



STRUCTURAL PATTERN RECOGNITION FOR CHEMICAL-COMPOUND VIRTUAL SCREENING

Carlos Jesús García Hernández

ADVERTIMENT. L'accés als continguts d'aquesta tesi doctoral i la seva utilització ha de respectar els drets de la persona autora. Pot ser utilitzada per a consulta o estudi personal, així com en activitats o materials d'investigació i docència en els termes establerts a l'art. 32 del Text Refós de la Llei de Propietat Intel·lectual (RDL 1/1996). Per altres utilitzacions es requereix l'autorització prèvia i expressa de la persona autora. En qualsevol cas, en la utilització dels seus continguts caldrà indicar de forma clara el nom i cognoms de la persona autora i el títol de la tesi doctoral. No s'autoritza la seva reproducció o altres formes d'explotació efectuades amb finalitats de lucre ni la seva comunicació pública des d'un lloc aliè al servei TDX. Tampoc s'autoritza la presentació del seu contingut en una finestra o marc aliè a TDX (framing). Aquesta reserva de drets afecta tant als continguts de la tesi com als seus resums i índexs.

ADVERTENCIA. El acceso a los contenidos de esta tesis doctoral y su utilización debe respetar los derechos de la persona autora. Puede ser utilizada para consulta o estudio personal, así como en actividades o materiales de investigación y docencia en los términos establecidos en el art. 32 del Texto Refundido de la Ley de Propiedad Intelectual (RDL 1/1996). Para otros usos se requiere la autorización previa y expresa de la persona autora. En cualquier caso, en la utilización de sus contenidos se deberá indicar de forma clara el nombre y apellidos de la persona autora y el título de la tesis doctoral. No se autoriza su reproducción u otras formas de explotación efectuadas con fines lucrativos ni su comunicación pública desde un sitio ajeno al servicio TDR. Tampoco se autoriza la presentación de su contenido en una ventana o marco ajeno a TDR (framing). Esta reserva de derechos afecta tanto al contenido de la tesis como a sus resúmenes e índices.

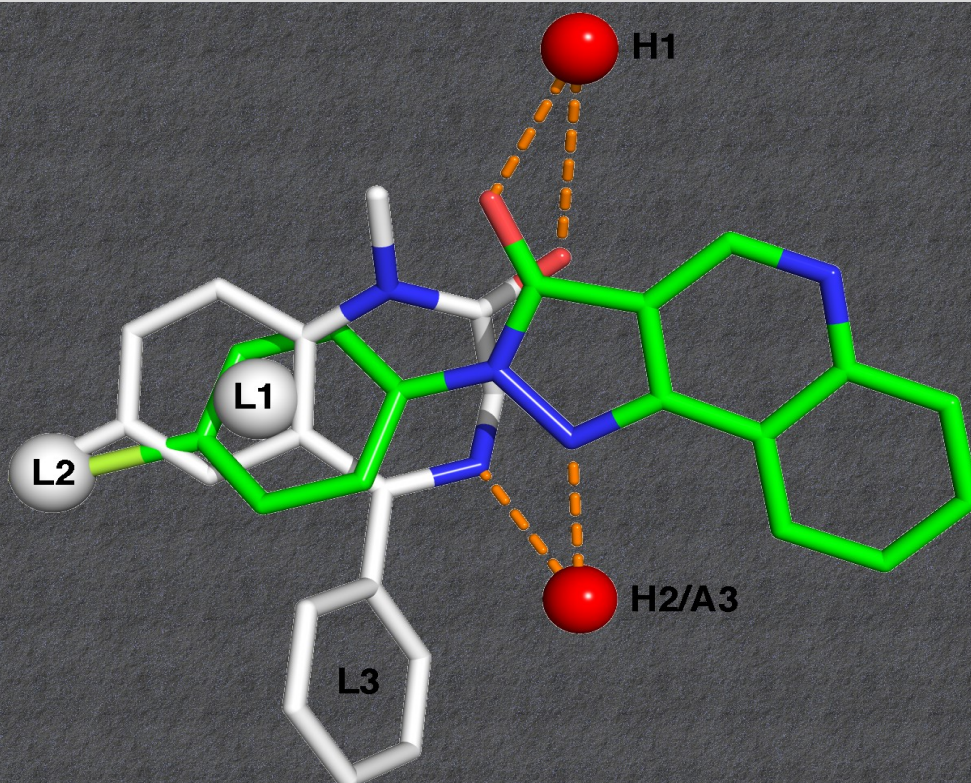
WARNING. Access to the contents of this doctoral thesis and its use must respect the rights of the author. It can be used for reference or private study, as well as research and learning activities or materials in the terms established by the 32nd article of the Spanish Consolidated Copyright Act (RDL 1/1996). Express and previous authorization of the author is required for any other uses. In any case, when using its content, full name of the author and title of the thesis must be clearly indicated. Reproduction or other forms of for profit use or public communication from outside TDX service is not allowed. Presentation of its content in a window or frame external to TDX (framing) is not authorized either. These rights affect both the content of the thesis and its abstracts and indexes.



UNIVERSITAT
ROVIRA i VIRGILI

Structural Pattern Recognition for Chemical-Compound Virtual Screening

Carlos Jesus Garcia Hernandez



DOCTORAL THESIS
2021

Structural Pattern Recognition for Chemical-Compound Virtual Screening

by

Carlos Jesus Garcia Hernandez

supervised by Dr. Alberto Fernández Sabater and Dr. Francesc Serratos Casanelles

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY
DEPARTMENT OF CHEMICAL ENGINEERING
UNIVERSITAT ROVIRA I VIRGILI

SEPTEMBER, 2021



DEPARTAMENT D'ENGINYERIA QUÍMICA

Avinguda dels Països Catalans 26
Campus Sescelades
43007 Tarragona, Catalunya (Spain)
Tel. 34 977 559 638
Fax 34 977 559 621

I STATE that the present study, entitled “Structural Pattern Recognition for Chemical-Compound Virtual Screening”, presented by Carlos Jesus Garcia Hernandez for the award of the degree of Doctor with international distinction, has been carried out under my supervision at the Department of Chemical Engineering of this university.

Tarragona, September 01, 2021

Doctoral Thesis Supervisors

Alberto Fernández Sabater

Francesc Serratosa Casanelles

The research presented in this thesis was carried out with funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 713679 and from the Universitat Rovira i Virgili (URV).



Acknowledgments

I would first like to thank my supervisors, Dr. Alberto Fernández and Dr. Francesc Seratosa, for allowing me to work with you during my Ph.D.; your knowledge and expertise were invaluable in formulating the scientific questions and methodology. Our weekly meetings and constant feedback helped sharpen my thinking and brought me a step closer to becoming a good scientist. Similarly, I want to thank my supervisors at ENSICAEN in France during my secondment, Dr. Luc Brun and Dr. Benoit Gaüzère. Your insightful advice and ideas gave me a new perspective that helped to complement and shape my Ph.D.

I want to thank my kind colleagues from “Lab-224”: Elena R., who patiently explained to me several math concepts being helpful throughout my whole research; Reyda, which long and pleasant conversations about research made me fall in love with science; Antonio, whose jokes and company during our several coffee-breaks made my days easier; Also, thanks to Elena B., Eszter, Laura, and Benjamin for your company and help during this journey. All those long days working in the office felt much better and easy-going with you guys there.

To my beautiful family: Mom, Dad, Sunny, and Valeria. The biggest thanks, your endless love during my long academic path from kindergarten until here, was crucial to keep me going and motivated. Finally, I’d like to thank my non-blood-related family, my friends. Starting with my old pals: Albert, Andrés, Antonio, Brenda, Juan, Franklin, Heredia, and Goyo, our friendship is stronger than distance. To my friends in Tarragona, those from “Vzlan flavor,” Verito and Lion, who accompanied me from the first day I arrived in Spain. From “Pu Team” and “The house of horror”: Antonio, Cristian, David, Ghassen, Ignasi, Inma, Noe, Ivan, Luquitas, Ricardo, and Yaride; All those table-football nights, movie nights, and other experiences together are some of my favorite memories from this period. A special thanks to Yaride and Luquitas for helping me fix and improve the images in this thesis. Thanks, Kate and Mari, for being so cool and visiting me when I needed it the most. Thanks to my friends from “The team”, particularly Adri, Eli, Junior, Samu, William, and Maeva, spending time with you during the last months gave me the energy boost I needed for the final weeks of writing. To conclude, thanks to my kind of imaginary friends Barney S. and Cosmo K. for all the jokes that gave rest to my mind and took it outside of my research; as one of them says: “Whatever you do in this life, it’s not legendary, unless your friends are there to see it”.

Abstract

Studying molecules and predicting their properties is an open problem in chemistry and drug design. Using computers to perform those analyses is known as cheminformatics. It aims to tackle the dimensionality problem and reduce the time and resources required to analyze millions of molecules. Drug discovery and design require satisfying important safety and efficacy objectives; therefore, it is inherently a multi-objective optimization process, making machine learning and graph theory a standard tool in cheminformatics research.

Molecules are naturally shaped as networks, making them ideal for studying by employing their graph representations, where nodes represent atoms and edges represent the chemical bonds. An alternative for this straightforward representation is the extended reduced graph, which summarizes the chemical structures using pharmacophore-type node descriptions to encode the relevant molecular properties. Once we have a suitable way to represent molecules as graphs, we need to choose the right tool to compare and analyze them. Graph edit distance is used to solve the error-tolerant graph matching; this methodology estimates a distance between two graphs by determining the minimum number of modifications required to transform one graph into the other. These modifications (known as edit operations) have an edit cost (also known as transformation cost) associated, which must be determined depending on the problem.

This study investigates the effectiveness of a graph-only driven molecular comparison employing extended reduced graphs and graph edit distance as a tool for ligand-based virtual screening applications. Those applications estimate the bioactivity of a chemical employing the bioactivity of similar compounds. An essential part of this study focuses on using machine learning and natural language processing techniques to optimize the transformation costs used in the molecular comparisons with the graph edit distance.

Overall, this work shows a framework that combines graph reduction and comparison with optimization tools and natural language processing to identify bioactivity similarities in a structurally diverse group of molecules. We confirm the efficiency of this framework with several cheminformatic tests applied to regression and classification problems over different publicly available datasets.

Contents

ACKNOWLEDGMENTS	10
ABSTRACT	11
1 GENERAL INTRODUCTION	22
1.1 Virtual screening	23
1.2 The molecule	24
1.3 Molecular similarity	31
1.4 Molecular descriptors	32
1.5 Similarity measure	51
1.6 Machine learning for molecular analysis	61
2 METHODOLOGY	68
2.1 Datasets	69
2.2 Molecular representations	69
2.3 Molecular comparison	72
2.4 Method evaluation	85
3 LIGAND-BASED VIRTUAL SCREENING USING GRAPH EDIT DISTANCE AS MOLECULAR SIMILARITY MEASURE	86
3.1 Chapter introduction	87
3.2 Specific materials and methods	87
3.3 Results	92
3.4 Discussion	100
4 LEARNING THE EDIT COSTS OF THE GRAPH EDIT DISTANCE APPLIED TO LIGAND-BASED VIRTUAL SCREENING APPLICATIONS	103
4.1 Chapter introduction	104
4.2 Specific materials and methods	104

4.3	Results	110
4.4	Discussion	117
5	NLP TOOLS TO SWIFTLY INFER GED TRANSFORMATION COSTS FOR VIRTUAL SCREENING	119
5.1	Chapter introduction	120
5.2	Specific materials and methods	131
5.3	Results	142
5.4	Discussion	146
6	GENERAL CONCLUSION	148
6.1	Prospects and Perspectives	152
	REFERENCES	169

Listing of figures

1.1	AFM image of 1,5,9-trioxo-13-azatriangulene (TOAT) molecule and its structural formula. (Source: Wikimedia.)	26
1.2	A covalent bond in a hydrogen molecule. (Source: Wikimedia.)	26
1.3	Non-covalent (hydrogen) bonds between water molecules. Hydrogen bonds appear between the weakly positive charge on the hydrogen atoms and the weakly negative charge on the oxygen atoms. Hydrogen bonds are represented with a dotted line rather than a continuous one since they are weak compared to covalent bonds. (Source: Wikimedia.)	27
1.4	Example of a mathematical graph. (Source: Wikimedia.)	28
1.5	Example of a diverse set of structures represented as graphs.	28
1.6	Three different models of the benzene molecule. From left to right: Space-filling model; Ball and stick model; Graph nodes and edges model. (Sources: Wikimedia and Wikimedia.)	29
1.7	Chiral amino acid and its mirror image. (Source: Wikimedia.)	30
1.8	Axial chirality of an example compound. Note that the two chiral variants are labeled as “R” and “S”. “R” from the Latin “rectus” meaning right-handed and “S” from the Latin “sinister” meaning left-handed.	30
1.9	Flowchart of a two-step molecular similarity searching process.	32
1.10	Example of a molecule with dipole moment motivated by nonuniform distributions of charges between the atoms.	34
1.11	Encoding of a molecule into a binary fingerprint.	36
1.12	Encoding of a molecule into its AP representation. The atom type is obtained employing the chemical type, the number of non-hydrogen neighbors, and the number of π electron pairs.	37
1.13	Encoding of a molecule into its TT representation. The atom type is obtained employing the chemical type, the number of non-hydrogen neighbors, and the number of π electron pairs.	38
1.14	Encoding of a molecule into its AP and BP representations.	39

1.15	Illustration of a hypothetical 10-bit substructure fingerprint. Each bit set is associated with the substructure marked with the circle.	39
1.16	Encoding of a molecule into its fragment-based Daylight fingerprint representation.	40
1.17	Illustration of 3 steps in the ECFP iterative process for one atom of a compound.	41
1.18	Hashed ECFP values for each atom in a compound during the initial iteration.	41
1.19	Generation of new hashed ECFP values by performing one iteration using an example compound. The initial atom identifiers are shown on the left. Each atom receives a new identifier after the update, shown on the right. . .	41
1.20	Solvent-accessible and van der Waals surface areas in a hypothetical molecule.	44
1.21	Example of a pharmacophore model of the benzodiazepine binding site on the GABAA receptor. White sticks represent carbon atoms of the benzodiazepine diazepam, while green represents carbon atoms of the nonbenzodiazepine CGS-9896. Red and blue sticks indicate oxygen and nitrogen atoms in both structures. Red spheres H1 and H2/A3 represent hydrogen bond donating and accepting sites in the receptor. L1, L2, and L3 denote lipophilic binding sites. (Source: Wikimedia.)	45
1.22	Example of a Markush structure and its reduced graph representation using a ring/non-ring reduction scheme.	47
1.23	Example of different types of reduced graphs obtained from the same molecule. (a) nodes represent ring systems (R) and acyclic components (Ac). (b) nodes represent carbon components (C) and hetero-atom components (H). (c) nodes represent aromatic rings (Ar), aliphatic rings (R), and functional groups (F). (d) nodes represent aromatic rings (Ar), functional groups (F), and linking groups (L).	48
1.24	Example of several chemical structures that reduce to the same reduced graph based on aromatic rings (Ar) and functional groups (F).	48
1.25	Example of the reduction of a chemical graph into its ErG representation as described by Stiefl <i>et al.</i> ¹⁶¹ (D) H-bond donor; (Ac) H-bond acceptor; (Hf) hydrophobic group; (Ar) aromatic ring system; (+/-) positive/negative charge.	50
1.26	Spatial inter-feature distances in Angstrom (Å) of an example molecule. . .	50
1.27	Example of molecule reduction using ErG. The original molecule is at the top, and its ErG representation is at the bottom. Ac: H-bond acceptor; Hf: hydrophobic group; Ar: aromatic ring system; +: positive charge. Colors are used to show how different parts of the original structure are reduced to nodes in the ErG.	52
1.28	Fingerprint-based method flowchart	55

1.29	SED-based method flowchart	56
1.30	String edit distance process to convert one path from graph A into one path from graph B. The total edit distance is 3 based on the substitution and insertion/deletion costs presented on the right side of the image.	57
1.31	Comparison of two molecules comprising two steps. First, we extract the ErGs. Second, we apply the GED.	58
1.32	Step-by-step GED transformations to go from g_1 to g_2	59
1.33	Molecular comparison flowcharts. Difference between traditional ErG methods and our proposal	59
1.34	An edit path that transforms graph A into graph B	60
1.35	Graph of a function. The global maximum is at $(x, y, z) = (0, 0, 4)$. (Source: Wikimedia.)	63
1.36	Example of local and global minima in a hypothetical optimization problem.	64
1.37	Representation of a genetic algorithm used to optimize the transformation costs (insertion, deletion, and substitution) used for the GED.	65
2.1	Illustration with different versions of labeled graphs.	71
2.2	Social network graph illustrating friendships among a group of users. (Source: Wikimedia.)	72
2.3	Example of an ErG with the corresponding pharmacophoric features. Ac: H-bond acceptor; Hf: hydrophobic group; Ar: aromatic ring system; +: positive charge.	73
2.4	Example of a maximum common subgraph for a group of molecules.	75
2.5	Simplified GED process to convert g_1 into g_2 . The total edit distance is 11 based on the substitution and insertion/deletion costs presented on the right side of the image.	76
2.6	Matrix of costs and workers. When applied to the above table, a pairwise assignment solver like the Kuhn–Munkres algorithm would yield the minimum cost of \$6, assigning Paul to clean the bathroom, Dave to sweep the floors, and Chris to wash the windows.	77
2.7	BP algorithm’s cost matrix.	79
2.8	Illustration of the GED used as a distance metric.	81
2.9	GED execution time with an increasing number of ErGs.	84
3.1	Examples of activation functions normalizing a method’s output to a probability distribution over predicted output classes. Logistic regression is the type of regression analysis often used when the dependent variable is binary.	90
3.2	Depiction of a ROC curve while changing the distribution of TP and FP values. TP and FP distribution changes by modifying the threshold.	90

3.3	Example of an array holding the predicted activities from one experiment. ROC curve and its corresponding AUC value are obtained employing the predictions.	91
3.4	AUC and BEDROC ($\alpha = 20$) for all available targets in the LBVS benchmarking platform. The scattered values on the left of both subplots represent the median value from 10 predefined random-built splits, using different colors and shapes per similarity method. Vertical segmented lines mark the edge between different datasets (from left to right: ULS-UDS, GLL&GDD, CAPST, DUD-E, NRLiSt_BDB, MUV). The box-and-whisker plots on the right of both subplots show the distribution of the resulting values for each similarity method. The boxes show the first and third quartile, the line is the median value (second quartile), and the whiskers extend from the boxes to show the range of the data (outliers are included if there are any).	94
3.5	AUC results for all available targets in the LBVS benchmarking platform separated by dataset. Each scattered value on the left of each subplot represents the median value of 10 predefined random-built splits. Different colors and shapes are used for each similarity method. Box-and-whisker plots on the right of each subplot show the distribution of results.	96
3.6	BEDROC ($\alpha = 20$) results for all available targets in the LBVS benchmarking platform separated by dataset. Each scattered value on the left of each subplot represents the median value of 10 predefined random-built splits. Different colors and shapes are used for each similarity method. Box-and-whisker plots on the right of each subplot show the distribution of results.	97
4.1	Objective function.	109
4.2	Training evolution curves for the FXA target in the DUD-E dataset.	111
4.3	The number of miss-classifications in Experiment 4 using the test set over the 127 targets available in the six datasets combined. The scattered values on the left of the plot represent the number of classification errors. Different colors and shapes represent different sets of edit costs. Vertical segmented lines mark the limit between different datasets (from left to right: ULS-UDS, GLL&GDD, CAPST, DUD-E, NRLiSt_BDB, MUV). The box-and-whisker plots on the right show the distribution of the resulting values. The boxes show the first and third quartile, the line in the middle of the box is the median value (second quartile), and the whiskers extend from the boxes to show the range of the data (outliers are not included).	112

