Rosalba Giugno. Arena-Idb: a platform to build human non-coding RNA interaction networks. *BMC bioinformatics*, 19(10):350, 2018.

[60] Xiayu Xiang, Zhongru Wang, Yan Jia, and Binxing Fang. Knowledge graph-based clinical decision support system reasoning: a survey. In *2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC)*, pages 373–380. IEEE, 2019.

[61] Daniel S Himmelstein and Sergio E Baranzini. Heterogeneous network edge prediction: a data integration approach to prioritize disease-associated genes. *PLoS Comput Biol*, 11(7):e1004259, 2015.

[62] Jiansong Fang, Qihui Wu, Fei Ye, Chuipu Cai, Lvjie Xu, Yong Gu, Qi Wang, Ai-lin Liu, Wenjie Tan, and Guan-hua Du. Network-based identification and experimental validation of drug candidates toward sars-cov-2 via targeting virus–host interactome. *Frontiers in Genetics*, 12:1590, 2021.

[63] Tongxin Wang, Wei Shao, Zhi Huang, Haixu Tang, Jie Zhang, Zhengming Ding, and Kun Huang. Mogonet integrates multi-omics data using graph convolutional networks allowing patient classification and biomarker identification. *Nature Communications*, 12(1):1–13, 2021.

[64] Joshua A Grochow and Manolis Kellis. Network motif discovery using subgraph enumeration and symmetry-breaking. In *Annual International Conference on Research in Computational Molecular Biology*, pages 92–106. Springer, 2007.

[65] Fabio Rinnone, Giovanni Micale, Vincenzo Bonnici, Gary D Bader, Dennis Shasha, Alfredo Ferro, Alfredo Pulvirenti, and Rosalba Giugno. NetMatchStar: an enhanced Cytoscape network querying app. *F1000Research*, 4, 2015.

[66] Mehdi Sadeghi, Bryce Ordway, Ilyia Rafiei, Punit Borad, Bin Fang, John L Koomen, Chaomei Zhang, Sean Yoder, Joseph Johnson, and Mehdi Damaghi. Integrative analysis of breast cancer cells reveals an epithelial-mesenchymal transition role in adaptation to acidic microenvironment. *Frontiers in Oncology*, 10:304, 2020.

[67] Ngoc Tam L Tran, Sominder Mohan, Zhuoqing Xu, and Chun-Hsi Huang. Current innovations and future challenges of network motif detection. *Briefings in bioinformatics*, 16(3):497–525, 2015.

[68] Elisabeth Wong, Brittany Baur, Saad Quader, and Chun-Hsi Huang. Biological network motif detection: principles and practice. *Briefings in bioinformatics*, 13(2):202–215, 2012.

[69] Shai S Shen-Orr, Ron Milo, Shmoolik Mangan, and Uri Alon. Network motifs in the transcriptional regulation network of Escherichia coli. *Nature genetics*, 31(1):64–68, 2002.

[70] Jane Rosemary Allison and Ivan D Welsh. CherryPicker: An Algorithm for the Automated Parameterisation of Large Biomolecules for Molecular Simulation. *Frontiers in chemistry*, 7:400, 2019.

[71] Brian P Kelley, Bingbing Yuan, Fran Lewitter, Roded Sharan, Brent R Stockwell, and Trey Ideker. PathBLAST: a tool for alignment of protein interaction networks. *Nucleic acids research*, 32(suppl_2):W83–W88, 2004.

[72] Qingwu Yang and Sing-Hoi Sze. Path matching and graph matching in biological networks. *Journal of Computational Biology*, 14(1):56–67, 2007.

[73] Roded Sharan, Igor Ulitsky, and Ron Shamir. Network-based prediction of protein function. *Molecular systems biology*, 3(1):88, 2007.

[74] Giovanni Micale, Alfredo Pulvirenti, Rosalba Giugno, and Alfredo Ferro. GASOLINE: a greedy and stochastic algorithm for optimal local multiple alignment of interaction networks. *PloS one*, 9(6):e98750, 2014.

[75] Günhan Gülsoy and Tamer Kahveci. RINQ: Reference-based indexing for network queries. *Bioinformatics*, 27(13):i149–i158, 2011.

[76] Daniel Scott Himmelstein, Antoine Lizee, Christine Hessler, Leo Brueggeman, Sabrina L Chen, Dexter Hadley, Ari Green, Pouya Khankhanian, and Sergio E Baranzini. Systematic integration of biomedical knowledge prioritizes drugs for repurposing. *Elife*, 6:e26726, 2017.

[77] Valeria Fionda and Luigi Palopoli. Biological network querying techniques: analysis and comparison. *Journal of Computational Biology*, 18(4):595–625, 2011.

[78] Luigi Pietro Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. An improved algorithm for matching large graphs. In *3rd IAPR-TC15 workshop on graph-based representations in pattern recognition*, pages 149–159, 2001.

[79] Vincenzo Carletti, Pasquale Foggia, Alessia Saggese, and Mario Vento. Introducing VF3: A new algorithm for subgraph isomorphism. In *International Workshop on Graph-Based Representations in Pattern Recognition*, pages 128–139. Springer, 2017.

[80] Vincenzo Bonnici and Rosalba Giugno. On the variable ordering in subgraph isomorphism algorithms. *IEEE/ACM transactions on computational biology and bioinformatics*, 14(1):193–203, 2016.

[81] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, 1971.

[82] Huahai He and Ambuj K Singh. Graphs-at-a-time: query language and access methods for graph databases. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 405–418, 2008.

[83] Wook-Shin Han, Jinsoo Lee, and Jeong-Hoon Lee. Turboiso: towards ultrafast and robust subgraph isomorphism search in large graph databases. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 337–348, 2013.

[84] Fei Bi, Lijun Chang, Xuemin Lin, Lu Qin, and Wenjie Zhang. Efficient subgraph matching by postponing cartesian products. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1199–1214, 2016.

[85] Rosalba Giugno and Dennis Shasha. Graphgrep: A fast and universal method for querying graphs. In *Object recognition supported by user interaction for service robots*, volume 2, pages 112–115. IEEE, 2002.

[86] Vincenzo Bonnici, Alfredo Ferro, Rosalba Giugno, Alfredo Pulvirenti, and Dennis Shasha. Enhancing graph database indexing by suffix tree structure. In *IAPR International Conference on Pattern Recognition in Bioinformatics*, pages 195–203. Springer, 2010.

[87] Rosalba Giugno, Vincenzo Bonnici, Nicola Bombieri, Alfredo Pulvirenti, Alfredo Ferro, and Dennis Shasha. Grapes: A software for parallel searching on biological graphs targeting multi-core architectures. *PloS one*, 8(10), 2013.

[88] Raffaele Di Natale, Alfredo Ferro, Rosalba Giugno, Misael Mongiovì, Alfredo Pulvirenti, and Dennis Shasha. Sing: Subgraph search in non-homogeneous graphs. *BMC bioinformatics*, 11(1):96, 2010.