4.4	The number of miss-classifications for all available targets in the LBVS benchmarking platform separated per dataset and experiment. The scattered values on the left of each subplot represent the number of classification errors (the lower the values, the better) using different colors and shapes depending on the edit costs used. Box-and-whisker plots on the right of each subplot show the distribution of the resulting values for each experiment.	115
4.5	AUROC values comparison between chapter 3 and chapter 4 over the 127 targets.	117
5.1	Example of conversion from an adjacency matrix to a low-dimensional continuous vector space.	121
5.2	Illustration of the 2-dimensional embeddings obtained from the graph on the left. The embeddings can be plotted and clustered based on metrics such as the Euclidean distance.	122
5.3	The green-box words are given to the network as targets; they are optimized to predict the words in the neighborhood (white-box words). In this example, we consider words located up to two places away from the input word. .	123
5.4	Components of a skip-gram neural network.	124
5.5	One-hot vectors construction for an example dictionary of words.	125
5.6	Wevi screenshot ¹⁷⁸ . The top left shows the control panel with the input dictionary and the learning settings. The top right shows the evolution of the neurons in the neural network during the training. The bottom left shows the weight matrices for the input and output vectors from each word in the dictionary. The bottom right shows the PCA distribution of the embedding vectors obtained from the words in the dictionary. (Source: http://bit.ly/wevi-online .)	126
5.7	Phases of DeepWalk.	127
5.8	Probabilities of a random walk step in Node2vec after moving one step from the red to the green node. The probability of going back to the red node is $1/P$; the probability of going to a non-neighbor of the previous node (red) is $1/Q$; the probability of going to a neighbor of the previous node (red) is 1 . .	128
5.9	Morgan algorithm's substructure selection on heavy atoms for two different radios.	129
5.10	Phases of the approach used in this chapter.	130

5.11	Wevi screenshot ¹⁷⁸ using an example of a small dictionary and after only 200 iterations. Left shows the weight matrices for the input and output vectors. Right shows the PCA distribution of the embedding vectors. In the bottom left of the PCA plot, we can notice a cluster with the words “orange”, “rice”, and “apple” in orange and the word “eat” in blue; implying a semantic relationship between the fruits and the verb “eat” higher than with “milk” or “water” whose vectors are closer to the verb “drink”. (Source: http://bit.ly/wevi-online .)	135
5.12	The process of joining together paths from different graphs.	137
5.13	2D and 3D T-SNE visualizations of embeddings from the AIDS dataset.	138
5.14	Stability evolution of embeddings with different datasets.	139
5.15	Heatmap using GED values with MAO dataset.	143
5.16	Results for PTC_0, PTC_1, PTC_2, and PTC_3 datasets.	144
5.17	(a) Results for MAO and AIDS datasets. (b) Results for the ACYCLIC dataset.	145

List of Tables

1.1	List of SMARTS patterns used to identify the pharmacophoric features in ECFPs. Adapted from previous work ⁷⁰ . (Source: RDKit.)	42
2.1	Input data used for the experiments. The column entitled ‘Dataset’ contains the name of each dataset, and the column entitled ‘Targets used’ contains the name of the targets used during the experiments for each dataset. Note that in the result plots shown below, per-target points are arranged in the same order as they are in this table.	70
2.2	Description of the node and edge attributes that make up an ErG.	82
2.3	Substitution and Insertion/Deletion costs used in the GED and SED calculation.	83
3.1	Input data used for the experiments. The column entitled ‘Dataset’ contains the name of each dataset, and the column entitled ‘Targets used’ contains the name of the targets used during the experiments for each dataset. Note that per-target points in the result plots shown below are arranged in the same order as in this table.	93
3.2	P-values for the Friedman test (AUC and BEDROC) comparing the three similarity methods (GED-based, FP-based, and SED-based) simultaneously. The test is done per dataset, and all datasets are combined in the last row. Here, a confidence level of $\alpha = 0.05$ is used, so p-values lower than α indicate statistically significant differences, which are marked with an asterisk (*) in the table.	98
3.3	P-values for the pairwise Wilcoxon signed-rank test (AUC and BEDROC) comparing the three similarity methods (GED-based, FP-based, and SED-based). The test is applied to all targets in the datasets combined. Here, a confidence level of $\alpha = 0.05$ is used, so p-values lower than α indicate statistically significant differences, which are marked with an asterisk (*) in the table.	99

3.4	P-values for the pairwise Wilcoxon signed-rank test (AUC) comparing the three similarity methods (GED-based, FP-based, and SED-based). The test is done using all targets separated by datasets. Here, a confidence level of $\alpha = 0.05$ is used, so p-values lower than α indicate statistically significant differences, which are marked with an asterisk (*) in the table.	99
3.5	P-values for the pairwise Wilcoxon signed-rank test (BEDROC) comparing the three similarity methods (GED-based, FP-based, and SED-based). The test is done using all targets separated by datasets. Here, a confidence level of $\alpha = 0.05$ is used, so p-values lower than α indicate statistically significant differences, which are marked with an asterisk (*) in the table.	100
3.6	Three sample molecules from the target VDR_Agonist in the NRLiSt_BDB dataset.	101
3.7	Distances between molecules shown in Table 3.6 computed using the FP-based, SED-based, and GED-based similarity methods.	101
4.1	Substitution, Insertion and Deletion costs for Nodes based on those proposed by Harper <i>et al.</i> ⁸⁰	106
4.2	Substitution, Insertion and Deletion costs for Edges based on those proposed by Harper <i>et al.</i> ⁸⁰	106
4.3	Harper's and learned costs (average values from all targets) per experiment.	116
5.1	Feature summary of the datasets.	131
5.2	Direct Substitution, Insertion, and Deletion costs for Nodes.	134
5.3	Direct Substitution, Insertion, and Deletion costs for Edges.	134
5.4	Actual values for all the experiments using the seven datasets and the three sets of transformation costs.	145

*If you wish to make an apple pie from scratch, you must first
invent the universe.*

Carl Sagan.

1

General introduction

1.1 VIRTUAL SCREENING

Advances in experimental techniques for molecular analysis like High-throughput screening and combinatorial chemistry have led to developing a broad catalog of chemical structures and their corresponding bioactivities^{100,19}. These structure-activity relationships form the basis of a field known as cheminformatics, where scientists mine molecular datasets to create predictive models to be used in the drug discovery pipeline^{66,50}.

The average time for pharmaceuticals to bring a new drug to market is about 13 years¹²⁸. An essential first step in drug discovery is creating a pool of candidates for synthesis and characterization; this is challenging since the space of possible molecules is vast and increasingly growing^{89,134}. The number of molecules synthesized is in the order of 10^8 , while potential drug-like molecules are estimated between 10^{23} and 10^{60} ¹³¹. Having those numbers in mind, it is evident the need for knowledge-guided virtual filtering and screening of compounds, which help prioritize synthesis and work as a complementary tool to high-throughput screening^{154,159}.

Quantitative structure-activity relationship (QSAR) models are computational or mathematical models that attempt to find significant correlations between molecular structure and molecular activity. With the massive increase in data on chemical compounds and their reactivities thanks to high-throughput screening techniques, there is also a rising need for computational tools to reduce the drug synthesis and test cycle execution times. These tools are essential if activity data are analyzed and new models are created for virtual screening techniques⁹².

Virtual screening – usually referred to as computational techniques to search and filter

chemical databases^{147,5} – is a common step in the drug discovery process. Two main categories of methods can be found in the virtual screening inventory: structure-based virtual screening (SBVS)⁸² and ligand-based virtual screening (LBVS)³⁸. SBVS uses the 3D structure information of a target (obtained from X-ray, NMR, or some other method) to dock a group of molecules into the binding site of a protein and estimate the likelihood that the molecules will bind to the protein^{91,37}. LBVS uses information about the known activity of some molecules, activity in terms of their behavior as ligands binding to a receptor, to predict the unknown activity of new molecules¹⁶³. In this work, the focus will be only on LBVS applications. The main LBVS approaches are pharmacophore mapping¹⁶³, shape-based similarity⁸⁸, fingerprint similarity, and various machine learning methods¹⁰⁸. The concept of molecular similarity is frequently used in LBVS contexts where the chosen measure of similarity might determine the success or not of a virtual screening method.

1.2 THE MOLECULE

Understanding matter and its composition might be one of the oldest, most important, and most complex challenges humanity has ever faced. This challenge can be traced back to the pre-scientific Greek era, around the 5th century BC. At that time, philosophers started to imagine the universe's composition as made of atoms and voids. They described the fundamental elements like fire, earth, air, and water and attracting and repulsing “forces” that make the elements interact with each other.

Modern materials science and chemistry are primarily based on the need to devise and create new molecules with specific desired properties; therefore, unraveling the working principles of the molecular world is vital to understand the physicochemical properties and the

reactivity of the molecules themselves. The atomic composition of molecules and the molecular shape (i.e., the 3d shape of a molecule) significantly influence the molecule's reactivity, influencing features like boiling point, melting point, and ligand-binding properties for specific proteins.

The following section will discuss basic concepts and topics about the molecule, its structure, and its working mechanism.

1.2.1 WHAT IS A MOLECULE?

Molecules can be considered the smallest fundamental unit of a chemical compound that can participate in a chemical reaction; they can interact with one another by modifying their structure and properties. Molecules are made up of a group of atoms joined together by physical forces called chemical bonds. Those atoms are always in rapid motion, making molecules not static but dynamic entities, where atoms constantly change position, stretching back and forth the chemical bonds.

There is a substantial range of different molecules, starting from just a few atoms up to many thousands. Figure 1.1 shows the Atomic force microscopy (AFM) image of a molecule and its structural formula.

1.2.2 MOLECULAR BONDS

Bonds can be seen as the "glue" holding together the atoms in a molecule; this "glue" is defined and managed by the shared electrons in different atoms. Molecular bonds can be broadly classified into two groups: covalent bonds and non-covalent bonds.

Covalent bonds are, as a definition, the most critical type of bond since the molecule is

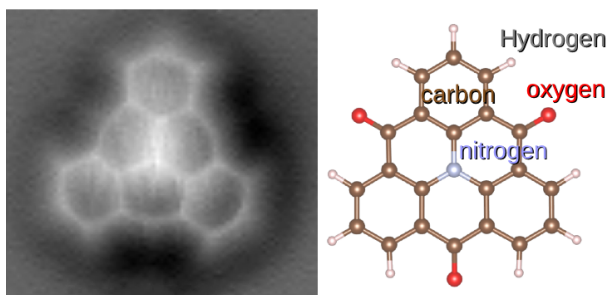


Figure 1.1: AFM image of 1,5,9-trioxo-13-azatriangulene (TOAT) molecule and its structural formula. (Source: [Wikimedia](#).)

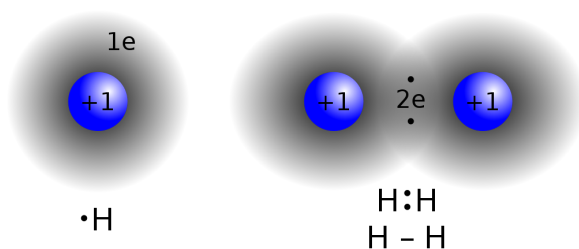


Figure 1.2: A covalent bond in a hydrogen molecule. (Source: [Wikimedia](#).)

technically defined as a group of atoms joined together by covalent bonds. They are the most robust and most stable type of chemical bond, i.e., once they are created through a chemical reaction, it requires a significant amount of energy to break them apart. This type of bond involves sharing electron pairs between atoms, which means that in a pair of atoms with a covalent bond, some electrons spend time moving around both atoms. Figure 1.2 depicts the covalent bond involved in the making of the water molecule.

Non-covalent bonds are not as strong as covalent bonds, making them more transitional and changeable in time, constantly breaking and joining together with different atoms. This kind of bond involves electromagnetic forces to tie atoms together, which are weak compared to the electron sharing present in the covalent bonds.

Non-covalent bonds can be of different types; they include hydrogen bonds (see Figure

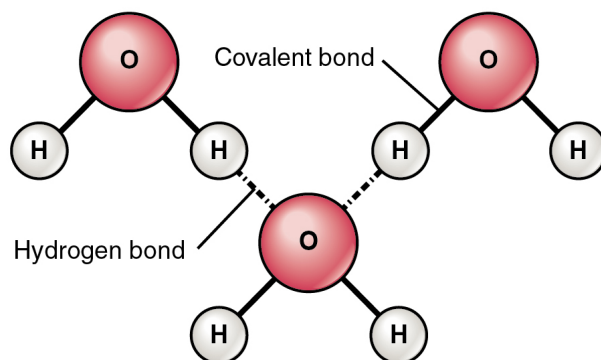


Figure 1.3: Non-covalent (hydrogen) bonds between water molecules. Hydrogen bonds appear between the weakly positive charge on the hydrogen atoms and the weakly negative charge on the oxygen atoms. Hydrogen bonds are represented with a dotted line rather than a continuous one since they are weak compared to covalent bonds. (Source: [Wikimedia](#).)

1.3), salt bridges, pi-stacking, among others. These bonds have an essential effect on determining the shape of molecules and the way how they interact with one another. Non-covalent and shape-dependent interactions are of paramount importance in drug design, given that most drug molecules interact with our biological molecules (in our body) through non-covalent interactions.

1.2.3 MOLECULAR GRAPHS

Later in this document, we will make a more technical definition of a graph; for the time being, it is enough to say that a graph is a mathematical data structure used to represent pairwise relationships between objects. These graphs are made up of nodes (also known as vertices) connected by edges (also known as links); check Figure 1.4 for an example. Graphs are used to represent a broad diversity of structures like computer networks, letters, pixels in an image, the shape of ridges in a fingerprint, and, more importantly for us, chemical structures (see Figure 1.5).

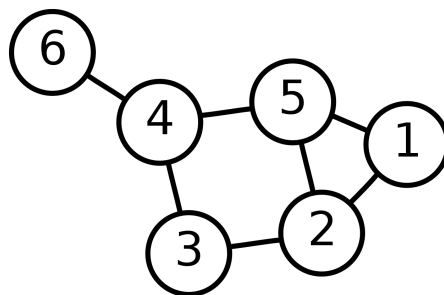


Figure 1.4: Example of a mathematical graph. (Source: [Wikimedia](#).)

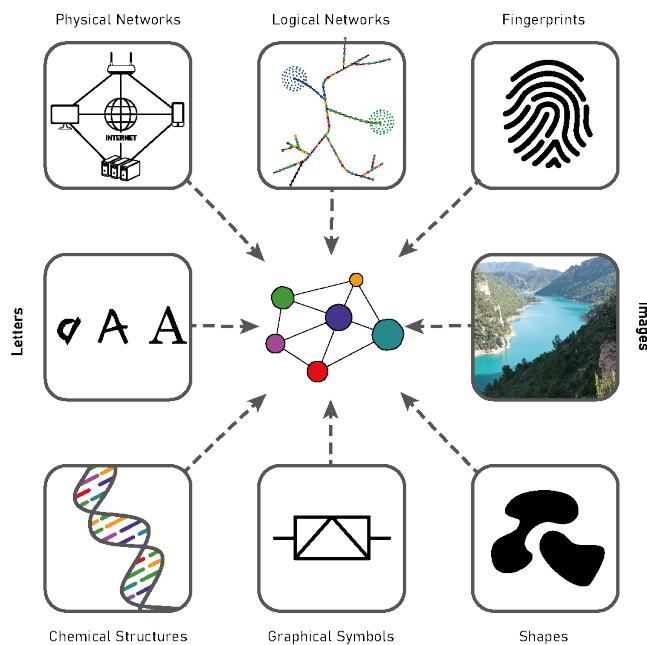


Figure 1.5: Example of a diverse set of structures represented as graphs.

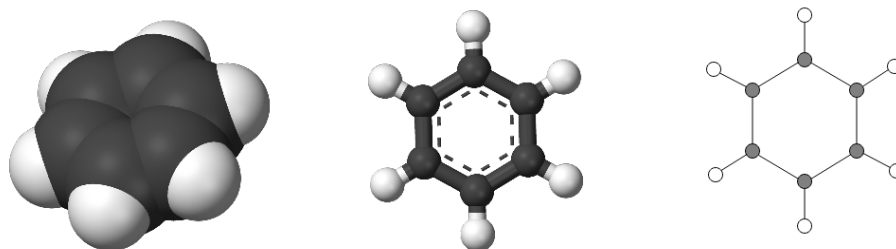


Figure 1.6: Three different models of the benzene molecule. From left to right: Space-filling model; Ball and stick model; Graph nodes and edges model. (Sources: [Wikimedia](#) and [Wikimedia](#).)

Molecules can be represented as graphs; the most straightforward way to do it is by using nodes to represent the atoms in the molecule and using edges to represent the chemical bonds between those atoms, as illustrated in Figure 1.6.

1.2.4 CHIRALITY OF MOLECULES AND STEREOISOMERS

An object, shape, or structure is called “chiral” if it cannot be superimposed onto its mirror image¹⁰⁶; in other words, the object and its mirror image are distinguishable. Hands are universally recognized chiral objects; the left hand is a mirror image of the right hand, which cannot be superimposed, no matter how the two hands are oriented¹⁷⁴. A sphere, for instance, is “achiral” and cannot be distinguished from its mirror image, which also means that both images can be superimposed.

A chiral molecule has a non-superposable mirror image, and the most common cause of molecular chirality is an asymmetric carbon atom²⁹ (see Figure 1.7). The two mirror images of a chiral molecule are usually called stereoisomers, and the information holding their differences is called stereochemical information.

The 3D spatial distribution of the atoms relative to each other in a molecule is called “molecular conformation”. A chiral molecule usually comes in two forms, one being the

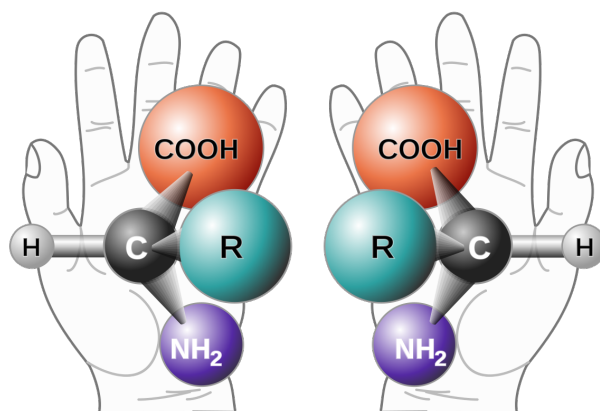


Figure 1.7: Chiral amino acid and its mirror image. (Source: [Wikimedia](#).)

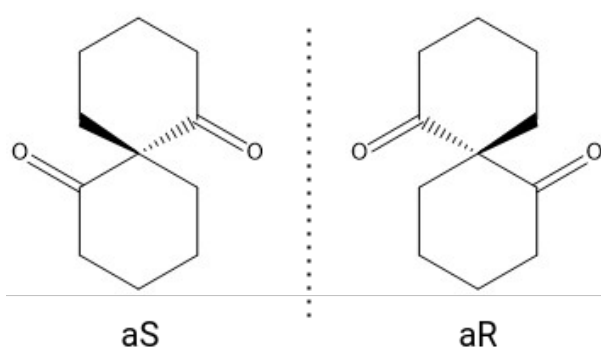


Figure 1.8: Axial chirality of an example compound. Note that the two chiral variants are labeled as “R” and “S”. “R” from the Latin “rectus” meaning right-handed and “S” from the Latin “sinister” meaning left-handed.

mirror image of the other; those two forms are known as “right-handed” and “left-handed,” as shown in Figure 1.8. In many experiments, distinguishing between the two versions of a molecule is difficult, especially in computational models. For instance, the molecular graphs, two graphs representations obtained from the left and right-handed forms of a chiral molecule, are the same. Considering the chiral information of molecules gets more complicated as molecules increase in size because the number of conformations they can take grows exponentially with the number of atoms; therefore, the computational power needed to simulate and analyze large molecules can be restrictive.

To reduce the computational expense, the molecular representation and molecular comparison techniques used in this work do not envisage the use of stereochemical information for molecules. Nevertheless, it should be possible to include this information since the 3D location of each atom is available for all datasets used; thus, a reference for the position of the neighbors for each atom should be possible to be established beforehand during the molecular representation process.

1.3 MOLECULAR SIMILARITY

It is assumed that structurally similar molecules are likely to have similar activity properties⁸⁴; therefore, molecular similarity methods are commonly used to select suitable candidates in the drug discovery industry. These similarity methods are used in applications related to molecular clustering, similarity searching, or molecular screening^{12,122,181,95,180}.

Regardless of the application, molecular similarity searching usually requires one descriptor representing the molecules and a similarity or comparison measure to define the level of similarity (or dissimilarity) between those molecules.

The purpose of featuring the molecule and creating molecular descriptors is to have a more convenient numerical representation of the molecules, which can be readily applied to mathematical models or learning algorithms, as illustrated in Figure 1.9.

Molecules are complex entities; as a result, researchers have created a varied group of different techniques for featuring them. In the following two sections, we will dive into the properties of different molecular descriptors and the similarity methods used to compare them.

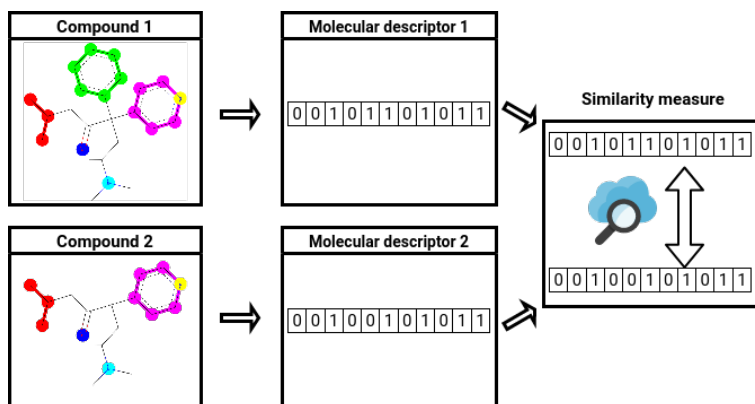


Figure 1.9: Flowchart of a two-step molecular similarity searching process.

1.4 MOLECULAR DESCRIPTORS

Molecular descriptors correspond to different quantities or numerical values describing physicochemical properties, structural information, behavioral traits, and some other possible features describing the molecules. There is a more formal definition by Todeschini and Consonni:

“The molecular descriptor is the final result of a logic and mathematical procedure which transforms chemical information encoded within a symbolic representation of a molecule into a useful number or the result of some standardized experiment.”¹⁶⁷

These descriptors are obtained from experimental measurements or theoretical symbolic representations; they can come from different fields like quantum chemistry, graph theory, organic chemistry, and information theory. They can be applied in different fields like toxicology, physical, pharmaceutical, or environmental chemistry^{153,182}. Molecular descriptors can be classified according to their “dimensionality”, referring to the molecular representa-

tions from which they are calculated. According to this classification, descriptors can be one of the following types: one-dimensional (1D) descriptors, including bulk or whole molecular properties and physiochemical parameters like logP or molecular weight; 2D descriptors such as structural fragments or connectivity indices. And 3D descriptors such as surface area, molecular volumes, or spatial pharmacophores^{5,185,10}.

1.4.1 ONE-DIMENSIONAL DESCRIPTORS

One-dimensional or whole molecular descriptors are general molecular properties derived from experimental measurements using classical physics or chemistry; they include different values such as molecular size, weight, logP, dipole moment, and polarizability^{109,129,148,98}.

Molecular size measures the volume a molecule occupies in the three-dimensional space; the smallest one is the diatomic hydrogen (H_2)⁴⁶. Similarly, **Molecular weight** refers to the mass of a given molecule. When working with macroscopic quantities of a substance, it is common to use the molar mass instead, which is computed as the mass of the substance divided by the amount of the substance, which in turn, is computed as the discrete number of atomic particles divided by the Avogadro constant⁷.

LogP is derived from the **partition coefficient**, which is the ratio of the concentration of a solution prepared with two solvents in a two-compartment system under equilibrium conditions¹⁶⁷. The concentration ratio is usually transformed into a logarithmic form, therefore the LogP abbreviation. When the solution is prepared with water as a solvent, the LogP value can be used to measure the lipophilicity or hydrophobicity of the compounds^{41,102}, which is essential information for the pharmaceuticals since it dictates the drug dissolution properties inside the body¹⁷⁷.

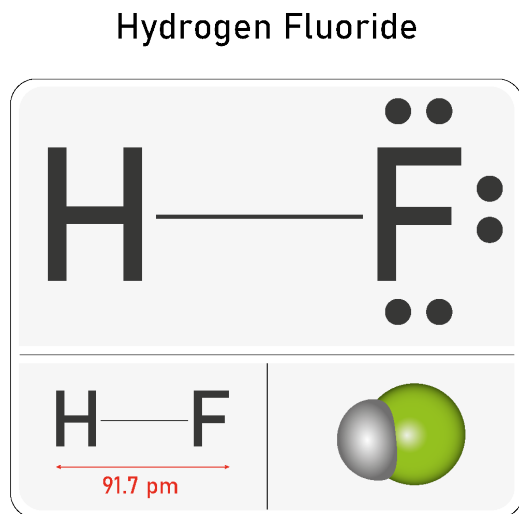


Figure 1.10: Example of a molecule with dipole moment motivated by nonuniform distributions of charges between the atoms.

Dipole moment (also known as electric dipole moment) is an electric polarization descriptor encoding information related to the charge distribution inside the molecule²³, which can be understood as the displacement of positive and negative charges in a molecule concerning the center of gravity. Dipole moments are motivated by nonuniform distributions of charges between the atoms due to an unequal electron density distribution (see Figure 1.10). A dipole moment equal to zero is called nonpolar and indicates a molecule with a center of symmetry; any other molecule having a dipole moment different from zero is called polar¹⁶⁷.

Similarly, **polarizability** (also known as static polarizability) is an electric polarization descriptor encoding the tendency of a molecule, when embedded into a uniform electric field, to acquire a proportional electric dipole moment¹⁶⁷. Regardless of polar or nonpolar, any molecule is polarizable, which means any molecule shows an induced dipole moment dif-

ferent from zero and proportional to the electric field strength¹⁶⁸. The molecular polarizability can be approximated as the sum of the atomic polarizabilities over all the molecule atoms¹⁶⁸; those atomic contributions to polarizability were before estimated by several authors^{85,114,124,167}.

These whole-molecular descriptors are more valuable in some cases than others; they will likely work better when trying to predict molecular behaviors relying on the generic properties of the molecules. Still, they will probably not be as much used to predict molecular behaviors relying on the specific arrangement of atoms. The descriptors in the following sections will be more helpful in those latter cases.

1.4.2 TWO-DIMENSIONAL DESCRIPTORS

This descriptor creates array representations of the molecules by simplifying the atomic information within them; a typical example of such descriptors is molecular fingerprints^{11,45,101,105}.

FINGERPRINTS

Molecular fingerprints are substructure descriptors generating a bit vector (or count vector) encoding 2D and/or 3D information about the molecule's structure to represent molecules as mathematical objects, particularly suitable for informatic treatment in machine learning or statistical analysis¹⁶⁷. Fingerprints often are made up of binary digits (1s and 0s) representing the presence or absence (in some cases, the frequency) of certain substructures or features in the molecule, as illustrated in Figure 1.11.

Most fingerprint implementations take molecules of different sizes and encode them into fixed-length vectors (typically between 1K and 4K bits). This fixed-length is a helpful ad-

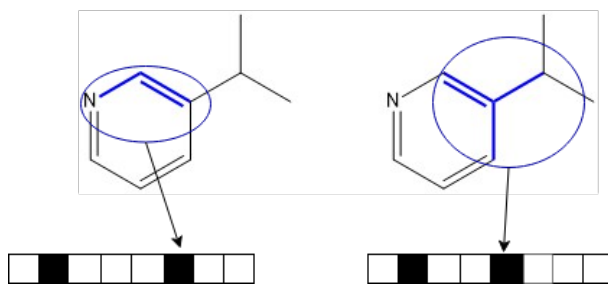


Figure 1.11: Encoding of a molecule into a binary fingerprint.

vantage because many computing models, such as those in machine learning, are built to work with inputs having the same size. Whereas traditional fingerprint vectors represent the presence or counting of structural fragments in each bin of the vector, some other implementations hash the counted information into vectors of a predefined length (number of bins). Therefore in some cases, the same bin may encode information from different fragments¹⁶⁷.

Some examples of well-known fingerprint implementations are atom pairs (AP)³⁶, topological torsion (TT)¹²³, binding property pairs (BP)⁸⁶, MACCS^{101,51}, Fragment-based Daylight⁴⁵, and Extended-connectivity fingerprints (ECFPs)¹⁴⁰.

Atom pairs (AP) are built using substructures of the form:



where (distance) represents the distance between “atom *i*” and “atom *j*” along the shortest path, it is expressed in terms of the number of bonds. The distance representation is performed for all pairs of non-hydrogen atoms in the structure of the chemical compound, and the description for each atom embeds the chemical type, the number of non-hydrogen atoms attached to it, and the number of π electrons that it bears³⁶ (see Figure 1.12).

For a molecule with n atoms, we will obtain $n(n - 1)/2$ atom pairs. As it is expected, generally, not all of the atom pairs are unique, even though only the presence of an atom pair

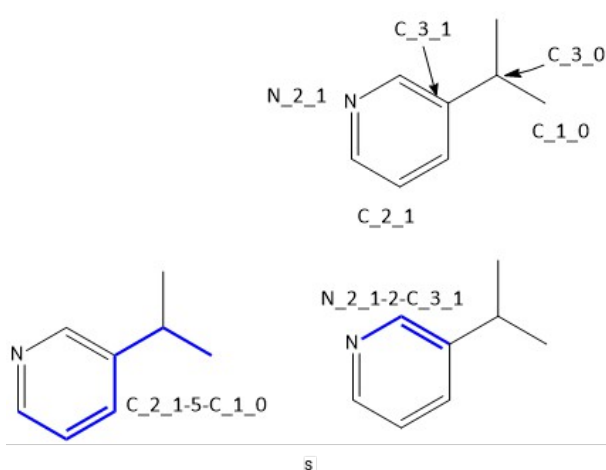


Figure 1.12: Encoding of a molecule into its AP representation. The atom type is obtained employing the chemical type, the number of non-hydrogen neighbors, and the number of π electron pairs.

is considered later for the comparison process and similarity computation.

A **Topological torsion (TT)** works similarly to AP, but it encodes the information from a linear sequence of four consecutively bonded non-hydrogen atoms¹²³. Each TT is of the form:

atom type i - atom type j - atom type k - atom type m

where i, j, k, and m are consecutive atoms. The description for each atom in the sequence includes its atomic type, the number of non-hydrogen branches attached to it, and its number of π electron pairs¹²³ (see Figure 1.13).

As reported by the authors, the combination of AP and TT descriptors may magnify the information obtained from the molecular topology; hence, using them together might enhance the performance significantly in the molecular virtual screening process¹²³.

Binding property pairs (BP) are an extension of the AP; they are built similarly, except that the atom type is described differently. In BP, the non-hydrogen atoms are assigned to

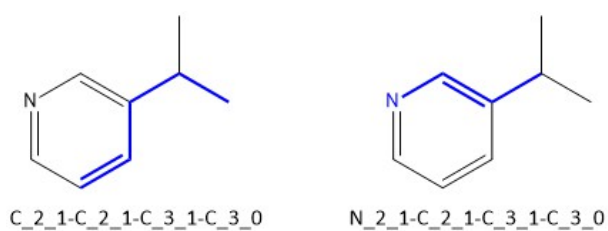


Figure 1.13: Encoding of a molecule into its TT representation. The atom type is obtained employing the chemical type, the number of non-hydrogen neighbors, and the number of π electron pairs.

one of the following seven binding property types⁸⁶:

1. cations
2. anions
3. neutral hydrogen bond donors
4. neutral hydrogen bond acceptors
5. polar atoms (both donor and acceptor, e.g., hydroxy oxygen)
6. hydrophobic atoms
7. other

The difference between AP and BP is illustrated in Figure 1.14.

MACCS is a keys-based fingerprint, which means that the algorithm tries to describe the compounds depending on certain substructures or features in the molecule (Figure 1.15). The presence of the substructures is matched with a given list of structural keys. The most commonly used version of MACCS uses a list of 166 structural keys based on SMARTS patterns, as exhibited in Table 1.1.

Fragment-based Daylight is the most commonly used among the topological or path-based fingerprints; similarly to TT, it analyses molecular fragments in the molecule built by following linear paths of consecutively connected atoms up to a certain number of bonds

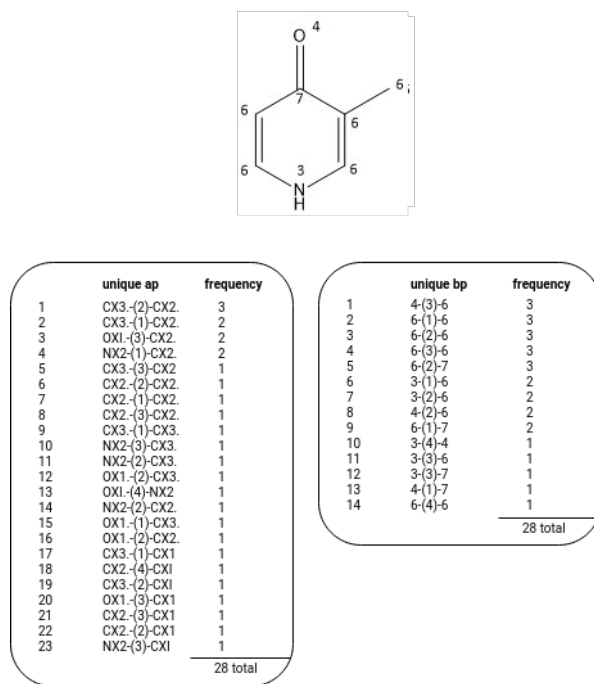


Figure 1.14: Encoding of a molecule into its AP and BP representations.

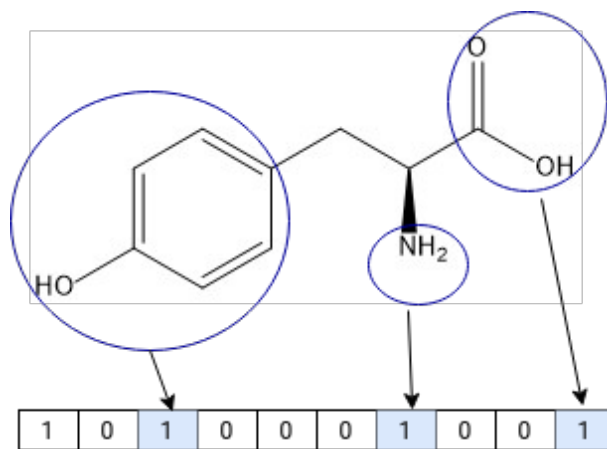


Figure 1.15: Illustration of a hypothetical 10-bit substructure fingerprint. Each bit set is associated with the substructure marked with the circle.

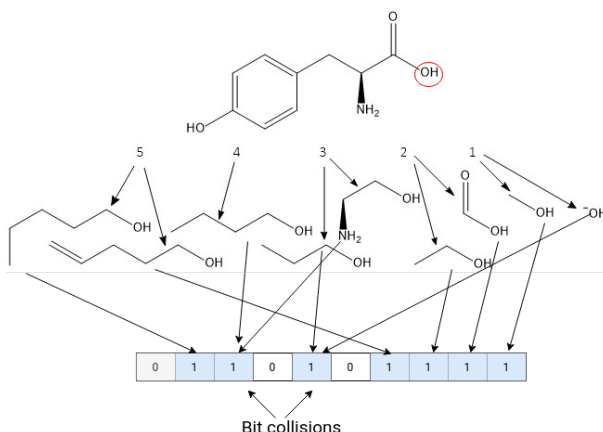


Figure 1.16: Encoding of a molecule into its fragment-based Daylight fingerprint representation.

(see Figure 1.16). The Daylight fingerprint checks all possible connectivity pathways starting from each atom in the molecule and up to a given length; later, the algorithm encodes this information in a hashed bit array up to 2048 bits. Hashed fingerprints often can identify different features with a single bit in the array; this is called “bit collision”. The hashed fingerprint array is adjusted in length according to the specific needs of the study, where shorter arrays are faster to compute and compare, but they also are more prone to bit collisions.

Extended-connectivity fingerprints (ECFPs), also known as Morgan, are the most commonly used circular fingerprints. They also are hashed topological fingerprints using a similar approach to that from Daylight fingerprint, but building the fragments within a circular radius for each atom in the molecule instead of doing it with linear pathways. ECFPs are based on the Morgan algorithm¹¹⁷. They are intended to represent the atom’s circular neighborhoods in different sizes of diameter. Still, the most commonly used value for the diameter is 4 (also known as ECFP₄), which allows sub-fragments to have a radius of two bonds around a central atom (see Figure 1.17).

The atom descriptors in ECFPs are created for each atom in an iterative process, first in-

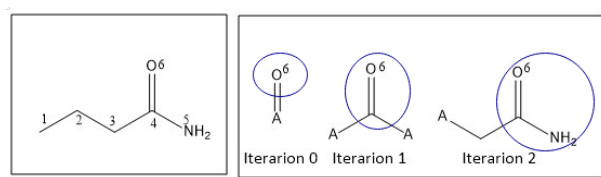


Figure 1.17: Illustration of 3 steps in the ECFP iterative process for one atom of a compound.