[89] Karsten Klein, Nils Kriege, and Petra Mutzel. CT-index: Fingerprint-based graph indexing combining cycles and trees. In *2011 IEEE 27th International Conference on Data Engineering*, pages 1115–1126. IEEE, 2011.

[90] David W Williams, Jun Huan, and Wei Wang. Graph database indexing using structured graph decomposition. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 976–985. IEEE, 2007.

[91] Lei Zou, Lei Chen, Jeffrey Xu Yu, and Yansheng Lu. A novel spectral coding in a large graph database. In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, pages 181–192, 2008.

[92] Haichuan Shang, Ying Zhang, Xuemin Lin, and Jeffrey Xu Yu. Taming verification hardness: an efficient algorithm for testing subgraph isomorphism. *Proceedings of the VLDB Endowment*, 1(1):364–375, 2008.

[93] Shijie Zhang, Meng Hu, and Jiong Yang. Treepi: A novel graph indexing method. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 966–975. IEEE, 2007.

[94] Peixiang Zhao, Jeffrey Xu Yu, and S Yu Philip. Graph indexing: Tree+ Delta¿= Graph. In *VLDB*, volume 7, pages 938–949, 2007.

[95] Yan Xie and Philip S Yu. CP-index: on the efficient indexing of large graphs. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1795–1804, 2011.

[96] Xifeng Yan, Philip S Yu, and Jiawei Han. Graph indexing: a frequent structure-based approach. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 335–346, 2004.

[97] James Cheng, Yiping Ke, Wilfred Ng, and An Lu. Fg-index: towards verification-free query processing on graph databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 857–872, 2007.

[98] Dayu Yuan and Prasenjit Mitra. Lindex: a lattice-based index for graph databases. *The VLDB Journal*, 22(2):229–252, 2013.

[99] Dipali Pal, Praveen Rao, Vasil Slavov, and Anas Katib. Fast processing of graph queries on a large database of small and medium-sized data graphs. *Journal of Computer and System Sciences*, 82(6):1112–1143, 2016.

[100] Foteini Katsarou, Nikos Ntarmos, and Peter Triantafillou. Hybrid algorithms for subgraph pattern queries in graph databases. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 656–665. IEEE, 2017.

[101] Shixuan Sun and Qiong Luo. Scaling Up Subgraph Query Processing with Efficient Subgraph Matching. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 220–231. IEEE, 2019.

[102] Jing Wang, Nikos Ntarmos, and Peter Triantafillou. Indexing query graphs to speed up graph query processing. *Proceedings of the 19th International Conference on Extending Database Technology (EDBT)*, 2016.

[103] Jing Wang, Nikos Ntarmos, and Peter Triantafillou. GraphCache: a caching system for graph queries. *Proceedings of the 20th International Conference on Extending Database Technology (EDBT)*, 2017.

[104] Karam Gouda and Mosab Hassaan. Compressed feature-based filtering and verification approach for subgraph search. In *Proceedings of the 16th*

International Conference on Extending Database Technology, pages 287–298, 2013.

[105] David Luaces, José RR Viqueira, Tomás F Pena, and José Manuel Cotos. Leveraging Bitmap Indexing for Subgraph Searching. In *EDBT*, pages 49–60, 2019.

[106] Foteini Katsarou, Nikos Ntarmos, and Peter Triantafillou. Performance and scalability of indexed subgraph query processing methods. *Proceedings of the VLDB Endowment*, 8(12):1566–1577, 2015.

[107] Malathi Veeraraghavan and Kishor S Trivedi. An improved algorithm for the symbolic reliability analysis of networks. In *Proceedings Ninth Symposium on Reliable Distributed Systems*, pages 34–43. IEEE, 1990.

[108] Gianfranco Ciardo, Yang Zhao, and Xiaoqing Jin. Ten years of saturation: A Petri net perspective. In *Transactions on Petri Nets and Other Models of Concurrency V*, pages 51–95. Springer, 2012.

[109] B. Bollig and I. Wegener. Improving the variable ordering of OBDDs is NP-complete. *IEEE Transactions on computers*, 45(9):993–1002, 1996.

[110] Masahiro Fujita, Yusuke Matsunaga, and Taeko Kakuda. On variable ordering of binary decision diagrams for the application of multi-level logic synthesis. In *Proceedings of the European Conference on Design Automation.*, pages 50–54. IEEE, 1991.

[111] Masahiro Fujita, Hisanori Fujisawa, and Yusuke Matsunaga. Variable ordering algorithms for ordered binary decision diagrams and their evaluation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(1):6–12, 1993.

[112] Elvio Gilberto Amparore, Susanna Donatelli, Marco Beccuti, Giulio Garbi, Andrew Miner, et al. Decision diagrams for petri nets: which variable ordering? In *CEUR WORKSHOP PROCEEDINGS*, volume 1846, pages 31–50. CEUR, 2017.

[113] Elvio Gilberto Amparore, Susanna Donatelli, Marco Beccuti, Giulio Garbi, and Andrew Miner. Decision diagrams for petri nets: a comparison of variable ordering algorithms. *Transactions on Petri Nets and Other Models of Concurrency XIII*, pages 73–92, 2018.

[114] Fabio Somenzi. CUDD: CU decision diagram package release 2.3. 0. *University of Colorado at Boulder*, 1998.

[115] Yann Thierry-Mieg, Denis Poitrenaud, Alexandre Hamez, and Fabrice Kordon. Hierarchical Set Decision Diagrams and Regular Models. In Stefan Kowalewski and Anna Philippou, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 1–15, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[116] Junaid Babar and Andrew Miner. Meddly: Multi-terminal and edge-valued decision diagram library. In *2010 Seventh International Conference on the Quantitative Evaluation of Systems*, pages 195–196. IEEE, 2010.

[117] Tom Van Dijk and Jaco Van de Pol. Sylvan: multi-core framework for decision diagrams. *International Journal on Software Tools for Technology Transfer*, 19:675–696, 2017.

[118] Karl S Brace, Richard L Rudell, and Randal E Bryant. Efficient implementation of a BDD package. In *27th ACM/IEEE design automation conference*, pages 40–45. IEEE, 1990.

[119] Thomas Weise. Global optimization algorithms-theory and application. *Self-Published Thomas Weise*, 361, 2009.

[120] John H Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[121] Nicola Licheri, Vincenzo Bonnici, Marco Beccuti, and Rosalba Giugno. GRAPES-DD: exploiting decision diagrams for index-driven search in biological graph databases. *BMC bioinformatics*, 22(1):1–24, 2021.

[122] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

[123] Barbara Pardini, Giulio Ferrero, Sonia Tarallo, Gaetano Gallo, Antonio Francavilla, Nicola Licheri, Mario Trompetto, Giuseppe Clerico, Carlo

Senore, Sergio Peyre, et al. A fecal mirna signature by small rna sequencing accurately distinguishes colorectal cancers: results from a multicentric study. *Gastroenterology*, 2023.