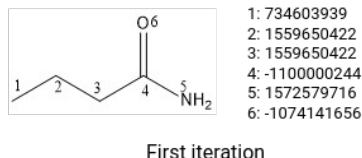


Figure 1.18: Hashed ECFP values for each atom in a compound during the initial iteration.

cluding only the information of the central atom (see Figure 1.18), then adding the information from the 1-bond-distance neighbor atoms, then the 2-bonds-distance neighbors, and so on up to the specified radius value. This information is condensed and hashed into an integer during each iteration (see Figure 1.19). The properties taken into account during the process are the number of immediate non-hydrogen neighbors atoms, the valence minus the number of hydrogens, the atomic number, the atomic mass, the atomic charge, and the number of attached hydrogens, and a 1 or 0 representing whether the atom is contained in at least one ring or not.

Since this type of fingerprint encodes a large amount of information about the molecule,

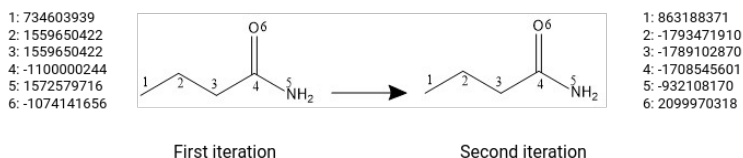


Figure 1.19: Generation of new hashed ECFP values by performing one iteration using an example compound. The initial atom identifiers are shown on the left. Each atom receives a new identifier after the update, shown on the right.

Donor	<chem>[([N;!H0;v3,v4&+1]),([O,S;H1;+0]),n&H1&+o]</chem>
Acceptor	<chem>[([O,S;H1;v2;!(*-=[O,N,P,S]))],([O,S;H0;v2]),([O,S;-]),([N;v3;!N-=[O,N,P,S]]),n&Ho&+o,([o,s;+0;!([o,s]:n);!\$([o,s]:c:n)])]</chem>
Aromatic	<chem>[a]</chem>
Halogen	<chem>[F,Cl,Br,I]</chem>
Basic	<chem>[#7;+,([N;H2&+0]([C,a];!([C,a](=O)))),([N;H1&+0]([C,a];!([C,a](=O))))([C,a];!([C,a](=O))),([N;H0&+0]([C;!C(=O)])([C;!C(=O)])[C;!C(=O)])]</chem>
Acidic	<chem>[\$([C,S](=[O,S,P])-[O;H1,-1])]</chem>

Table 1.1: List of SMARTS patterns used to identify the pharmacophoric features in ECFPs. Adapted from previous work⁷⁰. (Source: RDKit.)

two different molecules can have identical fingerprints due to bit collisions and hashing processes; in that case, it is impossible to identify the single-molecule where the fingerprint is coming from.

ECFPs can be extended to include as well the pharmacophoric functional information for the atoms; one example of this is the FCFPs (Functional-Class Fingerprints), used to identify the pharmacophoric features; the algorithm uses a list of structural keys based on SMARTS patterns as exhibited in Table 1.1.

In general, fingerprints combine several valuable features: they are fast to compute, uses a small space in memory, and are also relatively easy to compare since we just need to take the vector arrays for two molecules and compare corresponding elements bit by bit, where more similar molecules are supposed to have more elements in common. However, expe-

rience shows that finding the best type of fingerprint for a specific application (e.g., virtual screening) depends strongly on the data set.

Fingerprints may include important information about the molecule's composition but contain limited information about its structure and shape.

1.4.3 THREE-DIMENSIONAL DESCRIPTORS

3D descriptors use the 3D molecular representations to calculate specific properties such as solvent-accessible surface area, molecular volume, and spatial pharmacophores^{78,13}.

Solvent-accessible surface area (SASA) aims to improve the van der Waals surface, computed based on van der Waals radius cutoffs for individual atoms²¹. SASA is defined as the molecule's surface area accessible to a solvent; it was initially defined by Lee and Richards⁹⁶. For those molecules with several small cavities or folding parts, inaccessible to solvent interactions, the SASA will be significantly different from the van der Waals surface¹⁶⁷, as depicted in Figure 1.20.

Molecular volume represents the quantity of three-dimensional space enclosed by the region where a molecule is confined; it is often computed as the volume of a substance divided by the amount of substance at a given temperature and pressure. It is worthy to notice that the molecular volume comprises both the inherent molecular volume and the volume of the space between molecules¹⁶⁷.

One method computes the molecular volume by defining a set of spheres around the atomic nucleus for each atom depending on their type⁶⁴. Another method is based on the covalent bond distances and the van der Waals radius to calculate van der Waals volume²¹; as the author says, the volume is approximated by adding all the appropriate volume contributions of

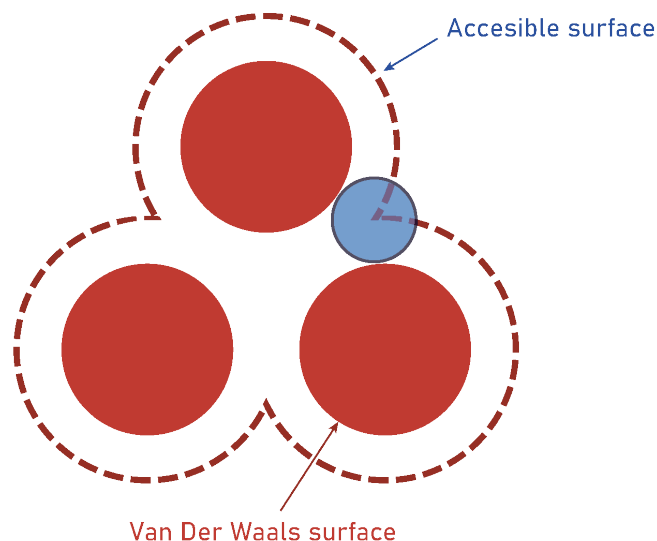


Figure 1.20: Solvent-accessible and van der Waals surface areas in a hypothetical molecule.

atoms and functional groups.

To understand the molecular structure and behavior, the information extracted from the surface area and volume is beneficial; it dictates, for instance, how the molecules bind to ligands in the proteins or how molecules interact with other molecules. This relationship is probably because the surface and the volume are directly correlated with the molecule's shape.

A **spatial pharmacophore** is defined as the 3D arrangement or the spatial configuration of certain features which allow a ligand molecule to interact with a biological target receptor in a specific binding site⁴⁸. Pharmacophore features often include aromatic rings, hydrogen bond acceptors or donors, hydrophobic centroids, anions, and cations.

A pharmacophore can be seen as the largest common feature shared by a set of active molecules towards a target structure; it does not represent an actual association of functional

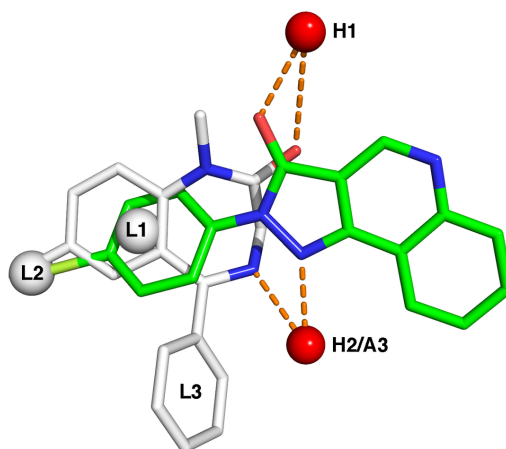


Figure 1.21: Example of a pharmacophore model of the benzodiazepine binding site on the GABA_A receptor. White sticks represent carbon atoms of the benzodiazepine diazepam, while green represents carbon atoms of the nonbenzodiazepine CGS-9896. Red and blue sticks indicate oxygen and nitrogen atoms in both structures. Red spheres H1 and H2/A3 represent hydrogen bond donating and accepting sites in the receptor. L1, L2, and L3 denote lipophilic binding sites. (Source: [Wikimedia](#).)

groups neither an actual molecule. It is an abstract concept for the typical molecular interaction capacity among a set of active compounds¹⁶⁷. In other words, it is possible to identify a pharmacophore by analyzing a set of active molecules that experimentally showed interactions with the target receptor⁴⁸. Figure 1.21 illustrates an example of the pharmacophore-receptor binding interaction.

On a final note, some comparative studies have shown that 2D descriptors such as fingerprints can be more effective than 3D descriptors at identifying molecules with similar features and similar activity^{27,28,103}. Therefore, during this study, we will compare our results mainly with fingerprint-based methods.

1.4.4 GRAPH-BASED DESCRIPTORS

Additionally to the classification mentioned before, there exist methods representing compounds as graphs^{164,8,161}. Those methods often start from the plain atomic graph representation and modify the graphs to suit their specific needs and purposes.

We described the **Atomic graph representation** in section 1.2.3; it is often built using nodes to represent atoms and edges representing chemical bonds between those atoms as they appear in the original molecule. An example can be seen in Figure 1.6.

REDUCED GRAPHS

Atomic graph representations allow customizations to highlight certain features in the molecules. Those modifications often include grouping together some connected nodes, sub-structure fragments, ring systems, acyclic components, among others, and substituting them for single nodes to create a new graph. Those methods are known as reduced graphs^{55,67,69,161}; they group atomic sub-structures in terms of related properties such as pharmacophoric features, hydrogen bonding, ring systems, or other rules.

Reduced graphs are smaller representations of the original atomic graph from a chemical compound; they are smaller since the preliminary information is condensed in feature nodes to give summary abstractions of the chemical structures. Different versions of reduced graphs exist^{10,67,80,8,161}; the main difference relies on the features they summarize and their use. In the ligand-based virtual screening context, the structures are reduced to track down features or sub-structures that can interact with a specific receptor or target from a protein. At the same time, the reduction aims to keep the topology and spatial distribution of those features since the edges are kept connecting the same substructures as they were in the original graph.

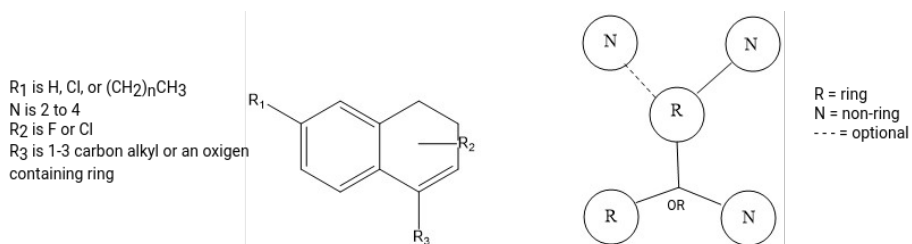


Figure 1.22: Example of a Markush structure and its reduced graph representation using a ring/non-ring reduction scheme.

Reduced graphs are used in the molecular analysis for different applications, ranging from the study of Markush structures^{68,67}, where substitutes are sought to replace chemical sub-structures connected to a common structural feature or central core (see Figure 1.22). Up to the detection of structure-activity relationships (SARs) among molecules. It is SARs where our work will be focused on.

Figure 1.23 depicts examples of different possible graph reductions for the same molecular graph. The examples in the figure aim to emphasize the binding features within the structures; consequently, the reductions illustrated are oriented toward ring systems and hydrogen-bond groups to create either Ring/Feature reduced graphs or Aromatic/Feature reduced graphs.

Since reduced graphs condense the information in the atomic graph, there is a many-to-one correspondence between atoms in the atomic graph and nodes in the reduced graph. In other words, more than one atomic graph could translate into the same reduced graph; figure 1.24 shows an example of such situations.

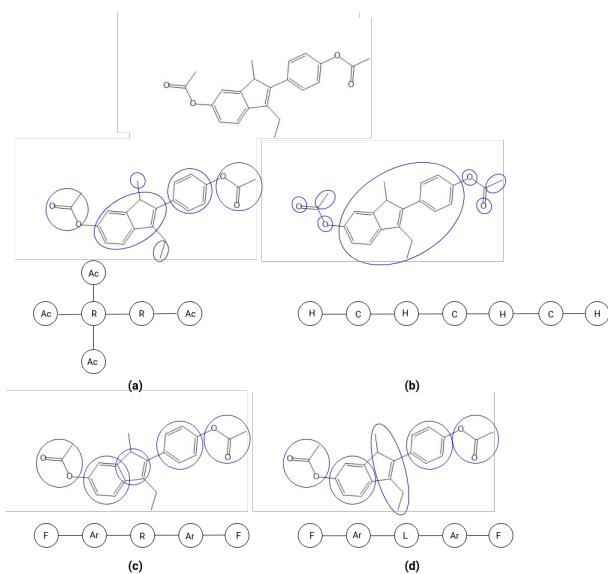


Figure 1.23: Example of different types of reduced graphs obtained from the same molecule. (a) nodes represent ring systems (R) and acyclic components (Ac). (b) nodes represent carbon components (C) and hetero-atom components (H). (c) nodes represent aromatic rings (Ar), aliphatic rings (R), and functional groups (F). (d) nodes represent aromatic rings (Ar), functional groups (F), and linking groups (L).

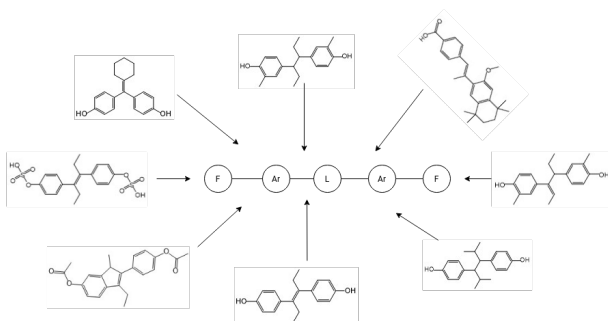


Figure 1.24: Example of several chemical structures that reduce to the same reduced graph based on aromatic rings (Ar) and functional groups (F).

EXTENDED REDUCED GRAPHS (ErG)

ErGs¹⁶¹ are an extension of the reduced graphs described by Gillet *et al.*⁶⁹, which, according to the authors, introduce some changes to better represent the size, shape, and pharmacophoric properties of the molecules. Some of the extensions in the ErGs compared with the reduced graphs include: rings are encoded as ring centroids, and ring atoms are encoded as separate nodes; charged and hydrophobic features are encoded explicitly; resulting feature nodes are connected following the shortest path between the reference nodes in the original chemical graph.

In ErGs, nodes can be a single or a combination of the following features: hydrogen-bond donor, hydrogen-bond acceptor, positive charge, negative charge, hydrophobic group, or aromatic ring system. There are also featureless nodes, which work as links between the main features and help them keep their spatial distribution. These featureless nodes can be carbon or non-carbon link nodes. Figure 1.25 depicts the step-by-step process to convert a molecular compound to its corresponding ErG. Even though the resulting graph from ErG is more complex than the reduced graph, the inter-feature distances between nodes and the 3D positional representation are more accurate than the original molecular graph¹⁷. An example of how the ErG references the positional information from the original molecule is illustrated in Figure 1.26.

In the specific context of ErGs, the fingerprint descriptor used by Stiefl *et al.* is based on the method used by Kearsley *et al.* for their binding property pairs⁸⁶, which, in turn, is an extension of the atom pairs described by Carhart *et al.*³⁶.

ErGs have been demonstrated to be a powerful tool for virtual screening¹⁶¹; they can be used as an abstraction layer from the complex physicochemical-atomic world and leave the

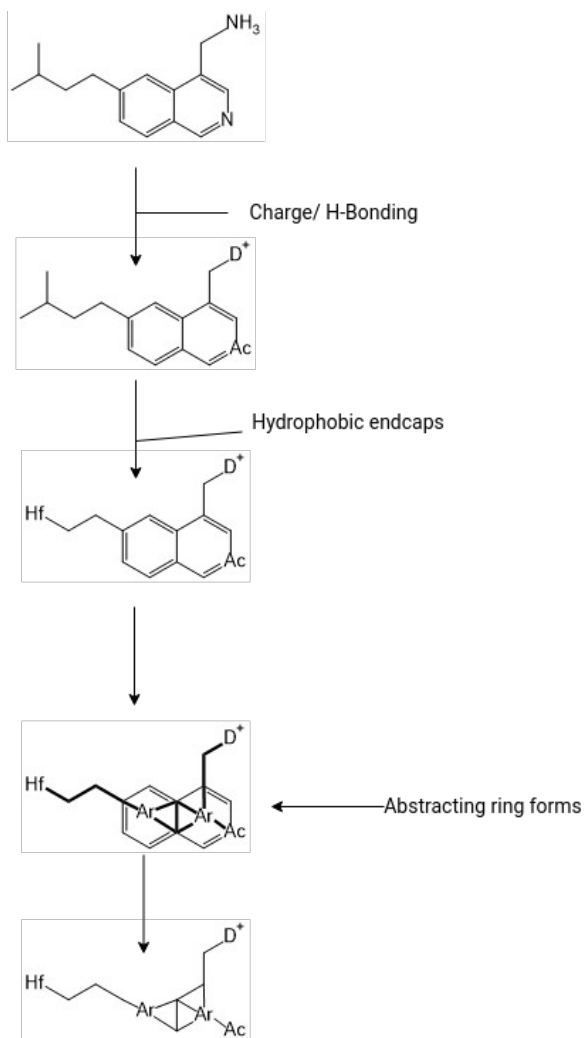


Figure 1.25: Example of the reduction of a chemical graph into its ErG representation as described by Stiefl *et al.*¹⁶¹ (D) H-bond donor; (Ac) H-bond acceptor; (Hf) hydrophobic group; (Ar) aromatic ring system; (+/-) positive/negative charge.

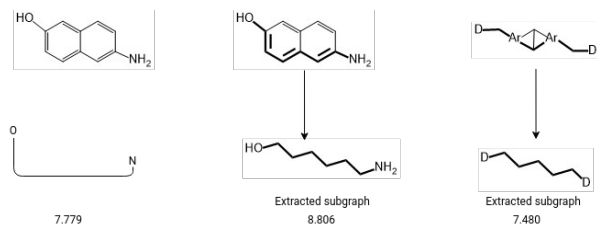


Figure 1.26: Spatial inter-feature distances in Angstrom (Å) of an example molecule.

path clear to work directly with the pharmacophoric chemical information inside the molecular structures.

The resulting ErG from a molecule can be encoded into a *fingerprint descriptor*, as it was used in the original work by Stiefl *et al.*¹⁶¹, or into a *graph-descriptor* obtained directly from the reduced graph after the ErG reduction. The *fingerprint descriptor* presented by Stiefl *et al.* uses only the property points or nodes with assigned features. As the authors point out, this methodology can be described as a hybrid approach between reduced graphs⁶⁹ and binding property pairs⁸⁶ (mentioned before in section 1.4.2), with the distinction that in this case, the point pairs are built using substructures of the form:

$$\text{Property Point 1} - (\text{distance}) - \text{Property Point 2}$$

where the inter-feature distances are computed from the reduced graph.

The *graph descriptor* instead uses the graph representation obtained as an outcome after applying the ErG reduction to the original molecular graph, as illustrated in Figure 1.27. The Figure exhibits an example of an ErG reduction where the upper half of the image shows a chemical molecule with its pharmacophoric substructures highlighted; the lower half shows the ErG obtained from that molecule. This graph descriptor is the preferred option throughout this work since our primary comparison method is based on a graph matching methodology.

1.5 SIMILARITY MEASURE

The concept of molecular similarity refers to the pairwise comparison among chemical compounds or molecules; it can be in terms of physical properties, behavioral qualities, or struc-

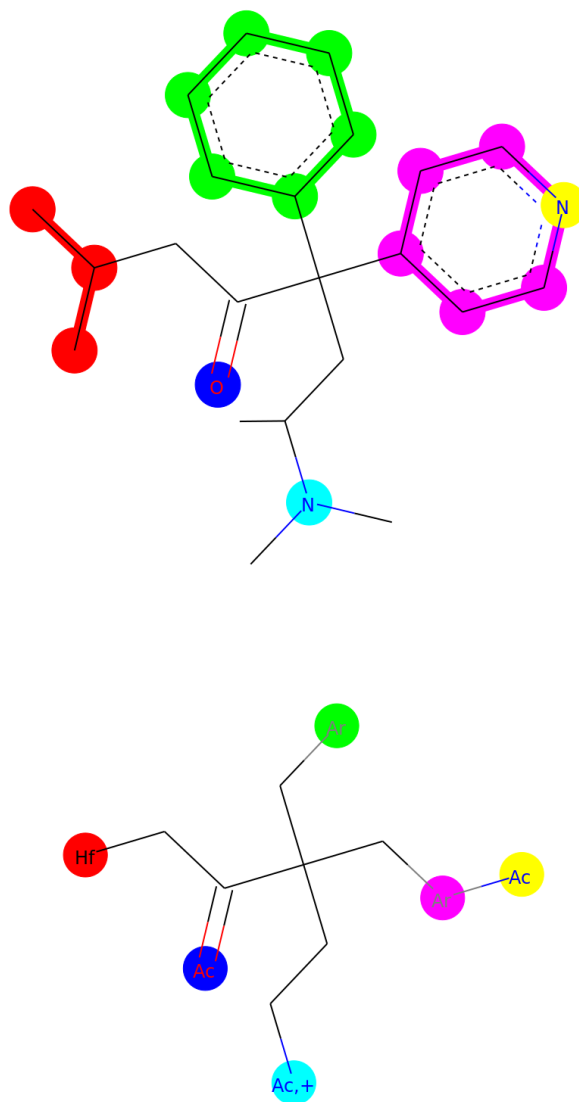


Figure 1.27: Example of molecule reduction using ErG. The original molecule is at the top, and its ErG representation is at the bottom. Ac: H-bond acceptor; Hf: hydrophobic group; Ar: aromatic ring system; +: positive charge. Colors are used to show how different parts of the original structure are reduced to nodes in the ErG.

tural features. This concept is of utmost importance in virtual screening, drug design, bioactivity prediction, and cheminformatics^{84,122}. These applications are based on the principle stated by Johnson and Maggiora, saying that structurally similar molecules are likely to have similar properties⁸⁴. Therefore, they are commonly used in the drug discovery industry to find drug candidates and perform scaffold hopping²⁶. Scaffold hopping is an area of virtual screening focused on identifying molecules with similar activity but belonging to different lead series; it is helpful, especially when we need to pinpoint compounds with a particular behavior while staying away from a patent space.

The first works related to molecular similarity searching were developed using fragment bit-strings as a tool for substructure search^{36,183}. Ever since the beginning of molecular screening in the 1980s, significant efforts have been made to create new and better descriptors to describe molecules in a way that correctly represents their activity and properties. One of the major challenges in the molecular similarity search is choosing a suitable descriptor for a particular task; research is ongoing into new methods for measuring molecular similarity, mainly because of the difficulty to describe the relationship between the complex chemical space and its biological activity. In this study, we work with two different descriptors based on the ErGs, the first one is the graph obtained after the reduction and the second one is the fingerprint obtained as described by Stiefl *et al.* in their article¹⁶¹.

Once the descriptor has been decided, it is time to decide on the most convenient similarity measure to compare such descriptors depending on the applications and needs of the study. Three similarity measures have been used to perform reduced graph comparisons: The first one (FP-based) maps the reduced graphs into a 2D fingerprint^{69,10,161}; the second one (SED-based) maps them into sets of shortest paths⁸⁰; the third one (Graph-based) makes the com-

parison directly on the graphs via the graph edit distance method⁶¹. These three similarity measures are used in this study, so let us introduce each of them:

1.5.1 FP-BASED METHOD

Generally speaking, fingerprint-based methods encode the molecular structure into a vector array called a fingerprint (see Figure 1.28). Each bin denotes the presence or absence of a particular feature in a fragment of the molecule. Then, the similarity between two molecules A and B is computed by comparing their fingerprints with a measure or intersection coefficient to find matching elements in both arrays since it is assumed that similar molecules tend to have a greater number of fragments in common. The most commonly used similarity measure when comparing compounds represented through their fingerprints is the Tanimoto coefficient, also known as the Jaccard index or Intersection over Union, exhibited in Equation 1.1.

$$T_s(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \quad (1.1)$$

where $A \cap B$ stands for the intersection and $A \cup B$ is the union of the bins in the fingerprints A and B. Furthermore, $|A|$ and $|B|$ represent the number of non-empty bins in sets A and B. The resulting value goes from 0 to 1, with 1 representing the highest degree of similarity.

As we mentioned before, in the *fingerprint descriptor* presented by Stiefl *et al.*, the property pairs are built using substructures of the form:

$$\text{Property Point 1 - (distance) - Property Point 2}$$

where the inter-feature distances are computed from the reduced graph.

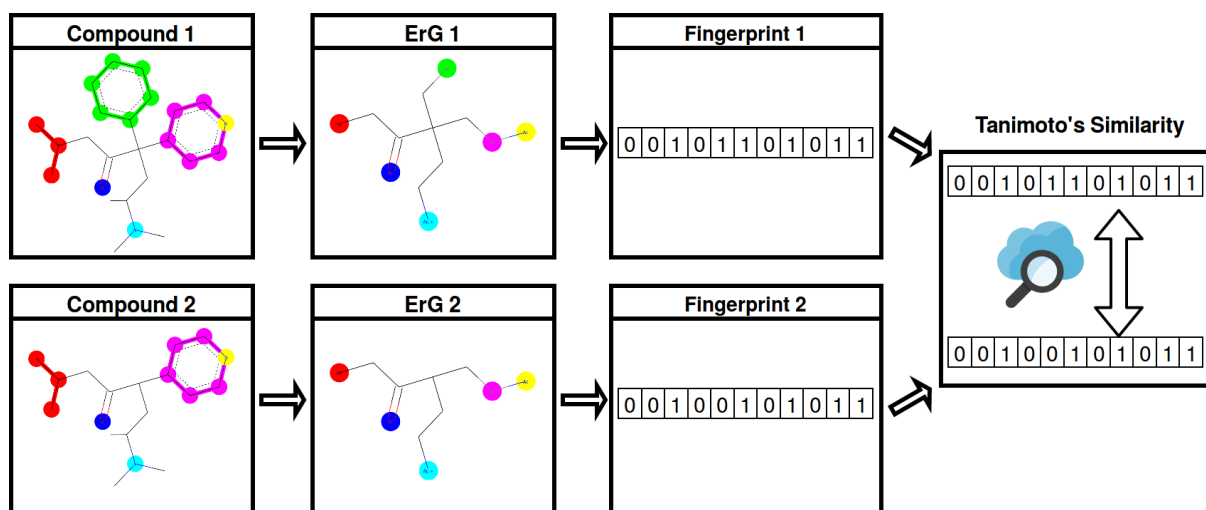


Figure 1.28: Fingerprint-based method flowchart

The resulting descriptor vector then encodes, for each bin, a specific property-property-distance triplet. This bin is incremented by a factor every time a corresponding triplet is found in the structure under study. The increment factor is a user-definable parameter, which can be set depending on the dataset traits and user needs. Our experiments used the default value (0.3) for this factor.

Different Tanimoto similarity coefficients can be applied depending on the type of descriptor being binary or algebraic. The summation form is used for ErG descriptors (Equation 1.2).

$$T_s(A, B) = \frac{\sum_{i=1}^m n_{A,i} n_{B,i}}{\sum_{i=1}^m (n_{A,i})^2 + \sum_{i=1}^m (n_{B,i})^2 - \sum_{i=1}^m n_{A,i} n_{B,i}} \quad (1.2)$$

where m is the size of the ErG vector, $n_{A,i}$ is the i th entry of the vector in compound A, and $n_{B,i}$ is the i th entry of the vector in compound B.

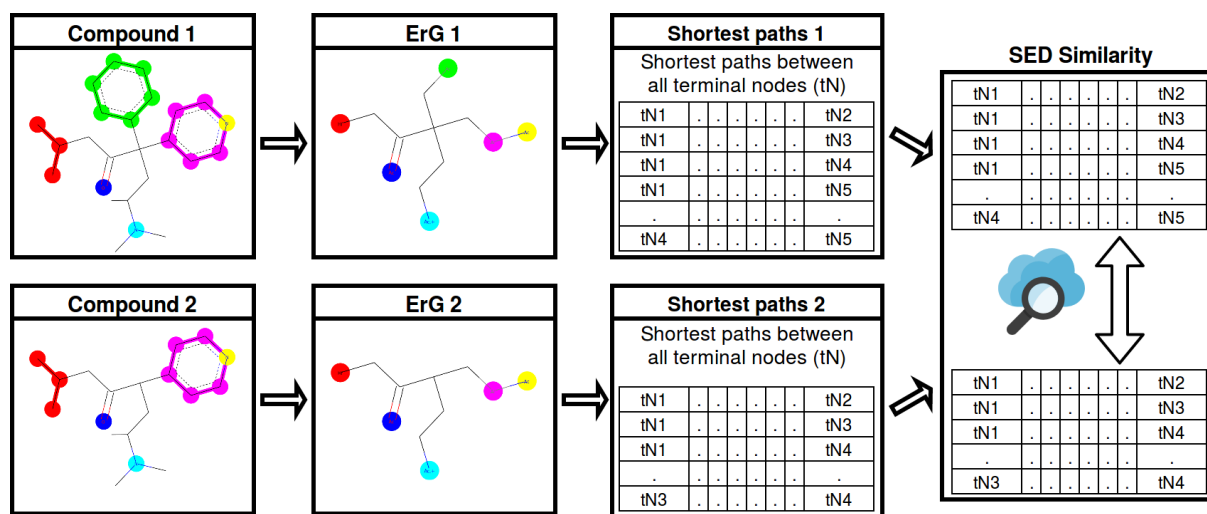


Figure 1.29: SED-based method flowchart

The computation of the ErG reduction and the Fingerprint-based similarity value presented in this study uses the RDKit¹³², an open-source cheminformatics toolkit made available under the Berkeley Software Distribution (BSD) license.

1.5.2 SED-BASED METHOD

Harper *et al.*⁸⁰ proposed a string edit distance procedure to quantify the similarity between reduced graphs (Figure 1.29); the distance between two reduced graphs A and B is calculated as the cost of transforming one graph into the other by substituting, inserting, and deleting nodes. Different node operations can yield different transformation costs. For instance, for Harper *et al.*, the substitution of a “donor” to a “donor and acceptor” node was assigned a low cost since they seemed to be more closely related than a “donor” and a “negatively ionizable group”, whose substitution was assigned a high cost. It is important to emphasize that the edit costs used for node operations by Harper *et al.* were based on intuition only.

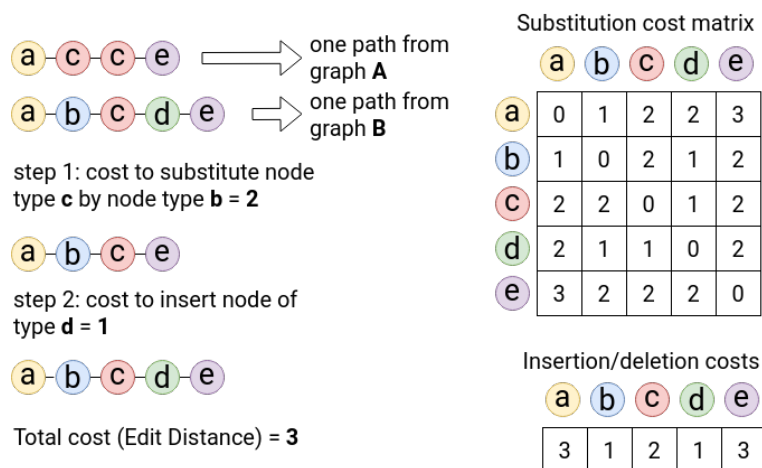


Figure 1.30: String edit distance process to convert one path from graph A into one path from graph B. The total edit distance is 3 based on the substitution and insertion/deletion costs presented on the right side of the image.

The graph transformation is performed by comparing the shortest paths between terminal nodes (nodes of degree one) in graphs A and B. This type of string edit distance methodology is commonly used in computational biology to compare sequences by counting the number of edit operations required to go from one sequence to another. The distance between two reduced graphs is the maximum value obtained from the minimum costs after all paths have been compared in A and B.

Figure 1.30 illustrates the string edit distance methodology. On the left side of the image, we can see the step-by-step procedure to convert one path from graph A into one path from graph B; the substitution costs and the insertion/deletion costs are located on the right side of the image.

The SED-based similarity value presented in this study is computed with an in-house implementation that uses C++ and Python languages, following the steps described by Harper *et al.*⁸⁰.

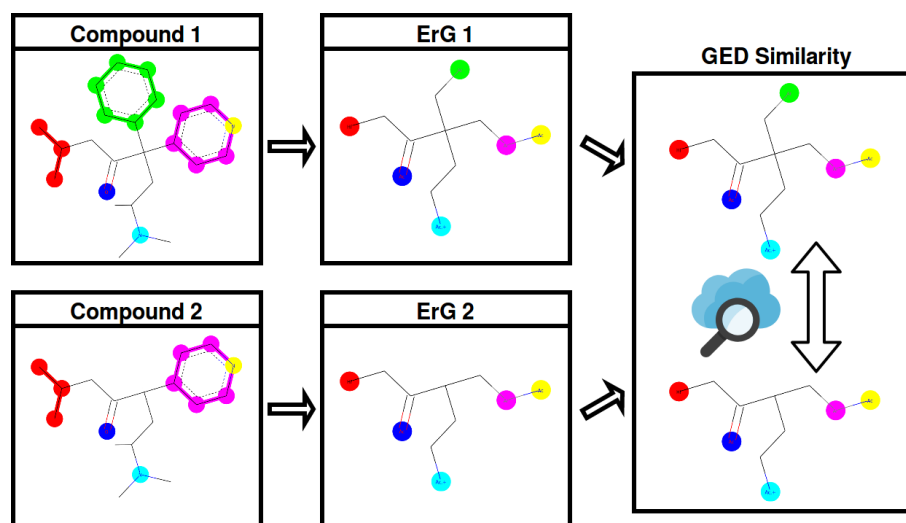


Figure 1.31: Comparison of two molecules comprising two steps. First, we extract the ErGs. Second, we apply the GED.

1.5.3 GED-BASED METHOD

Figure 1.31 shows a molecular comparison procedure using the GED-based similarity method. The GED is defined as the minimum cost of modifications required to transform one graph into the other (see Figure 1.32). These modifications are called edit operations, and six of them have been defined: insertion, deletion, and substitution of both nodes and edges. The idea is similar to the SED-based presented before. Still, in this case, the transformation is carried out directly on the graphs instead of doing it on the sequences obtained from the shortest paths between terminal nodes. As shown in Figure 1.33, using a methodology applied directly over graphs allow us to skip the step to convert those graphs into 2D representations such as fingerprints or shortest paths.

In this way, for every pair of graphs A and B , there are several $editPath(A, B) = (\sigma_1, \dots, \sigma_k)$ that transform one graph into the other, where each σ_i denotes an edit operation.

Figure 1.34 shows a possible edit path holding the transformations necessary to go from

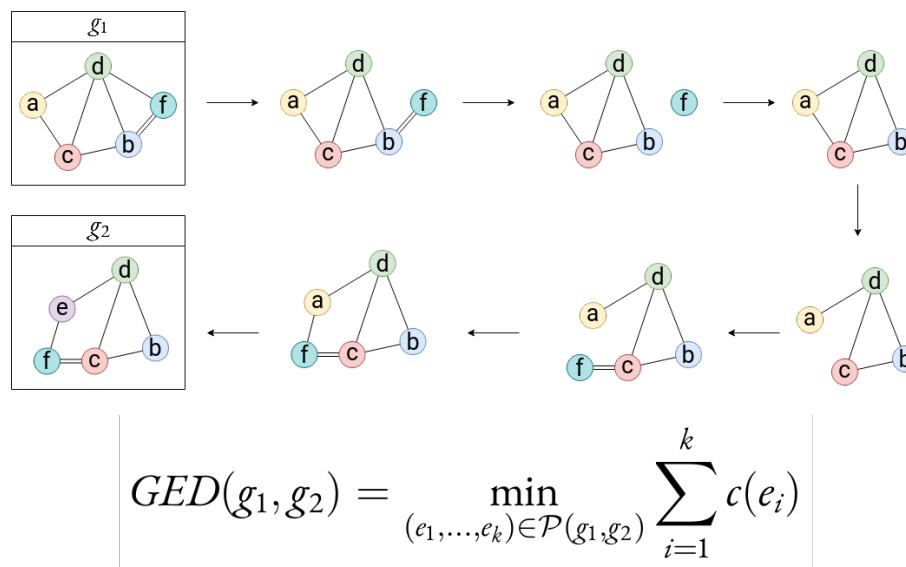


Figure 1.32: Step-by-step GED transformations to go from g_1 to g_2 .