[124] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[125] Tiago P. Peixoto. The graph-tool python library. *figshare*, 2014.

[126] *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley Longman Publishing Co., Inc., USA, 2002.

[127] David Freedman, Robert Pisani, and Roger Purves. Statistics (international student edition). *Pisani, R. Purves, 4th edn. WW Norton & Company, New York*, 2007.

[128] S Das Gupta. Point biserial correlation coefficient and its generalization. *Psychometrika*, 25(4):393–408, 1960.

[129] Harald Cramér. *Mathematical methods of statistics*, volume 43. Princeton university press, 1999.

[130] Wayne W Daniel. The spearman rank correlation coefficient. *Biostatistics: A Foundation for Analysis in the Health Sciences*, 1987.

[131] Roger B Newson et al. Somers' d: A common currency for associations. In *United Kingdom Stata Users' Group Meetings*, number 01, 2015.

[132] Henry Scheffe. *The analysis of variance*, volume 72. John Wiley & Sons, 1999.

[133] Joy A Thomas and TM Cover. Elements of information theory. *John Wiley & Sons, Inc., New York. Toni, T., Welch, D., Strelkowa, N., Ipsen, A., and Stumpf, MPH (2009),"Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems," Journal of the Royal Society Interface*, 6:187–202, 1991.

[134] Student. The probable error of a mean. *Biometrika*, 6(1):1–25, 1908.

[135] Davide Chicco and Giuseppe Jurman. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):1–13, 2020.

[136] Joy Paul Guilford. Psychometric methods. 1954.

[137] Jan Salomon Cramer. The origins of logistic regression. 2002.

[138] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232, 1958.

[139] Ronan Gautier, Gregoire Jaffre, and Bibi Ndiaye. Skope Rules: Machine Learning with Logical Rules in Python, 2017.

[140] Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022.

[141] Nicola Licheri, Elvio Amparone, Vincenzo Bonnici, Rosalba Giugno, and Marco Beccuti. An entropy heuristic to optimize decision diagrams for index-driven search in biological graph databases. In *CIKM Workshops*, 2021.

[142] NaNa Keum and Edward Giovannucci. Global burden of colorectal cancer: emerging trends, risk factors and prevention strategies. *Nature reviews Gastroenterology & hepatology*, 16(12):713–732, 2019.

[143] D Maxwell Parkin. Global cancer statistics in the year 2000. *The lancet oncology*, 2(9):533–543, 2001.

[144] Jacques Ferlay, Murielle Colombet, Isabelle Soerjomataram, Colin Mathers, Donald M Parkin, Marlon Piñeros, Ariana Znaor, and Freddie Bray. Estimating the global cancer incidence and mortality in 2018: Globocan sources and methods. *International journal of cancer*, 144(8):1941–1953, 2019.

[145] J Kral, V Kojecky, M Stepan, M Vladarova, O Zela, J Knot, M Jakovljevic, Z Kralova, R Buresova, T Grega, et al. The experience with colorectal cancer screening in the czech republic: the detection at earlier stages and improved clinical outcomes. *Public health*, 185:153–158, 2020.

[146] Béatrice Lauby-Secretan, Nadia Vilahur, Franca Bianchini, Neela Guha, and Kurt Straif. The iarc perspective on colorectal cancer screening. *New England Journal of Medicine*, 378(18):1734–1740, 2018.

[147] Carlo Senore, Partha Basu, Ahti Anttila, Antonio Ponti, Mariano Tomatis, Diama Bhadra Vale, Gugliemo Ronco, Isabelle Soerjomataram, Maja Primic-Žakelj, Emilia Riggi, et al. Performance of colorectal cancer screening in the european union member states: data from the second european screening report. *Gut*, 68(7):1232–1244, 2019.

[148] Linda Rabeneck, Han-Mo Chiu, and Carlo Senore. International perspective on the burden of colorectal cancer and public health effects. *Gastroenterology*, 158(2):447–452, 2020.

[149] Douglas J Robertson, Jeffrey K Lee, C Richard Boland, Jason A Dominitz, Francis M Giardiello, David A Johnson, Tonya Kaltenbach, David Lieberman, Theodore R Levin, and Douglas K Rex. Recommendations on fecal immunochemical testing to screen for colorectal neoplasia: a consensus statement by the us multi-society task force on colorectal cancer. *Gastroenterology*, 152(5):1217–1237, 2017.

[150] Alexandre Loktionov. Biomarkers for detecting colorectal cancer non-invasively: Dna, rna or proteins? *World journal of gastrointestinal oncology*, 12(2):124, 2020.

[151] Mingjiao Weng, Di Wu, Chao Yang, Haisheng Peng, Guangyu Wang, Tianzhen Wang, and Xiaobo Li. Noncoding rnas in the development, diagnosis, and prognosis of colorectal cancer. *Translational Research*, 181:108–120, 2017.

[152] Andrew Maltez Thomas, Paolo Manghi, Francesco Asnicar, Edoardo Pasolli, Federica Armanini, Moreno Zolfo, Francesco Beghini, Serena Manara, Nicolai Karcher, Chiara Pozzi, et al. Metagenomic analysis of colorectal cancer datasets identifies cross-cohort microbial diagnostic signatures and a link with choline degradation. *Nature medicine*, 25(4):667–678, 2019.

[153] Yulin Sun, Zhengguang Guo, Xiaoyan Liu, Lijun Yang, Zongpan Jing, Meng Cai, Zhaoxu Zheng, Chen Shao, Yefan Zhang, Haidan Sun, et al.

Noninvasive urinary protein signatures associated with colorectal cancer diagnosis and metastasis. *Nature communications*, 13(1):2757, 2022.

[154] Antonio Francavilla, Szimonetta Turoczi, Sonia Tarallo, Pavel Vodicka, Barbara Pardini, and Alessio Naccarati. Exosomal microRNAs and other non-coding RNAs as colorectal cancer biomarkers: a review. *Mutagenesis*, 35(3):243–260, 2020.

[155] Sonja Hombach and Markus Kretz. Non-coding rnas: classification, biology and functioning. *Non-coding RNAs in colorectal cancer*, pages 3–17, 2016.

[156] Gianpiero Di Leva and Carlo M Croce. mirna profiling of cancer. *Current opinion in genetics & development*, 23(1):3–11, 2013.

[157] Abdullah Moridikia, Hamed Mirzaei, Amirhossein Sahebkar, and Jafar Salimian. Micrornas: Potential candidates for diagnosis and treatment of colorectal cancer. *Journal of cellular physiology*, 233(2):901–913, 2018.

[158] Mihnea Paul Dragomir, Scott Kopetz, Jaffer A Ajani, and George Adrian Calin. Non-coding rnas in gi cancers: from cancer hallmarks to clinical utility. *Gut*, 69(4):748–763, 2020.

[159] Barbara Pardini, Alexandru Anton Sabo, Giovanni Birolo, and George Adrian Calin. Noncoding rnas in extracellular fluids as cancer biomarkers: the new frontier of liquid biopsies. *Cancers*, 11(8):1170, 2019.

[160] Klara Cervena, Vendula Novosadova, Barbara Pardini, Alessio Naccarati, Alena Opattova, Josef Horak, Sona Vodenkova, Tomas Buchler, Pavel Skrobanek, Miroslav Levy, et al. Analysis of microrna expression changes during the course of therapy in rectal cancer patients. *Frontiers in oncology*, 11:702258, 2021.

[161] Sonia Tarallo, Giulio Ferrero, Gaetano Gallo, Antonio Francavilla, Giuseppe Clerico, Alberto Realis Luc, Paolo Manghi, Andrew Maltez Thomas, Paolo Vineis, Nicola Segata, et al. Altered fecal small RNA profiles in colorectal cancer reflect gut microbiome composition in stool samples. *Msystems*, 4(5):10–1128, 2019.

[162] Saray Duran-Sanchon, Lorena Moreno, Josep M Augé, Miquel Serra-Burriel, Míriam Cuatrecasas, Leticia Moreira, Agatha Martín, Anna Serradesanferm, Àngels Pozo, Rosa Costa, et al. Identification and validation of microrna profiles in fecal samples for detection of colorectal cancer. *Gastroenterology*, 158(4):947–957, 2020.

[163] Zitong Zhao, Anna Zhu, Megha Bhardwaj, Petra Schrotz-King, and Hermann Brenner. Fecal micrornas, fecal microrna panels, or combinations of fecal micrornas with fecal hemoglobin for early detection of colorectal cancer and its precursors: A systematic review. *Cancers*, 14(1):65, 2021.

[164] Antonio Francavilla, Sonia Tarallo, Barbara Pardini, and Alessio Naccarati. Fecal microRNAs as non-invasive biomarkers for the detection of colorectal cancer: A systematic review. *Minerva Biotecnologica*, 31(1):3–10, 2019.

[165] Sonia Tarallo, Giulio Ferrero, Francesca De Filippis, Antonio Francavilla, Edoardo Pasolli, Valentina Panero, Francesca Cordero, Nicola Segata, Sara Grioni, Ruggero Gaetano Pensa, et al. Stool microrna profiles reflect different dietary and gut microbiome patterns in healthy individuals. *Gut*, 71(7):1302–1314, 2022.

[166] Antonio Francavilla, Amedeo Gagliardi, Giulia Piaggeschi, Sonia Tarallo, Francesca Cordero, Ruggero G Pensa, Alessia Impeduglia, Gian Paolo Caviglia, Davide Giuseppe Ribaldone, Gaetano Gallo, et al. Faecal mirna profiles associated with age, sex, bmi, and lifestyle habits in healthy individuals. *Scientific Reports*, 11(1):20645, 2021.

[167] Ana E Jenike and Marc K Halushka. mir-21: a non-specific biomarker of all maladies. *Biomarker research*, 9(1):1–7, 2021.

[168] Thomas M Zarchy and Daniel Ershoff. Do characteristics of adenomas on flexible sigmoidoscopy predict advanced lesions on baseline colonoscopy? *Gastroenterology*, 106(6):1501–1504, 1994.

[169] Andrew Maltez Thomas, Paolo Manghi, Francesco Asnicar, Edoardo Pasolli, Federica Armanini, Moreno Zolfo, Francesco Beghini, Serena Manara, Nicolai Karcher, Chiara Pozzi, et al. Metagenomic analysis

of colorectal cancer datasets identifies cross-cohort microbial diagnostic signatures and a link with choline degradation. *Nature medicine*, 25(4):667–678, 2019.

[170] Jakob Wirbel, Paul Theodor Pyl, Ece Kartal, Konrad Zych, Alireza Kashani, Alessio Milanese, Jonas S Fleck, Anita Y Voigt, Albert Palleja, Ruby Ponnudurai, et al. Meta-analysis of fecal metagenomes reveals global microbial signatures that are specific for colorectal cancer. *Nature medicine*, 25(4):679–689, 2019.

[171] Barbora Zwinsová, Vyacheslav A Petrov, Martina Hrivňáková, Stanislav Smatana, Lenka Micenková, Natálie Kazdová, Vlad Popovici, Roman Hrstka, Roman Šefr, Beatrix Bencsiková, et al. Colorectal tumour mucosa microbiome is enriched in oral pathogens and defines three subtypes that correlate with markers of tumour progression. *Cancers*, 13(19):4799, 2021.

[172] Antonio Francavilla, Giulio Ferrero, Barbara Pardini, Sonia Tarallo, Laura Zanatto, Gian Paolo Caviglia, Sabina Sieri, Sara Grioni, Giulia Francescato, Francesco Stalla, et al. Gluten-free diet affects fecal small non-coding rna profiles and microbiome composition in celiac disease supporting a host-gut microbiota crosstalk. *Gut Microbes*, 15(1):2172955, 2023.

[173] Sonia Tarallo, Giulio Ferrero, Gaetano Gallo, Antonio Francavilla, Giuseppe Clerico, Alberto Realis Luc, Paolo Manghi, Andrew Maltez Thomas, Paolo Vineis, Nicola Segata, et al. Altered fecal small rna profiles in colorectal cancer reflect gut microbiome composition in stool samples. *Msystems*, 4(5):10–1128, 2019.

[174] Tudor Moisoiu, Mihnea P Dragomir, Stefania D Iancu, Simon Schallenberg, Giovanni Birolo, Giulio Ferrero, Dan Burghelea, Andrei Stefancu, Ramona G Cozan, Emilia Licarete, et al. Combined mirna and sers urine liquid biopsy for the point-of-care diagnosis and molecular stratification of bladder cancer. *Molecular Medicine*, 28(1):39, 2022.

[175] Huber Love and W Huber. Anders (2014). moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, 15(12):550.

[176] Jing Zhang and Kenneth B Storey. Rbiomirgs: an all-in-one mirna gene set analysis solution featuring target mrna mapping and expression profile integration. *PeerJ*, 6:e4262, 2018.

[177] Ondrej Slaby. *Non-coding RNAs as biomarkers for colorectal cancer screening and early detection*. Springer, 2016.

[178] Julia Alles, Tobias Fehlmann, Ulrike Fischer, Christina Backes, Valentina Galata, Marie Minet, Martin Hart, Masood Abu-Halima, Friedrich A Grässer, Hans-Peter Lenhof, et al. An estimate of the total number of true human mirnas. *Nucleic acids research*, 47(7):3353–3364, 2019.