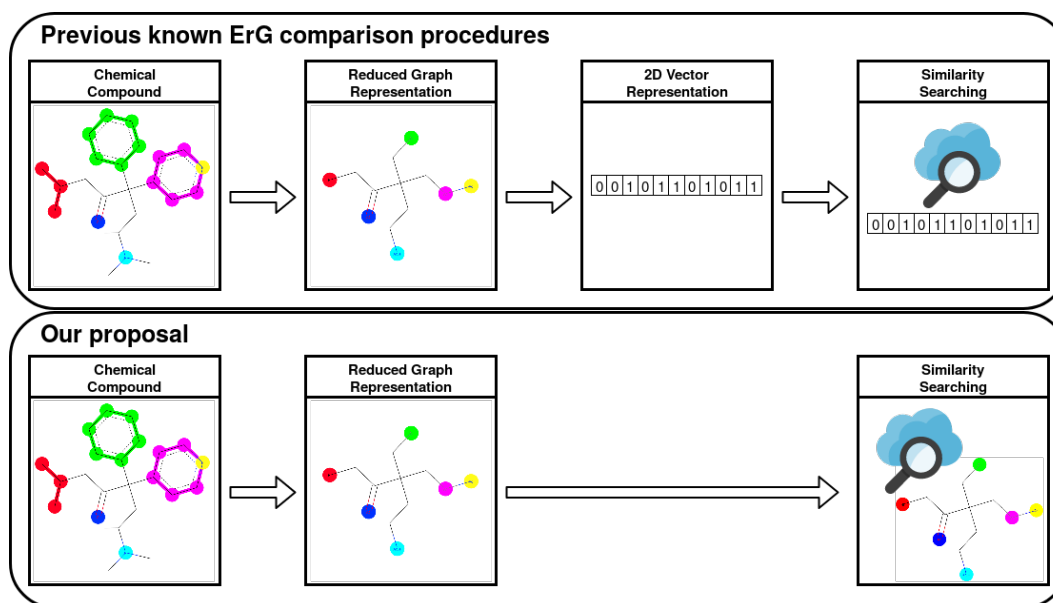


Figure 1.33: Molecular comparison flowcharts. Difference between traditional ErG methods and our proposal

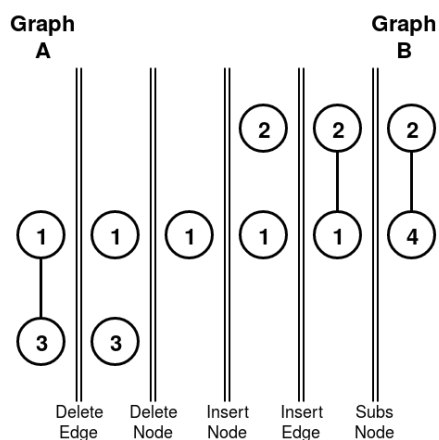


Figure 1.34: An edit path that transforms graph A into graph B

graph A to graph B; it consists of the following five edit operations: Delete edge, Delete node, Insert node, Insert edge, and Substitute Node. The substitution operation in the last step is needed since it is assumed that the attributes in nodes 1 and 4 are different.

Edit costs have been introduced to quantitatively evaluate each edit operation and ultimately determine which edit path has the minimum total cost. The edit costs aim to designate a coherent transformation penalty in proportion to how it modifies the transformation sequence.

The final edit cost for a given edit path is obtained by adding up all individual transformation costs. Figure 1.34 shows an example of an edit path, in this case, the transformation sequence is the sum for the cost of deleting an edge, plus the cost of deleting node 3, plus the cost of inserting node 2, plus the cost of inserting an edge, plus the cost of substitution from node 1 to node 4. The process of adding up all transformation costs and getting the edit cost is repeated for any possible edit path able to transform one graph into the other. In the end, the GED resulting value for any pair of graphs A and B is defined as the minimum cost under all those possible edit path sequences.

Usually, the GED is normalized according to the number of nodes in both graphs compared by dividing the obtained GED by their average number of nodes. This normalization is performed to make the measurement independent of the graphs' size and thus avoid favoring smaller graphs since larger graphs tend to have higher overall transformation costs.

1.6 MACHINE LEARNING FOR MOLECULAR ANALYSIS

Machine learning is a subset of artificial intelligence and has been defined as “the science (and art) of programming computers so that they can learn from data”⁶⁵. The technological tools necessary for machine learning were developed in the 1950s and improved in the 1980s. Still, it was just over the last ten years that computer hardware advances made those tools applicable to real-world problems. Machine learning algorithms are now ubiquitous in the technology and science fields, taking part in various applications from online shopping to protein structure prediction¹⁴⁹; some of the most renowned are: image recognition, recommender systems, language translation, voice recognition, and drug discovery.

Machine learning is transforming all areas of science; disciplines are now utilizing big datasets that are impossible to analyze manually through human observation. For instance, it is common to use machine learning to analyze cell images and classify them as normal or cancer cells to decide future treatments¹²⁶. Molecular simulation and property predictions in molecular physics are other areas deeply impacted by machine learning algorithms, obtaining significant improvements in the complex and time-consuming calculations by learning the patterns in the data^{125,53} while reducing the execution time by orders of magnitude as compared with traditional computational chemistry methods³³.

A standard way of functioning in machine learning experiments is working with an input

dataset used as “training data”, which the algorithm uses to learn and make predictions or make decisions without the need to be programmed for that purpose explicitly⁹⁰. We are currently at a time when there is widely available scientific data and different methods to process it; therefore, combining those two things properly to learn from patterns in the data can allow us to develop significant scientific advances.

1.6.1 OPTIMIZING THE EDIT COSTS FOR GED

Mathematical optimization is a branch of applied mathematics; it can be defined as the science of finding the best element (out of a set of available alternatives) to solve a mathematically defined problem according to specific criteria¹⁵⁶. Those problems are obtained from physical reality or manufacturing and management systems; they can arise from disciplines such as computer science, engineering, operations research, and economics.⁴⁹ In order to perform the optimization, the algorithm maximizes or minimizes a real function by iterating systematically among a set of input values and computing the output value of the function; this process is repeated until it obtains a feasible solution (see Figure 1.35). This function is usually known as the objective function, loss function, cost function, energy function, or fitness function. On the other hand, the best result (minimizes or maximizes depending on the problem) for the objective function is called an optimal solution.

Usually, in mathematics, the convention states all optimization problems in terms of “minimization”. This convention intends to avoid confusion between minimization and maximization problems. Once the objective function has found an optimal solution, this minimum might be one of two options: local minimum or global minimum. A local minimum of a function is where the function value is at least as good as any nearby point but possi-

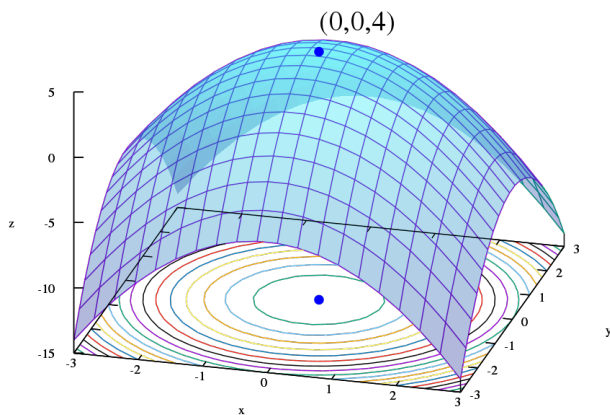


Figure 1.35: Graph of a function. The global maximum is at $(x, y, z) = (0, 0, 4)$. (Source: [Wikimedia](#).)

bly worse than a distant point. A global minimum is a point where the function value is at least as good as any other feasible point. Generally, there may be several local minima for a function; therefore, finding an arbitrary local minimum is relatively straightforward. On the other hand, finding the global minimum of a function is significantly more difficult (see Figure 1.36), requiring complex optimization techniques, such as genetic algorithms.

A genetic algorithm (GA)^{71,162} is an optimization methodology belonging to the group of evolutionary algorithms¹⁷³ and inspired by the process of natural biological evolution. It includes mechanisms such as reproduction, mutation, recombination (also known as crossover), and selection, where the available candidate solutions take part in the optimization problem as individuals in a population. GAs are often used in optimization and search problems to obtain high-quality results¹¹⁵, which might also be helpful to find global minima.

As mentioned above, the edit costs used for node operations by Harper *et al.* and Garcia-

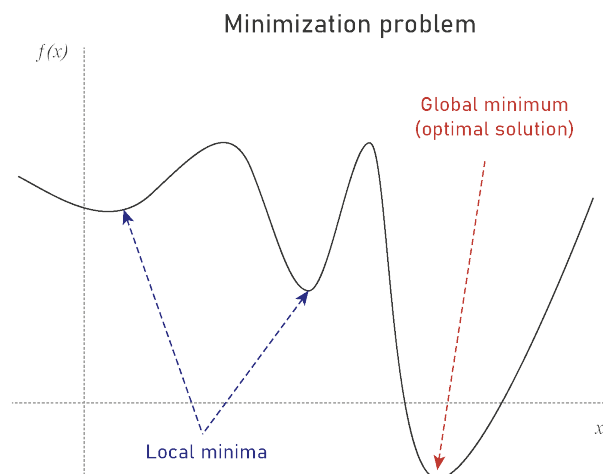


Figure 1.36: Example of local and global minima in a hypothetical optimization problem.

Hernandez *et al.* in the original SED-based⁸⁰ and GED-based⁶¹ methodologies were based on intuition; hence, it is reasonable trying to improve those values through an optimization procedure. That is precisely what Birchall *et al.* did in their optimization work for the SED-based methodology¹⁸ and what we will do in chapter 4 of this document. Both studies use a GA to identify optimized sets of edit costs employing an objective function that minimizes the classification error over a group of molecules with different activity classes extracted from well-known available drug databases (see Figure 1.37).

1.6.2 GRAPH EMBEDDINGS TO INFER THE EDIT COSTS FOR GED

Machine learning and graph analytics have advanced enormously in the last years due to the vast natural presence of graphs in real-world applications such as social networks, word co-occurrence networks, and communication networks. Those advances reported several high-impact new methods in the top journals every year¹⁸⁶, yielding insights into different topics

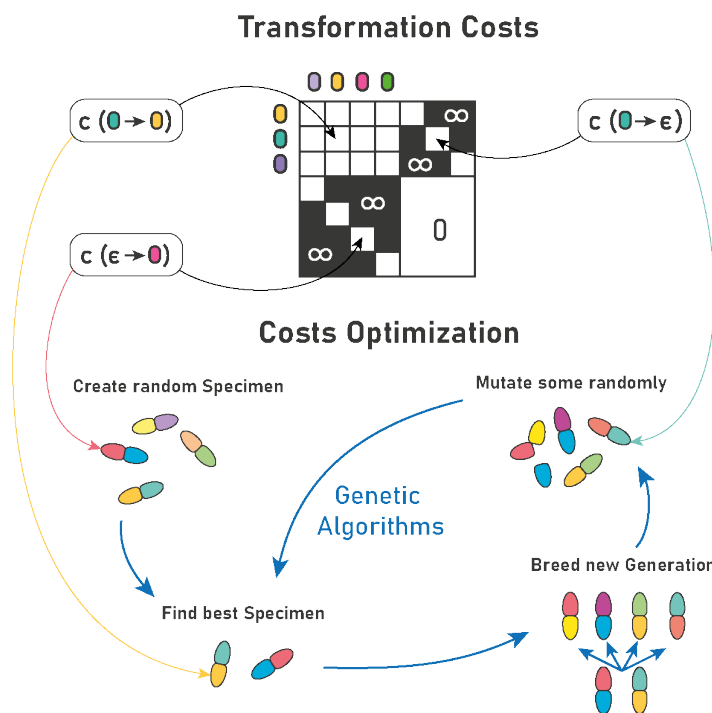


Figure 1.37: Representation of a genetic algorithm used to optimize the transformation costs (insertion, deletion, and substitution) used for the GED.

such as the structure of society, language, and different patterns of communication. Some examples of these advancements are: graph neural-networks⁷⁴, which directly process graphs and extend recursive neural networks; they can be applied on most of the types of graphs; another example is the methods that use the low-dimensional representation of graph nodes in vector space, also known as graph embeddings⁷⁵.

Graph embeddings are trained to catch the graph's topology, the relationship between vertices, and even some other information about the subgraphs present in the structure. In the application of word co-occurrence, many natural language processing (NLP) techniques treat words as atomic units since these words are represented as one-hot indices in a vocabulary¹¹². An example of such applications is the N-gram model used for statistical language modeling, trained with trillions of words²⁵. A popular application of the N-gram model is Word2vec¹¹², which learns the embeddings of words and later confirms that embedding vectors of similar words are also near in the vector space. The same concept has been adapted to analyze graphs⁷⁷; protein sequences for the classification of protein families⁴; and even for molecular substructures derived from the Morgan algorithm, those substructures are treated as “words,” and the whole compounds play the role of “sentences”⁸³.

In this work, specifically in chapter 5, we use the NLP techniques mentioned before to obtain abstract low-dimensional vectors from the different types of nodes in a group of molecules represented as ErGs. The pairwise distances among those vectors in the vector space are interpreted as a dissimilarity value used as the transformation costs in the GED methodology. The GED will be applied as a similarity estimation among molecules to predict the bio-activity group they belong to.

The main goal of this study is to implement a graph-only driven molecular comparison

methodology without the array representations. The comparison is made directly on graphs, and there is no need to perform any transformation from graphs into 2D vectors (Figure 1.33). Additionally, we implement optimization tools to improve the recognition ratio when classifying molecules represented as ErGs according to their biological activity. The optimization techniques help determine better values to be used as edit costs when computing the graph edit distance^{2 145,60,157}, which works as a dissimilarity measure between molecular graphs based on ErGs.

The outline of this document is as follows. The second chapter presents and explains the general materials and methods used in this work, including datasets, the GED methodology, and the optimization processes. The third, fourth, and fifth chapters describe three different works using the methodologies defined in the second chapter, including the specific computational results for each work. Chapter six concludes the document with a final discussion.

*If we knew what it was we were doing, it would not be called
research, would it?*

Albert Einstein.

2

Methodology

2.1 DATASETS

During the experiments described in chapters 3 and 4, we used six publicly available datasets: ULS and UDS¹⁸⁴, GDD and GLL⁶², DUD Release 2¹²¹ (DUD-E), NRLiSt BDB⁹⁴, MUV¹⁴¹, and a dataset from Comparative Analysis of Pharmacophore Screening Tools¹⁴⁴ (CAPST). All these datasets were normalized in a format ready to use by the LBVS benchmarking platform developed by Skoda and Hoksza¹⁵⁵. This platform is similar in concept to another benchmarking platform developed by Riniker and Landrum and has extended some of its features¹³⁹. The datasets within this platform consist of several “selections” of active and inactive molecules arranged according to the target used to perform the physical tests. Each selection is separated into two sub-groups named “test” and “train” sets to be readily used for machine learning applications. Table 3.1 shows all targets available in the datasets.

Furthermore, as for chapter 5, the datasets used will be described later in the same chapter.

2.2 MOLECULAR REPRESENTATIONS

Section 1.4 discussed different molecular descriptors used to represent physicochemical, structural, or behavioral traits from molecules.

Throughout this work, we will be using the ErG molecular descriptor described by Stieff *et al.*¹⁶¹, where node features encode pharmacophore-type information. According to the authors, this methodology can be described as a hybrid approach between reduced graphs⁶⁹ and binding property pairs⁸⁶ (both mentioned before in section 1.4). ErGs will be used in two ways: fingerprints, as employed in the original work by the authors¹⁶¹ and described before in this document in section 1.4.2; and attributed graphs, similarly to the representation

Dataset	Targets used
ULS-UDS	5HT _{1F} _Agonist, MTR _{1B} _Agonist, OPRM_Agonist, PE _{2R3} _Antagonist
GLL&GDD	5HT _{1A} _Agonist, 5HT _{1A} _Antagonist, 5HT _{1D} _Agonist, 5HT _{1D} _Antagonist, 5HT _{1F} _Agonist, 5HT _{2A} _Antagonist, 5HT _{2B} _Antagonist, 5HT _{2C} _Agonist, 5HT _{2C} _Antagonist, 5HT _{4R} _Agonist, 5HT _{4R} _Antagonist, AA _{1R} _Agonist, AA _{1R} _Antagonist, AA _{2AR} _Antagonist, AA _{2BR} _Antagonist, ACM ₁ _Agonist, ACM ₂ _Antagonist, ACM ₃ _Antagonist, ADA _{1A} _Antagonist, ADA _{1B} _Antagonist, ADA _{1D} _Antagonist, ADA _{2A} _Agonist, ADA _{2A} _Antagonist, ADA _{2B} _Agonist, ADA _{2B} _Antagonist, ADA _{2C} _Agonist, ADA _{2C} _Antagonist, ADRB ₁ _Agonist, ADRB ₁ _Antagonist, ADRB ₂ _Agonist, ADRB ₂ _Antagonist, ADRB ₃ _Agonist, ADRB ₃ _Antagonist, AG _{2R} _Antagonist, BKRB ₁ _Antagonist, BKRB ₂ _Antagonist, CCKAR_Antagonist, CLTR ₁ _Antagonist, DRD ₁ _Antagonist, DRD ₂ _Agonist, DRD ₂ _Antagonist, DRD ₃ _Antagonist, DRD ₄ _Antagonist, EDNRA_Antagonist, EDNRB_Antagonist, GASR_Antagonist, HRH ₂ _Antagonist, HRH ₃ _Antagonist, LSHR_Antagonist, LT _{4R1} _Antagonist, LT _{4R2} _Antagonist, MTR _{1A} _Agonist, MTR _{1B} _Agonist, MTR _{1L} _Agonist, NK _{1R} _Antagonist, NK _{2R} _Antagonist, NK _{3R} _Antagonist, OPRD_Agonist, OPRK_Agonist, OPRM_Agonist, OXYR_Antagonist, PE _{2R1} _Antagonist, PE _{2R2} _Antagonist, PE _{2R3} _Antagonist, PE _{2R4} _Antagonist, TA _{2R} _Antagonist, V _{1AR} _Antagonist, V _{1BR} _Antagonist, V _{2R} _Antagonist
CAPST	CDK ₂ , CHK ₁ , PTP _{1B} , UROKINASE
DUD-E	COX ₂ , DHFR, EGFR, FGFR ₁ , FXA, P ₃₈ , PDGFRB, SRC, AA _{2AR}
NRLiSt_BDB	AR_Agonist, AR_Antagonist, ER_Alpha_Agonist, ER_Alpha_Antagonist, ER_Beta_Agonist, FXR_Alpha_Agonist, GR_Agonist, GR_Antagonist, LXR_Alpha_Agonist, LXR_Beta_Agonist, MR_Antagonist, PPAR_Alpha_Agonist, PPAR_Beta_Agonist, PPAR_Gamma_Agonist, PR_Agonist, PR_Antagonist, PXR_Agonist, RAR_Alpha_Agonist, RAR_Beta_Agonist, RAR_Gamma_Agonist, RXR_Alpha_Agonist, RXR_Alpha_Antagonist, RXR_Gamma_Agonist, VDR_Agonist
MUV	466, 548, 600, 644, 652, 689, 692, 712, 713, 733, 737, 810, 832, 846, 852, 858, 859

Table 2.1: Input data used for the experiments. The column entitled 'Dataset' contains the name of each dataset, and the column entitled 'Targets used' contains the name of the targets used during the experiments for each dataset. Note that in the result plots shown below, per-target points are arranged in the same order as they are in this table.

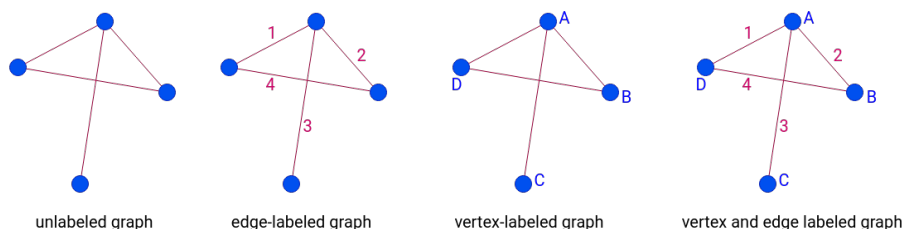


Figure 2.1: Illustration with different versions of labeled graphs.

used by Harper *et al.* in their work about the similarity between reduced graphs⁸⁰.