[179] Dereje D Jima, Jenny Zhang, Cassandra Jacobs, Kristy L Richards, Cherie H Dunphy, William WL Choi, Wing Yan Au, Gopesh Srivastava, Magdalena B Czader, David A Rizzieri, et al. Deep sequencing of the small rna transcriptome of normal and malignant human b cells identifies hundreds of novel microrna. *Blood, The Journal of the American Society of Hematology*, 116(23):e118–e127, 2010.

[180] Marc R Friedländer, Esther Lizano, Anna JS Houben, Daniela Bezdan, Mónica Báñez-Coronel, Grzegorz Kudla, Elisabet Mateu-Huertas, Birgit Kagerbauer, Justo González, Kevin C Chen, et al. Evidence for the biogenesis of more than 1,000 novel human microrna. *Genome biology*, 15:1–17, 2014.

[181] Esther K Wei, Edward Giovannucci, Kana Wu, Bernard Rosner, Charles S Fuchs, Walter C Willett, and Graham A Colditz. Comparison of risk factors for colon and rectal cancer. *International journal of cancer*, 108(3):433–442, 2004.

[182] Fumiaki Imamura, Renata Micha, Shahab Khatibzadeh, Saman Fahimi, Peilin Shi, John Powles, and Dariush Mozaffarian. Dietary quality among men and women in 187 countries in 1990 and 2010: a systematic assessment. *The lancet global health*, 3(3):e132–e142, 2015.

[183] Martin CS Wong, Junjie Huang, Veeleah Lok, Jingxuan Wang, Franklin Fung, Hanyue Ding, and Zhi-Jie Zheng. Differences in incidence and mortality trends of colorectal cancer worldwide based on sex, age, and

anatomic location. *Clinical Gastroenterology and Hepatology*, 19(5):955–966, 2021.

[184] Fanny ER Vuik, Stella AV Nieuwenburg, Marc Bardou, Iris Lansdorp-Vogelaar, Mário Dinis-Ribeiro, Maria J Bento, Vesna Zadnik, María Pellisé, Laura Esteban, Michal F Kaminski, et al. Increasing incidence of colorectal cancer in young adults in europe over the last 25 years. *Gut*, 68(10):1820–1826, 2019.

[185] Swati G Patel, Jordan J Karlitz, Timothy Yen, Christopher H Lieu, and C Richard Boland. The rising tide of early-onset colorectal cancer: a comprehensive review of epidemiology, clinical features, biology, risk factors, prevention, and early detection. *The lancet Gastroenterology & hepatology*, 7(3):262–274, 2022.

[186] Brendan J Desmond, Elizabeth R Dennett, and Kirsty M Danielson. Circulating extracellular vesicle microrna as diagnostic biomarkers in early colorectal cancer—a review. *Cancers*, 12(1):52, 2019.

[187] Tomer Cooks, Ioannis S Pateras, Lisa M Jenkins, Keval M Patel, Ana I Robles, James Morris, Tim Forshew, Ettore Appella, Vassilis G Gorgoulis, and Curtis C Harris. Mutant p53 cancers reprogram macrophages to tumor supporting macrophages via exosomal mir-1246. *Nature communications*, 9(1):771, 2018.

[188] Songhe Guo, Jun Chen, Fangfang Chen, Qiuyao Zeng, Wan-Li Liu, and Ge Zhang. Exosomes derived from fusobacterium nucleatum-infected colorectal cancer cells facilitate tumour metastasis by selectively carrying mir-1246/92b-3p/27a-3p and cxcl16. *Gut*, 70(8):1507–1519, 2021.

[189] Aikun Fu, Bingqing Yao, Tingting Dong, and Shang Cai. Emerging roles of intratumor microbiota in cancer metastasis. *Trends in Cell Biology*, 2022.

[190] Yingying Cao, Zhenhua Wang, Yuqing Yan, Linhua Ji, Jie He, Baoqin Xuan, Chaoqin Shen, Yanru Ma, Shanshan Jiang, Dan Ma, et al. Enterotoxigenic bacteroides fragilis promotes intestinal inflammation and malignancy by inhibiting exosome-packaged mir-149-3p. *Gastroenterology*, 161(5):1552–1566, 2021.

[191] Slater L Clay, Diogo Fonseca-Pereira, Wendy S Garrett, et al. Colorectal cancer: the facts in the case of the microbiota. *The Journal of Clinical Investigation*, 132(4), 2022.

[192] TCGA Network, MN Bainbridge, K Chang, HH Dinh, JA Drummond, G Fowler, et al. Comprehensive molecular characterization of human colon and rectal cancer. nat 2012 4877407, 2012.

[193] Aurora Esquela-Kerscher and Frank J Slack. Oncomirs—micrornas with a role in cancer. *Nature reviews cancer*, 6(4):259–269, 2006.

[194] Elena Vila-Navarro, Maria Vila-Casadesús, Leticia Moreira, Saray Duran-Sanchon, Rupal Sinha, Àngels Ginés, Glòria Fernández-Esparrach, Rosa Miquel, Miriam Cuatrecasas, Antoni Castells, et al. Micrornas for detection of pancreatic neoplasia: biomarker discovery by next-generation sequencing and validation in 2 independent cohorts. *Annals of surgery*, 265(6):1226, 2017.

[195] Yuanyuan Liang, Shun Li, and Liling Tang. Microrna 320, an anti-oncogene target mirna for cancer therapy. *Biomedicines*, 9(6):591, 2021.

[196] Friederike Cordes, Claudia Demmig, Arne Bokemeyer, Markus Brückner, Frank Lenze, Philipp Lenz, Tobias Nowacki, Phil Tepasse, Hartmut H Schmidt, M Alexander Schmidt, et al. Microrna-320a monitors intestinal disease activity in patients with inflammatory bowel disease. *Clinical and Translational Gastroenterology*, 11(3), 2020.

[197] Stephanie Muenchau, Rosalie Deutsch, Ines J de Castro, Thomas Hielscher, Nora Heber, Beate Niesler, Marina Lusic, Megan L Stanifer, and Steeve Boulant. Hypoxic environment promotes barrier formation in human intestinal epithelial cells through regulation of microrna 320a expression. *Molecular and cellular biology*, 39(14):e00553–18, 2019.

[198] Blair B Madison, Arjun N Jeganathan, Rei Mizuno, Monte M Winslow, Antoni Castells, Miriam Cuatrecasas, and Anil K Rustgi. Let-7 represses carcinogenesis and a stem cell phenotype in the intestine via regulation of hmga2. *PLoS genetics*, 11(8):e1005408, 2015.

[199] Christian T Wohnhaas, Ramona Schmid, Marcel Rolser, Eric Kaaru, Dominik Langgartner, Kathrin Rieber, Benjamin Strobel, Claudia

Eisele, Franziska Wiech, Ines Jakob, et al. Fecal micrornas show promise as noninvasive crohn's disease biomarkers. *Crohn's & colitis 360*, 2(1):otaa003, 2020.

[200] Julien Verdier, Irene Raphaela Breunig, Margarete Clara Ohse, Silvia Roubrocks, Sandra Kleinfeld, Sanchari Roy, Konrad Streetz, Christian Trautwein, Christoph Roderburg, and Gernot Sellge. Faecal micrornas in inflammatory bowel diseases. *Journal of Crohn's and Colitis*, 14(1):110–117, 2020.

[201] Filip Ambrozkiewicz, Jakub Karczmarski, Maria Kulecka, Agnieszka Paziewska, Magdalena Niemira, Natalia Zeber-Lubecka, Edyta Zagorowicz, Adam Kretowski, and Jerzy Ostrowski. In search for interplay between stool micrornas, microbiota and short chain fatty acids in crohn's disease-a preliminary study. *BMC gastroenterology*, 20(1):1–18, 2020.

[202] Yuan-Hong Xie, Qin-Yan Gao, Guo-Xiang Cai, Xiao-Ming Sun, Tian-Hui Zou, Hui-Min Chen, Si-Yi Yu, Yi-Wen Qiu, Wei-Qi Gu, Xiao-Yu Chen, et al. Fecal clostridium symbiosum for noninvasive detection of early and advanced colorectal cancer: test and validation studies. *EBioMedicine*, 25:32–40, 2017.

[203] Linda JW Bosch, Frank A Oort, Maarten Neerincx, Carolina AJ Khalid-de Bakker, Jochim S Terhaar sive Droste, Veerle Melotte, Daisy MAE Jonkers, Ad AM Masclee, Sandra Mongera, Madeleine Grooteclaes, et al. Dna methylation of phosphatase and actin regulator 3 detects colorectal cancer in stool and complements fit. *Cancer prevention research*, 5(3):464–472, 2012.

[204] Thomas F Imperiale, David F Ransohoff, Steven H Itzkowitz, Theodore R Levin, Philip Lavin, Graham P Lidgard, David A Ahlquist, and Barry M Berger. Multitarget stool dna testing for colorectal-cancer screening. *New England Journal of Medicine*, 370(14):1287–1297, 2014.

[205] Christophe H Schilling et al. Combining pathway analysis with flux balance analysis for the comprehensive study of metabolic systems. *Biotechnology and bioengineering*, 71(4):286–306, 2000.

[206] Weihua Guo and Xueyang Feng. Om-fba: integrate transcriptomics data with flux balance analysis to decipher the cell metabolism. *PloS one*, 11(4):e0154188, 2016.

[207] Meng Zhang et al. Metabolism-associated molecular classification of colorectal cancer. *Frontiers in Oncology*, 10:602498, 2020.

[208] Elizabeth Brunk et al. Recon3D enables a three-dimensional view of gene variation in human metabolism. *Nature biotechnology*, 36(3):272–281, 2018.

[209] M. R. Watson. Metabolic maps for the Apple II. *Biochemical Society Transactions*, 12(6):1093–1094, 12 1984.

[210] Sara Saheb Kashaf et al. Making life difficult for clostridium difficile: augmenting the pathogen's metabolic model with transcriptomic and codon usage data for better therapeutic target characterization. *BMC systems biology*, 11(1):1–13, 2017.

[211] Zhenling Wang et al. Machine learning-based glycolysis-associated molecular classification reveals differences in prognosis, TME, and immunotherapy for colorectal cancer patients. *Frontiers in Immunology*, 14, 05 2023.

[212] Jong Min Lee et al. Flux balance analysis in the era of metabolomics. *Briefings in Bioinformatics*, 7(2):140–150, 04 2006.

[213] The Istanbul consensus workshop on embryo assessment: proceedings of an expert meeting. *Human reproduction*, 26(6):1270–1283, 2011.

[214] Ashleigh Storr, Christos A Venetis, Simon Cooke, Suha Kilani, and William Ledger. Inter-observer and intra-observer agreement between embryologists during selection of a single day 5 embryo for transfer: a multicenter study. *Human Reproduction*, 32(2):307–314, 2017.

[215] Connie C Wong, Kevin E Loewke, Nancy L Bossert, Barry Behr, Christopher J De Jonge, Thomas M Baer, and Renee A Reijo Pera. Non-invasive imaging of human embryos before embryonic genome activation predicts development to the blastocyst stage. *Nature biotechnology*, 28(10):1115–1121, 2010.

[216] ESHRE Working Group on Time-Lapse Technology, Susanna Apter, Thomas Ebner, Thomas Freour, Yves Guns, Borut Kovacic, Nathalie Le Clef, Monica Marques, Marcos Meseguer, Debbie Montjean, et al. Good practice recommendations for the use of time-lapse technology. *Human Reproduction Open*, 2020(2):hoaa008, 2020.

[217] Csaba Pribenszky, Anna-Maria Nilselid, and Markus Montag. Time-lapse culture with morphokinetic embryo selection improves pregnancy and live birth chances and reduces early pregnancy loss: a meta-analysis. *Reproductive biomedicine online*, 35(5):511–520, 2017.

[218] Marcos Meseguer, Javier Herrero, Alberto Tejera, Karen Marie Hilligsøe, Niels Birger Ramsing, and Jose Remohí. The use of morphokinetics as a predictor of embryo implantation. *Human reproduction*, 26(10):2658–2671, 2011.

[219] Lukasz T Polanski, MA Coelho Neto, Carolina O Nastri, Paula A Navarro, Rui Alberto Ferriani, Nick Raine-Fenning, and Wellington P Martins. Time-lapse embryo imaging for improving reproductive outcomes: systematic review and meta-analysis. *Ultrasound in Obstetrics & Gynecology*, 44(4):394–401, 2014.

[220] Sarah Armstrong, Priya Bhide, Vanessa Jordan, Allan Pacey, Jane Marjoribanks, and Cindy Farquhar. Time-lapse systems for embryo incubation and assessment in assisted reproduction. *Cochrane Database of Systematic Reviews*, (5), 2019.

[221] Catherine Racowsky, Peter Kovacs, and Wellington P Martins. A critical appraisal of time-lapse imaging for embryo selection: where are we and where do we need to go? *Journal of assisted reproduction and genetics*, 32:1025–1030, 2015.

[222] Minghao Chen, Shiyou Wei, Junyan Hu, Jing Yuan, and Fenghua Liu. Does time-lapse imaging have favorable results for embryo incubation and selection compared with conventional methods in clinical in vitro fertilization? a meta-analysis and systematic review of randomized controlled trials. *PloS one*, 12(6):e0178720, 2017.

[223] Jason E Swain. Controversies in art: considerations and risks for uninterrupted embryo culture. *Reproductive BioMedicine Online*, 39(1):19–26, 2019.

[224] Nikica Zaninovic, Mohamad Irani, and Marcos Meseguer. Assessment of embryo morphology and developmental dynamics by time-lapse microscopy: is there a relation to implantation and ploidy? *Fertility and Sterility*, 108(5):722–729, 2017.