2.2.1 ATTRIBUTED GRAPHS

Graph theory is a field of science devoted to the analysis of graphs. It dates back to 1736 when Leonard Euler published his paper on “The Seven Bridges of Königsberg”, which is considered the first paper in graph theory¹⁶. In graph theory, a graph is a mathematical structure used to describe pairwise relationships between objects. Those objects are represented in the graph as nodes (also known as vertices), and pairwise relationships are represented as edges (also known as links). Attributed graphs’ name comes because nodes and edges are associated with a set of attributes (or labels) holding information to describe the features in those objects (represented by the nodes) and those relationships (represented by the edges), as depicted in Figure 2.1. An increase in prediction accuracy has been observed by annotating nodes and edges with additional attributes^{22,59,79}.

Graphs have been used in different fields to represent a wide diversity of structures such as computer networks, social networks, or chemical structures (see Figure 2.2),

The sub-field of *Chemical graph theory* uses the molecular graph as a means to model molecules. This graph model is intuitive since nodes represent atoms and edges represent chemical bonds. For the attributed molecular graph, nodes are usually labeled with the type

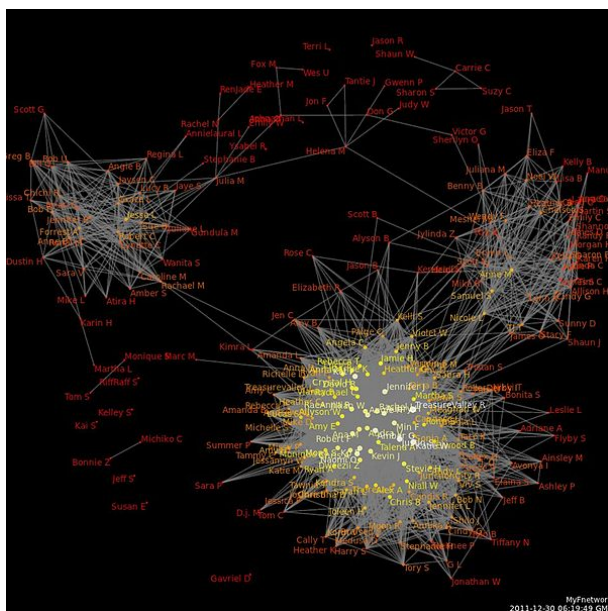


Figure 2.2: Social network graph illustrating friendships among a group of users. (Source: Wikimedia.)

of the corresponding atom, and edges are labeled with the type of the corresponding chemical bonds¹⁰⁶. In this work, nodes are labeled with the corresponding pharmacophoric features obtained with the ErG methodology (see Figure 2.3).

Typically, molecular graphs do not bear any information about the 3D distribution of atoms or bonds. Therefore, they cannot pinpoint the difference between geometric isomers or other stereoisomers. The concepts of chirality and stereochemical information were defined before in section 1.2.4.

2.3 MOLECULAR COMPARISON

Once the molecular structure has been represented as an ErG, the next step is to define a similarity procedure to measure how similar or different two molecules or ErGs are.

In this document, we make use of two reported methods, and we present a new one. First

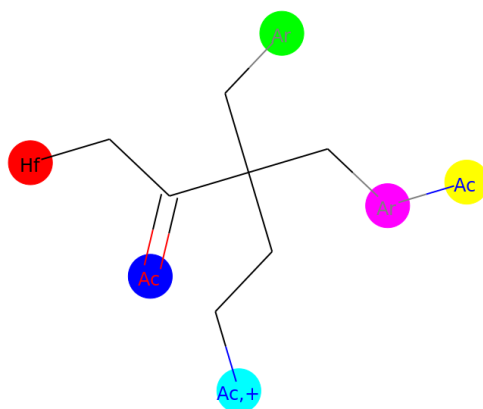


Figure 2.3: Example of an ErG with the corresponding pharmacophoric features. Ac: H-bond acceptor; Hf: hydrophobic group; Ar: aromatic ring system; +: positive charge.

method: Fingerprint-based, used by the authors in the paper that reported ErGs for the first time¹⁶¹ (see section 1.5.1 for further details). Second method: String Edit Distance (SED)-based, presented by Harper *et al.*⁸⁰ (see section 1.5.2 for further details). Third method: our new proposal, GED-based, compares ErGs directly with no intermediate representation. We explained this GED-based methodology before in section 1.5.3. Let us now take a more in-depth description of the graph matching and graph edit distance definitions.

2.3.1 GRAPH MATCHING

Graph matching is a fundamental problem in graph theory and pattern recognition, having applications in various fields such as computer vision and computational biology, where problems are often formulated as an attributed graph matching problem. The term graph matching refers to the pairwise comparison between two or more graphs. This comparison takes two graphs and performs a pairwise correspondence (or mapping) between all nodes

from one graph and all nodes from the other while intending to preserve the relationships between those nodes in their corresponding graph.

Many problems of interest in science can be formulated as a problem of correspondence, where mapping is needed between two separate sets of points. Since typically, those point sets have different internal structures, they are often treated not simply as two sets of points but as two separate graphs. Therefore the correspondence problem is often referred to as graph matching⁴³.

Due to its combinatorial nature, we have two ways to solve the graph matching problem: exactly, with a very restricted setting, or approximately, developing approximate relaxations to the problem. Graph matching problems have been extensively studied, and research focuses mainly on designing more accurate and faster algorithms to approximately solve the quadratic assignment problem, which is typically NP-hard^{3,34}.

An example NP-hard problem related to graph matching is the maximum common subgraph problem, which attempts to find the largest possible subgraph in two graphs (see Figure 2.4). However, graph similarity does not necessarily require an exact solution to be a useful measure in a broad range of applications.

Maximum common subgraph techniques were employed before by Barker *et al.* to compare chemical compounds represented as reduced graphs and thus compute a similarity value between pairs of molecules⁹. Other similarity measures for attributed graphs have been defined^{30,31,111,120,145,175}. A popular one is the graph edit distance, which we will focus on throughout this work and the one we will describe in-depth in the next section.

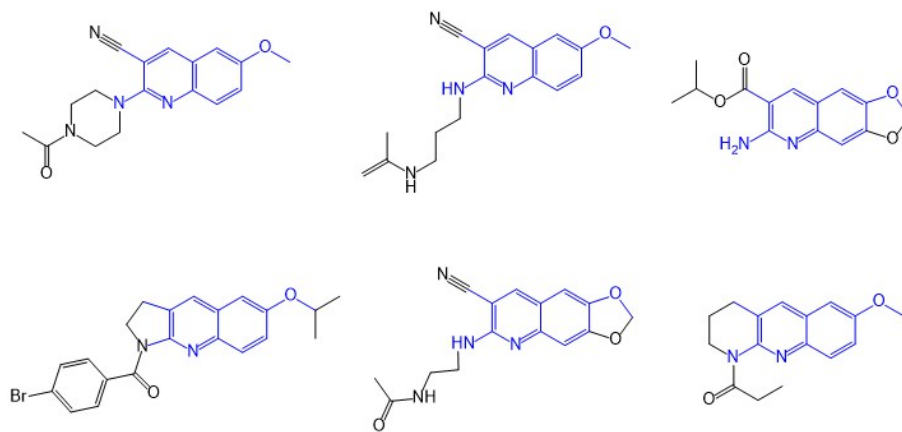


Figure 2.4: Example of a maximum common subgraph for a group of molecules.

2.3.2 GRAPH EDIT DISTANCE

Graph edit distance (GED)^{145,60,157,31} is the base of error-tolerant graph matching and the most commonly used inexact method to measure the pairwise similarity between graphs. The GED is considered one of the most flexible and versatile graph matching models available, useful for many applications in structural pattern analysis, pattern recognition, and related fields.

The GED defines the dissimilarity between two attributed graphs as the cost of the least expensive sequence of *edit operations* needed to transform one graph into another. Each *edit operation* can be one of the following six: insertion, deletion, and substitution of both nodes and edges in the graph, as shown in Figure 2.5.

The GED may be seen as the minimal amount of *edit operations* required to transform one graph into another. These operations are restricted to substitutions, insertions, and removals; and are applied either on nodes or edges. The complete sequence of *edit operations* needed to transform one graph into another is called *edit path*. There might be several possible *edit*

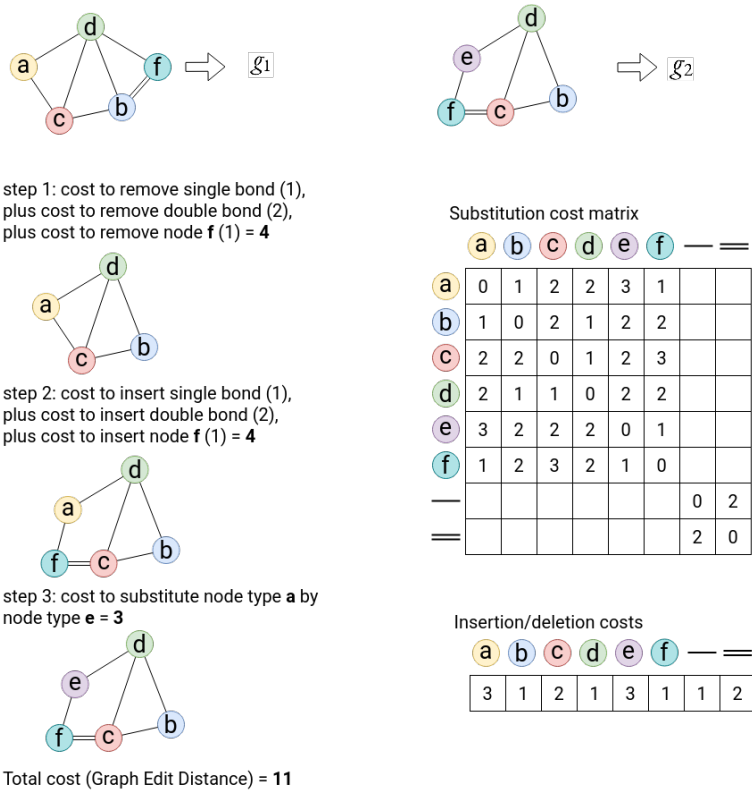


Figure 2.5: Simplified GED process to convert g_1 into g_2 . The total edit distance is 11 based on the substitution and insertion/deletion costs presented on the right side of the image.

	Clean bathroom	Sweep floors	Wash windows
Paul	\$2	\$3	\$3
Dave	\$3	\$2	\$3
Chris	\$3	\$3	\$2

Figure 2.6: Matrix of costs and workers. When applied to the above table, a pairwise assignment solver like the Kuhn–Munkres algorithm would yield the minimum cost of \$6, assigning Paul to clean the bathroom, Dave to sweep the floors, and Chris to wash the windows.

paths with the same result, mainly when working with attributed graphs. To identify the *edit path* holding the minimum global cost, we add up all the *edit operations* making up each *edit path*.

For a set of *edit paths* $\mathcal{P}(g_1, g_2)$ transforming graph g_1 into g_2 , where each *edit operation* e is penalized by a non-negative cost $c(e)$, the GED between graphs g_1 and g_2 is defined as:

$$GED(g_1, g_2) = \min_{(e_1, \dots, e_k) \in \mathcal{P}(g_1, g_2)} \sum_{i=1}^k c(e_i)$$

We approach this problem of finding the *edit path* yielding the minimal global cost as the *assignment problem*, which attempts to find an optimal pairwise assignment of the elements in two different sets having the same number of elements. For instance, for two sets $|A|$ and $|B|$ with the same cardinality n , we can consider an $n \times n$ cost matrix C where each element $C_{i,j}$ represents the cost of assigning the i -th element of $|A|$ to the j -th element of $|B|$ as the example depicted in Figure 2.6 considering the problem of pairwise assigning tasks to workers.

In this example, the optimal assignment of tasks and workers is the one that minimizes the global cost; in other words, finding the assignment that minimizes $\sum_{i=1}^n C_{i,p(i)}$. The Kuhn–Munkres algorithm (also known as the Hungarian algorithm)^{119,93} and its extension to work with non-square matrices²⁴ solve the assignment problem in $O(n^3)$ ⁵².

The Bipartite algorithm (BP)^{137,136} is an efficient method to compute the approximation of the GED faster than traditional methods, while the approximated distance accuracy is not much affected. BP's basic idea is to reduce the difficulty of a *quadratic assignment problem* in the GED computation to a *linear sum assignment problem*^{160,32}. BP algorithm is made up of three main steps: the first step defines a cost matrix based on the *edit operations* and transformation costs; the second step applies a lineal solver such as the Kuhn–Munkres algorithm to the cost matrix and deduces the pairwise assignation; the third step uses the assignation matching to compute the edit distance.

Figure 2.7 depicts a $(n + m) \times (n + m)$ BP cost matrix C for two given attributed graphs: G^p and G^q , where n and m indicate the graph orders. In this matrix, values $C_{a,i}$ in quadrant $Q1$ represent the substitution cost between clique K_a^p and K_i^q . Values $C_{a,\varepsilon}$ in quadrant $Q2$ represent the cost of deleting clique K_a^p ; values $C_{\varepsilon,i}$ in quadrant $Q3$ represent the cost of inserting clique K_i^q , and quadrant $Q4$ is filled with zeros since the cost of mapping two null cliques is always defined to be zero. A clique K_a on an attributed graph G is a local star-like structure concerning a central node and its adjacent outgoing edges.

In this context, Munkres' algorithm optimally solves the assignation problem in polynomial time $O((n + m)^3)$. Still, it is worth pointing out that the Graph Edit Distance computation keeps being sub-optimal since cliques are evaluated individually. Therefore the second-order information is incomplete compared with an optimal method in which the computational cost is exponential.

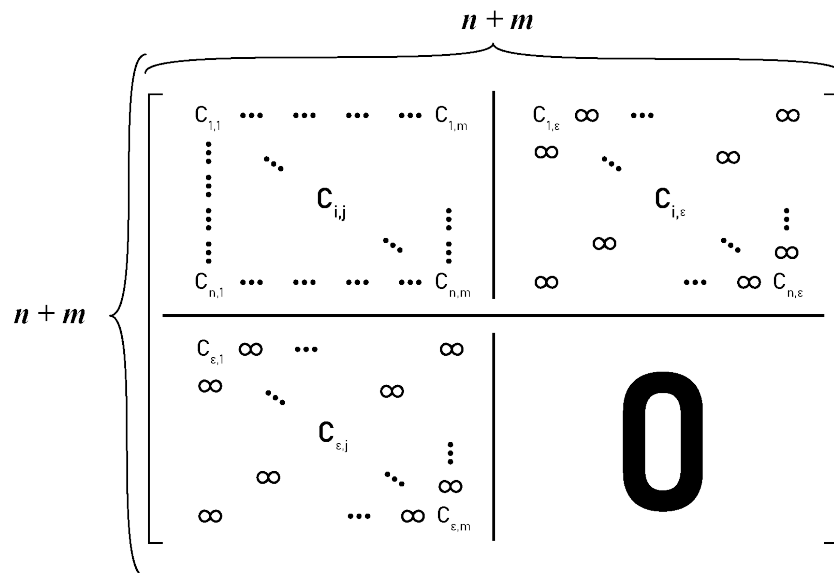


Figure 2.7: BP algorithm's cost matrix.

2.3.3 GED COMPUTATION

Developing an efficient similarity or dissimilarity measure for graphs is a major problem in structural pattern recognition. In the last three decades, several GED computation methods have been proposed. There are two types: those that return the exact value of the GED in exponential time concerning the number of nodes²⁰ and those that return an approximation of the GED in polynomial time^{150,146}. These GED computation methods have been subject to studies and comparisons in previous works^{42,172}. We used an in-house implementation of the bipartite graph matching method proposed by Serratos¹⁵⁰ to compute an approximation of the GED in polynomial time; it was developed in C++ and Python languages.

The GED definition has two main constraints: first, its computational expense grows exponentially with the number of nodes held by the graphs being compared; second, it relies

on a proper definition of the edit costs associated with the edit operations.

Independently of the definition of the edit costs, to define the GED as a distance metric, it must fulfill the following four properties³¹:

1. $C_{vs} \leq C_{vd} + C_{vi}$ and $C_{es} \leq C_{ed} + C_{ei}$
2. $C_{vd} = C_{vi}$ and $C_{ed} = C_{ei}$
3. $C_{vs} = 0$ and $C_{es} = 0$ if same attributes
4. All costs have to be non-negative

where C_{vs} is the cost of node substitution, C_{vd} is the cost of node deletion, C_{vi} is the cost of node insertion. For edges: C_{es} is the cost of edge substitution, C_{ed} is the cost of edge deletion, and C_{ei} is the cost of edge insertion.

Additionally, for the GED to be a metric, it also needs to satisfy the following requirements¹⁵¹:

- $GED(G_1, G_2) \leq GED(G_1, G_3) + GED(G_2, G_3)$
- $GED(G_1, G_2) = GED(G_2, G_1)$
- $GED(G_1, G_2) = 0 \Leftrightarrow G_1 = G_2$

Figure 2.8 shows a toy illustration of several GED examples using simple transformation costs (all equal to one). We start from two equal graphs, and then we make subtle progressive changes to one of the graphs (remove, substitute, or add nodes and edges) and re-compute the edit distance to exemplify the usage of the GED as a distance metric.

Now, let us discuss the edit costs employed in this work for the GED computation.