[225] Renjie Wang, Wei Pan, Lei Jin, Yuehan Li, Yudi Geng, Chun Gao, Gang Chen, Hui Wang, Ding Ma, and Shujie Liao. Artificial intelligence in reproductive medicine. *Reproduction*, 158(4):R139–R154, 2019.

[226] Nikica Zaninovic and Zev Rosenwaks. Artificial intelligence in human in vitro fertilization and embryology. *Fertility and Sterility*, 114(5):914–920, 2020.

[227] Claudio Manna, Loris Nanni, Alessandra Lumini, and Sebastiana Pappalardo. Artificial intelligence techniques for embryo and oocyte classification. *Reproductive biomedicine online*, 26(1):42–49, 2013.

[228] Irene Dimitriadis, Nikica Zaninovic, Alejandro Chavez Badiola, and Charles L Bormann. Artificial intelligence in the embryology laboratory: a review. *Reproductive biomedicine online*, 44(3):435–448, 2022.

[229] Behnaz Raef and Reza Ferdousi. A review of machine learning approaches in assisted reproductive technologies. *Acta Informatica Medica*, 27(3):205, 2019.

[230] Sandrine Giscard d'Estaing, Elsa Labrune, Maxence Forcellini, Cecile Edel, Bruno Salle, Jacqueline Lornage, and Mehdi Benchaib. A machine learning system with reinforcement capacity for predicting the fate of an art embryo. *Systems Biology in Reproductive Medicine*, 67(1):64–78, 2021.

[231] Alberto Revelli, Valentina Rovei, Paola Dalmasso, Gianluca Gennarelli, C Racca, Francesca Evangelista, and C Benedetto. Large randomized trial comparing transabdominal ultrasound-guided embryo transfer with a technique based on uterine length measurement before embryo transfer, 2016.

[232] Malin Huber, Nermin Hadziosmanovic, Lars Berglund, and Jan Holte. Using the ovarian sensitivity index to define poor, normal, and high response after controlled ovarian hyperstimulation in the long gonadotropin-releasing hormone-agonist protocol: suggestions for a new principle to solve an old problem. *Fertility and sterility*, 100(5):1270–1276, 2013.

[233] Stefano Canosa, Carlotta Paschero, Andrea Carosso, Sara Leoncini, Noemi Mercaldo, Gianluca Gennarelli, Chiara Benedetto, and Alberto Revelli. Effect of a combination of myo-inositol, alpha-lipoic acid, and folic acid on oocyte morphology and embryo morphokinetics in non-pcos overweight/obese patients undergoing ivf: a pilot, prospective, randomized study. *Journal of Clinical Medicine*, 9(9):2949, 2020.

[234] Jan Holte, Lars Berglund, K Milton, C Garello, Gianluca Gennarelli, Alberto Revelli, and Torbjörn Bergh. Construction of an evidence-based integrated morphology cleavage embryo score for implantation potential of embryos scored and transferred on day 2 after oocyte retrieval. *Human Reproduction*, 22(2):548–557, 2007.

[235] Stefano Canosa, Loredana Bergandi, Chiara Macrì, Lorena Charrier, Carlotta Paschero, Andrea Carosso, Noemi Di Segni, Francesca Silvagno, Gianluca Gennarelli, Chiara Benedetto, et al. Morphokinetic analysis of cleavage stage embryos and assessment of specific gene expression in cumulus cells independently predict human embryo development to expanded blastocyst: a preliminary study. *Journal of Assisted Reproduction and Genetics*, 37:1409–1420, 2020.

[236] H Nadir Ciray, Alison Campbell, Inge Errebo Agerholm, Jesus Aguilar, Sandrine Chamayou, Marga Esbert, and Shabana Sayed. Proposed guidelines on the nomenclature and annotation of dynamic human embryo monitoring by a time-lapse user group. *Human reproduction*, 29(12):2650–2660, 2014.

[237] Bruce S Shapiro, Kevin S Richter, Dee C Harris, and Said T Daneshmand. A comparison of day 5 and day 6 blastocyst transfers. *Fertility and sterility*, 75(6):1126–1130, 2001.

[238] Gorka Barrenetxea, Arantza López de Larruzea, Teresa Ganzabal, Rosario Jiménez, Koldo Carbonero, and Miren Mandiola. Blastocyst culture after repeated failure of cleavage-stage embryo transfers: a comparison of day 5 and day 6 transfers. *Fertility and sterility*, 83(1):49–53, 2005.

[239] Jason M Franasiak, Eric J Forman, George Patounakis, Kathleen H Hong, Marie D Werner, Kathleen M Upham, Nathan R Treff, and Richard T Scott Jr. Investigating the impact of the timing of blastulation on implantation: management of embryo-endometrial synchrony improves outcomes. *Human reproduction open*, 2018(4):hoy022, 2018.

[240] Joe Conaghan, Alice A Chen, Susan P Willman, Kristen Ivani, Philip E Chenette, Robert Boostanfar, Valerie L Baker, G David Adamson, Mary E Abusief, Marina Gvakharia, et al. Improving embryo selection using a computer-automated time-lapse image analysis test plus day 3 morphology: results from a prospective multicenter trial. *Fertility and sterility*, 100(2):412–419, 2013.

[241] Yamileth Motato, María José de los Santos, María José Escriba, Belén Aparicio Ruiz, José Remohí, and Marcos Meseguer. Morphokinetic analysis and embryonic prediction for blastocyst formation through an integrated time-lapse system. *Fertility and sterility*, 105(2):376–384, 2016.

[242] Liubin Yang, Mary Peavey, Khalied Kaskar, Neil Chappell, Lynn Zhu, Darius Devlin, Cecilia Valdes, Amy Schutt, Terri Woodard, Paul Zarutskie, et al. Development of a dynamic machine learning algorithm to predict clinical pregnancy and live birth rate with embryo morphokinetics. *F&S Reports*, 3(2):116–123, 2022.

[243] M VerMilyea, JMM Hall, SM Diakiw, A Johnston, T Nguyen, D Perugini, A Miller, A Picou, AP Murphy, and M Perugini. Development of an artificial intelligence-based assessment model for prediction of embryo viability using static images captured by optical light microscopy during ivf. *Human Reproduction*, 35(4):770–784, 2020.

[244] Alberto Revelli, Stefano Canosa, Andrea Carosso, Claudia Filippini, Carlotta Paschero, Gianluca Gennarelli, Luisa Delle Piane, and Chiara

Benedetto. Impact of the addition of early embryo viability assessment to morphological evaluation on the accuracy of embryo selection on day 3 or day 5: a retrospective analysis. *Journal of Ovarian Research*, 12:1–7, 2019.

[245] Katarina Kebbon Vaegter, Tatevik Ghukasyan Lakic, Matts Olovsson, Lars Berglund, Thomas Brodin, and Jan Holte. Which factors are most predictive for live birth after in vitro fertilization and intracytoplasmic sperm injection (ivf/icsi) treatments? analysis of 100 prospectively recorded variables in 8,400 ivf/icsi single-embryo transfers. *Fertility and sterility*, 107(3):641–648, 2017.

[246] C Scarica, D Cimadomo, L Dovere, A Giancani, M Stoppa, A Capalbo, FM Ubaldi, L Rienzi, and R Canipari. An integrated investigation of oocyte developmental competence: expression of key genes in human cumulus cells, morphokinetics of early divisions, blastulation, and euploidy. *Journal of assisted reproduction and genetics*, 36:875–887, 2019.

[247] Kirstine Kirkegaard, US Kesmodel, JJ Hindkjaer, and HJ Ingerslev. Time-lapse parameters as predictors of blastocyst development and pregnancy outcome in embryos from good prognosis patients: a prospective cohort study. *Human Reproduction*, 28(10):2643–2651, 2013.

[248] Mariabeatrice Dal Canto, Giovanni Coticchio, Mario Mignini Renzini, Elena De Ponti, Paola Vittoria Novara, Fausta Brambillasca, Ruggero Comi, and Rubens Fadini. Cleavage kinetics analysis of human embryos predicts development to blastocyst and implantation. *Reproductive biomedicine online*, 25(5):474–480, 2012.

[249] Giovanni Coticchio, Kenji Ezoe, Cristina Lagalla, Carlotta Zacà, Andrea Borini, and Keiichi Kato. The destinies of human embryos reaching blastocyst stage between day 4 and day 7 diverge as early as fertilization. *Human Reproduction*, page dead136, 2023.

[250] Danilo Cimadomo, Gemma Fabozzi, Alberto Vaiarelli, Nicolò Ubaldi, Filippo Maria Ubaldi, and Laura Rienzi. Impact of maternal age on oocyte and embryo competence. *Frontiers in endocrinology*, 9:327, 2018.

[251] Filippo Maria Ubaldi, Danilo Cimadomo, Alberto Vaiarelli, Gemma Fabozzi, Roberta Venturella, Roberta Maggiulli, Rossella Mazzilli, Su-

sanna Ferrero, Antonio Palagiano, and Laura Rienzi. Advanced maternal age in ivf: still a challenge? the present and the future of its treatment. *Frontiers in endocrinology*, 10:94, 2019.

[252] Kenji Ezoe, Tetsuya Miki, Hikari Akaike, Kiyoe Shimazaki, Tsubasa Takahashi, Yuko Tanimura, Ayumi Amagai, Ayano Sawado, Mai Mogi, Shigeru Kaneko, et al. Maternal age affects pronuclear and chromatin dynamics, morula compaction and cell polarity, and blastulation of human embryos. *Human Reproduction*, 38(3):387–399, 2023.

[253] Valentina Biasoni, Ambra Patriarca, Paola Dalmasso, Angela Bertagna, Chiara Manieri, Chiara Benedetto, and Alberto Revelli. Ovarian sensitivity index is strongly related to circulating amh and may be used to predict ovarian response to exogenous gonadotropins in ivf. *Reproductive Biology and Endocrinology*, 9(1):1–5, 2011.

[254] Andrea Roberto Carosso, Rik van Eekelen, Alberto Revelli, Stefano Canosa, Noemi Mercaldo, Chiara Benedetto, and Gianluca Gennarelli. Women in advanced reproductive age: are the follicular output rate, the follicle-oocyte index and the ovarian sensitivity index predictors of live birth in an ivf cycle? *Journal of Clinical Medicine*, 11(3):859, 2022.

[255] Neelke De Munck, Aşina Bayram, Ibrahim Elkhatib, Andrea Abdala, Ahmed El-Damen, Ana Arnanz, Laura Melado, Barbara Lawrenz, and Human Mousavi Fatemi. Marginal differences in preimplantation morphokinetics between conventional ivf and icsi in patients with preimplantation genetic testing for aneuploidy (pgt-a): A sibling oocyte study. *PloS one*, 17(4):e0267241, 2022.

[256] Linda Sundvall, Hans Jakob Ingerslev, Ulla Breth Knudsen, and Kirstine Kirkegaard. Inter-and intra-observer variability of time-lapse annotations. *Human Reproduction*, 28(12):3215–3221, 2013.

[257] Danilo Cimadomo, Laura Rienzi, Alessandro Conforti, Eric Forman, Stefano Canosa, Federica Innocenti, Maurizio Poli, Jenna Hynes, Laura Gemmell, Alberto Vaiarelli, et al. O-193 opening the black box: why do euploid blastocysts fail to implant? a systematic review and meta-analysis. *Human Reproduction*, 38(Supplement_1):dead093–234, 2023.

[258] Antonino Aparo, Vincenzo Bonnici, Giovanni Micale, Alfredo Ferro, Dennis Shasha, Alfredo Pulvirenti, and Rosalba Giugno. Fast Subgraph Matching Strategies Based on Pattern-Only Heuristics. *Interdisciplinary Sciences: Computational Life Sciences*, 11(1):21–32, 2019.

[259] Vincenzo Bonnici, Rosalba Giugno, Alfredo Pulvirenti, Dennis Shasha, and Alfredo Ferro. A subgraph isomorphism algorithm and its application to biochemical data. *BMC bioinformatics*, 14(S7):S13, 2013.

[260] Vincenzo Carletti, Pasquale Foggia, and Mario Vento. Performance comparison of five exact graph matching algorithms on biological databases. In *International Conference on Image Analysis and Processing*, pages 409–417. Springer, 2013.

[261] National cancer institute. National cancer institute. Accessed: 2021 september 21.

[262] Helen M Berman, Tammy Battistuz, Talapady N Bhat, Wolfgang F Bluhm, Philip E Bourne, Kyle Burkhardt, Zukang Feng, Gary L Gilliland, Lisa Iype, Shri Jain, et al. The protein data bank. *Acta Crystallographica Section D: Biological Crystallography*, 58(6):899–907, 2002.

[263] Rolf Huehne and Juergen Suehnel. The Jena Library of Biological Macromolecules-JenaLib. *Nature Precedings*, pages 1–1, 2009.

[264] Corinna Vehlow, Henning Stehr, Matthias Winkelmann, José M Duarte, Lars Petzold, Juliane Dinse, and Michael Lappe. CMView: interactive contact map visualization and analysis. *Bioinformatics*, 27(11):1573–1574, 2011.

[265] Damian Szklarczyk, Andrea Franceschini, Michael Kuhn, Milan Simonovic, Alexander Roth, Pablo Minguez, Tobias Doerks, Manuel Stark, Jean Muller, Peer Bork, et al. The STRING database in 2011: functional interaction networks of proteins, globally integrated and scored. *Nucleic acids research*, 39(suppl_1):D561–D568, 2010.

[266] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[267] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, 2005.