2.3.4 GED COSTS

Edit costs are often selected depending on how similar the nodes and edges are. For instance, when ErGs are compared, it is logical to think that the cost of substituting a “hydrogen-

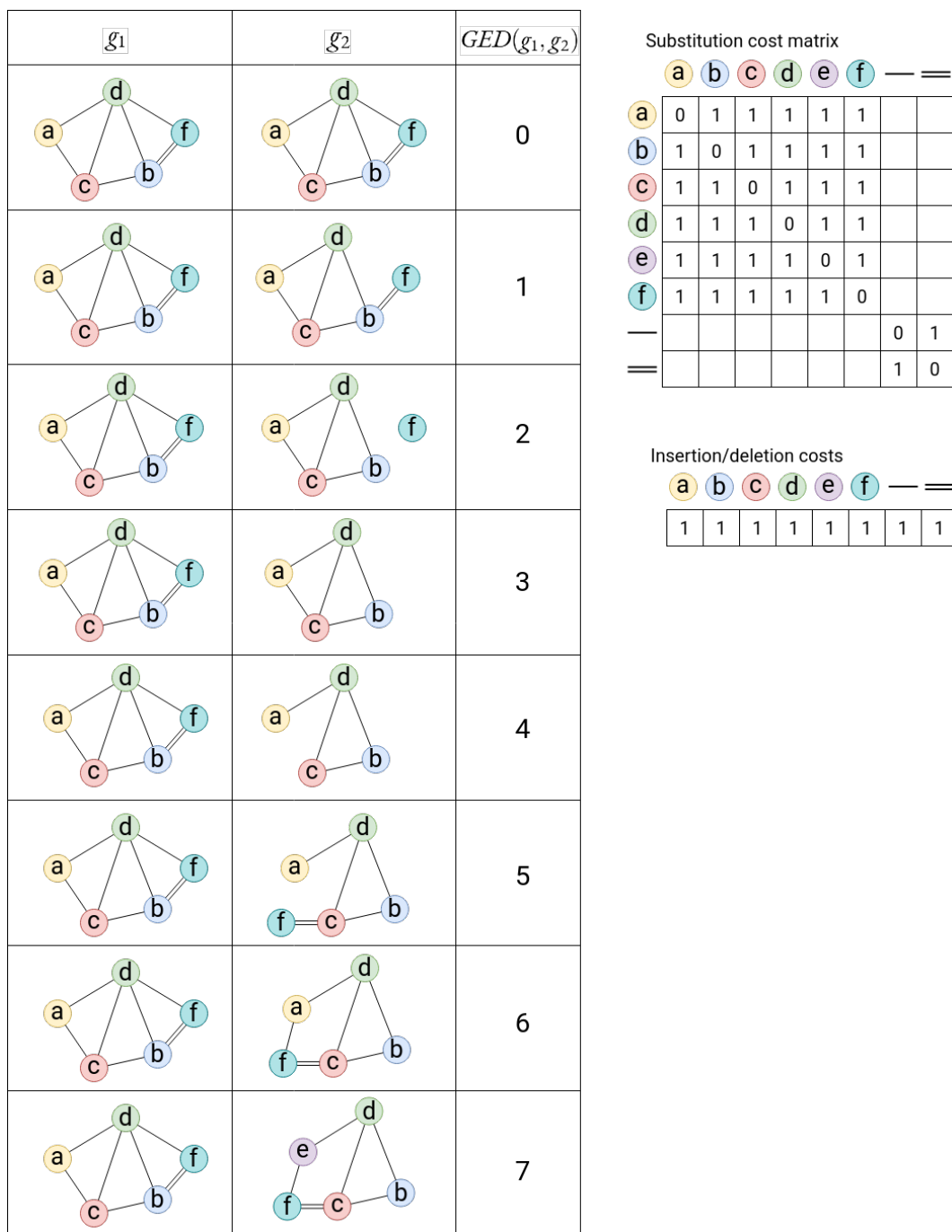


Figure 2.8: Illustration of the GED used as a distance metric.

Node attributes	
Attribute	Description
[0]	hydrogen-bond donor
[1]	hydrogen-bond acceptor
[2]	positive charge
[3]	negative charge
[4]	hydrophobic group
[5]	aromatic ring system
[6]	carbon link node
[7]	non-carbon link node
[0, 1]	hydrogen-bond donor + hydrogen-bond acceptor
[0, 2]	hydrogen-bond donor + positive charge
[0, 3]	hydrogen-bond donor + negative charge
[1, 2]	hydrogen-bond acceptor + positive charge
[1, 3]	hydrogen-bond acceptor + negative charge
[2, 3]	positive charge + negative charge
[0, 1, 2]	hydrogen-bond donor + hydrogen-bond acceptor + positive charge
Edge attributes	
Attribute	Description
-	single bond
=	double bond
≡	triple bond

Table 2.2: Description of the node and edge attributes that make up an ErG.

bond donor” feature with a joint “hydrogen-bond donor-acceptor” feature should be relatively lowly penalized. On the other hand, the cost of substituting a “hydrogen-bond donor” feature with an “aromatic ring” system should be penalized with a higher value. Similarly, inserting a single bond should have a lower penalization cost than inserting a double bond, and so on.

In chapters 3 and 4, we use the edit costs proposed by Harper *et al.*⁸⁰ adapted to ErG. The node and edge descriptions are found in Table 2.2, and our specific costs can be shown in Table 2.3.

Matrix of Substitution Costs																		
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[0, 1]	[0, 2]	[0, 3]	[1, 2]	[1, 3]	[2, 3]	[0, 1, 2]	-	=	≡
[0]	0	2	2	2	2	2	2	3	1	1	1	2	2	2	1	2	3	3
[1]	2	0	2	2	2	2	2	3	1	2	2	1	1	2	1	2	3	3
[2]	2	2	0	2	2	2	2	3	2	1	2	1	2	1	1	2	3	3
[3]	2	2	2	0	2	2	2	3	2	2	1	2	1	1	2	2	3	3
[4]	2	2	2	2	0	2	2	3	2	2	2	2	2	2	2	2	3	3
[5]	2	2	2	2	2	0	2	3	2	2	2	2	2	2	2	2	3	3
[6]	2	2	2	2	2	2	0	3	2	2	2	2	2	2	2	2	3	3
[7]	3	3	3	3	3	3	3	0	3	3	3	3	3	3	3	3	3	3
[0, 1]	1	1	2	2	2	2	2	3	0	2	2	2	2	2	2	2	3	3
[0, 2]	1	2	1	2	2	2	2	3	2	0	2	2	2	2	2	2	3	3
[0, 3]	1	2	2	1	2	2	2	3	2	2	0	2	2	2	2	2	3	3
[1, 2]	2	1	1	2	2	2	2	3	2	2	2	0	2	2	2	2	3	3
[1, 3]	2	1	2	1	2	2	2	3	2	2	2	2	0	2	2	2	3	3
[2, 3]	2	2	1	1	2	2	2	3	2	2	2	2	2	0	2	2	3	3
[0, 1, 2]	1	1	1	2	2	2	2	3	2	2	2	2	2	2	0	2	3	3
-	2	2	2	2	2	2	2	3	2	2	2	2	2	2	2	0	3	3
=	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	0	3
≡	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	0
Insertion/Deletion Costs																		
insert	1	1	1	1	1	1	0.5	0.5	1	1	1	1	1	1	1	0.1	1	1
delete	1	1	1	1	1	1	0.5	0.5	1	1	1	1	1	1	1	0.1	1	1

Table 2.3: Substitution and Insertion/Deletion costs used in the GED and SED calculation.

2.3.5 GED COMPUTATIONAL EXPENSE

When representing data by attributed graphs, the main challenge is the computational difficulty to compare them, which in many cases restricts the GED applicability to only graphs of a relatively small size. Any optimal implementation may become exponential with the number of nodes, at least in the worst case. One example of this is the $A\Box$ algorithm⁸¹, a classical tree search algorithm, which is the most popular method to compute the GED optimally. A^* works by exploring the space of all possible mappings among nodes and edges in both graphs. On the other hand, we have sub-optimal implementations to solve the graph-matching problem; they have polynomial complexity but do not guarantee to find the optimal solution^{14,152}. Several such implementations and surveys have surged^{145,54,42,171,172,56,60}. Sub-optimal techniques are meant to reduce the search space and alleviate the computational

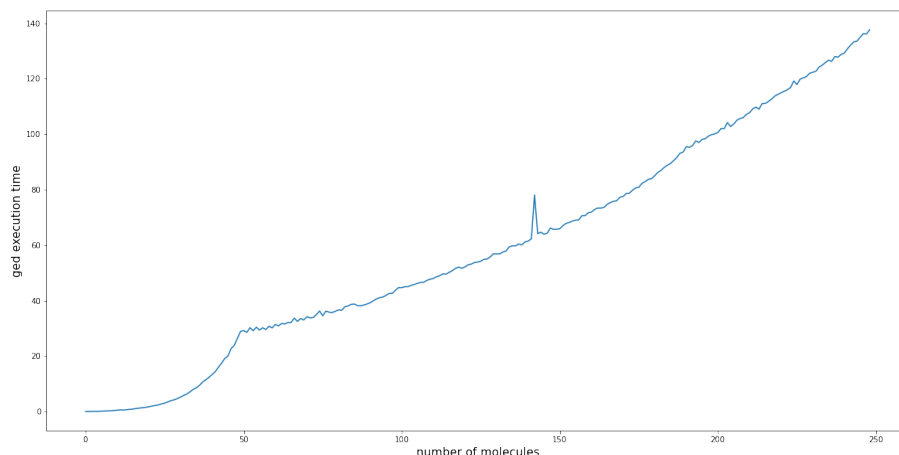


Figure 2.9: GED execution time with an increasing number of ErGs.

expense problem.

As a measure to reduce the computational burden for graph comparison, in this work, we use a sub-optimal GED implementation based on the Bipartite Graph Edit Distance. As mentioned before, this sub-optimal implementation has the primary goal of reducing the difficulty of a *quadratic assignment problem* in the GED to that of a *linear sum assignment problem* in the BP, since the difficulty of the latter is the same as the problem of finding an optimal assignment for two sets of items. Another measure taken in this work to reduce the computational burden is to use reduced graphs, which by definition contain fewer nodes and edges than the original chemical graphs from the molecule.

Figure 2.9 shows different GED execution times while comparing an increasing number of randomly selected ErGs obtained from the publicly available dataset AIDS (see section 5.2.1 for more details about the dataset).

2.4 METHOD EVALUATION

Throughout this work, we carry out two different types of evaluation methods: the first one, used in chapter 3, uses the tools and metrics available in the LBVS benchmarking platform and the RDKit; This evaluation computes the AUC performance value and the BEDROC as an “early recognition” method, the usage of both metrics combined is recommended as good practice by Riniker and Landrum in their fingerprint benchmarking study¹³⁹. The second evaluation method, used in chapters 4 and 5, is based on bioactivity prediction accuracy. More precisely, the number of errors or miss-classifications when trying to predict the bioactivity of molecules with a particular protein target; We use the same metric during the training process of a genetic algorithm used to fit a model over 127 targets in 4.

To predict compounds’ bioactivity, we handle the AGs in the context of nearest-neighbors classification. We decide the belonging group of an unknown input AG by comparing it with many ground-truth known AGs in the database; Later on, the unknown input is assigned to the same class or group as the most similar ground-truth.

All the techniques presented in this chapter will serve as the foundations upon which later chapters will build; nevertheless, we might expand some of the concepts later in those chapters.

*I have had my results for a long time: but I do not yet know
how I am to arrive at them.*

Carl Friedrich Gauss.

3

Ligand-Based Virtual Screening Using Graph Edit Distance as Molecular Similarity Measure

3.1 CHAPTER INTRODUCTION

This chapter investigates the effectiveness of a graph-only driven molecular comparison using extended reduced graphs and graph edit distance methods for molecular similarity calculation as a tool for ligand-based virtual screening applications. As mentioned before, LBVS applications estimate a molecule's bioactivity based on that of similar compounds.

The results proved to be very stable. The graph edit distance method performed better than other methods previously used on reduced graphs; this is exemplified using six publicly available datasets. The screening and statistical tools available on the ligand-based virtual screening benchmarking platform and the RDKit were also used. Our method performed better in most of the experiments than other molecular similarity methods using array representations.

Overall, it is shown that extended reduced graphs and graph edit distance are a suitable combination of methods, having numerous applications and identifying bioactivity similarities in a structurally diverse group of molecules.

This chapter is organized as follows. First, materials and methods are presented and explained in detail. Second, computational results are shown. In the end, a final discussion concludes the chapter.

3.2 SPECIFIC MATERIALS AND METHODS

3.2.1 DATASETS

The datasets in this chapter were presented in section 2.1; they are publicly available and normalized in a ready-to-use format inside the LBVS benchmarking platform; Those datasets

are DUD-E, MUV, GLL&GDD, CAPST, NRLiSt BDB, and ULS-UDS.

3.2.2 MOLECULAR REPRESENTATION AND COMPARISON.

Three different molecular representation methods are employed in this chapter: FP-based, SED-based, and GED-based methods; they were presented in detail in sections 1.5.1, 1.5.2, and 1.5.3, respectively.

SED-based and GED-based methods share the exact edition costs needed to balance the transformation from one node to another and one edge to another.

3.2.3 METHOD EVALUATION

The LBVS benchmarking platform and the RDKit library were used to evaluate the three methods mentioned earlier: Fingerprint-based, SED-based, and GED-based. The screening phase consists of using all active and inactive compounds in the test set compared with the active compounds in the training set. Then, all test molecules are ranked according to their similarity to the active molecules in the training set. Only the information from the test molecule with the highest similarity value is kept. Subsequently, the performance of each method is calculated by using the information obtained in the previous step and some of the performance evaluation methods.

As recommended by Riniker and Landrum in their fingerprint benchmarking study¹³⁹, it is a good practice to provide the performance values for AUC and one of the “early recognition” methods, enrichment factor (EF) or Boltzmann-enhanced discrimination of ROC (BEDROC)¹⁷⁰. In this study, we selected the BEDROC ($\alpha = 20$) as our “early recognition” method since it has the advantage of running from 0 to 1.

The step-by-step evaluation processed followed by the LBVS benchmarking platform is as follows:

1. The library generates ten random splits of molecules from the databases to limit the size and work with smaller subsets. Having several subsets helps to mitigate the randomness used to select the molecules on each split and better generalize the different molecular features available. The following steps are performed equally on each split.
2. The split is divided into two parts, train and test sets.
3. For each molecule in the test set (also known as the “query molecule”), the library predicts its activity by comparing it with all the molecules in the train set and finding the most similar one. The “query molecule’s” activity is assumed to be the same as that of the most similar molecule from the train set.
4. After having an activity prediction for each molecule in the test set, a ROC curve is generated employing the true positive rate (TPR) against the false positive rate (FPR). We use a threshold through the accepted similarity values (similarity between the “query molecule” and the most similar one) to determine whether the prediction was accurate or not. In other words, even if the prediction is performed according to the most similar molecule, maybe the similarity value is not good enough; therefore, we check TPR and FPR while changing the threshold for the accepted values, see Figure 3.1.

While varying the threshold (vertical line), different TP and FP values are produced, as illustrated in Figure 3.2.
5. Once the ROC curve is created, the library computes the area under the curve or

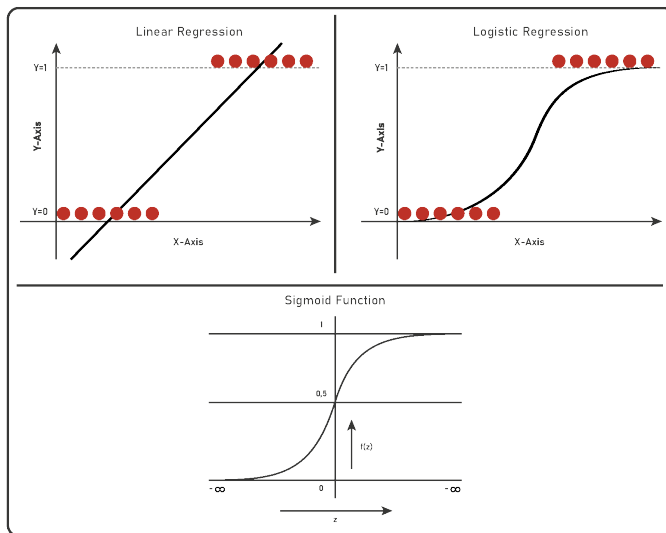


Figure 3.1: Examples of activation functions normalizing a method's output to a probability distribution over predicted output classes. Logistic regression is the type of regression analysis often used when the dependent variable is binary.

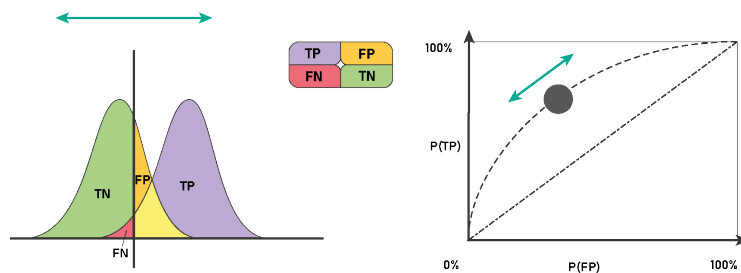


Figure 3.2: Depiction of a ROC curve while changing the distribution of TP and FP values. TP and FP distribution changes by modifying the threshold.

Predicted activities: [1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, ...]
AUC: 0.921295

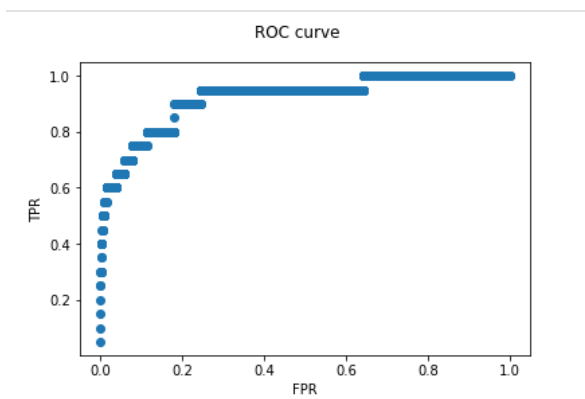


Figure 3.3: Example of an array holding the predicted activities from one experiment. ROC curve and its corresponding AUC value are obtained employing the predictions.

(AUC) and uses this value to compare the performance between different ROC curves.

Figure 3.3 shows an example of a ROC curve and its corresponding AUC values resulting from one of the experiments below.

3.2.4 COMPUTATIONAL EXPENSE

The experiments reported in this chapter were performed using a quad-core, 2.0 GHz Intel Core i7 laptop with 8 GB RAM (operating system version Ubuntu 16.04). The most time-consuming step during the benchmarking process is calculating the similarity between graphs for each method; the remaining steps require negligible computational expense. An accurate time-consuming comparison would not be fair since different methods were developed using different proportions of Python or C++ programming language. Nevertheless, none of the similarity calculations took longer than 20 minutes per target (executing ten iterations per target where each iteration performs almost 20 thousand individual molecular comparisons) using the predefined random-built splits available in the LBVS benchmarking platform.

3.3 RESULTS

3.3.1 EXPERIMENTAL SETUP

Six experiments were carried out using different activity classes from six publicly available datasets. The classification accuracy for the methods tested was computed using the screening and statistical tools available in the LBVS benchmarking platform and the RDKit. The AUC and BEDROC results show the behavior of all the targets available in each data set in the platform using the Fingerprint-based, SED-based, and GED-based methods.

3.3.2 ANALYSIS OF THE EXPERIMENTS

Table 3.1 summarizes the input data of the experiments, and Figure 3.4 shows the overall behavior for the three similarity methods for all the targets available in the LBVS platform. These results show that the GED-based method has a slight advantage over the FP-based and the SED-based methods, which can be observed in the median, first quartile, and the third quartile for both the AUC and BEDROC results in the box-and-whisker plots. An overall comparison might not be very informative, so a more profound analysis should be carried out for each dataset separately.

Figures 3.5 and 3.6 show the same information as Figure 3.4 separated for each dataset (one dataset per sub-figure). Figure 3.5 represents the AUC values, and Figure 3.6 represents the BEDROC values. Again, box-and-whisker plots are located on the right of each subplot to illustrate the distribution. The following analysis will focus on these distribution results.

For the ULS-UDS dataset, the median value of results for the GED-based method is better than for the FP-based and SED-based methods. This difference is noticeable in the AUC and

Dataset	Targets used
ULS-UDS	5HT _{1F} _Agonist, MTR _{1B} _Agonist, OPRM_Agonist, PE _{2R3} _Antagonist
GLL&GDD	5HT _{1A} _Agonist, 5HT _{1A} _Antagonist, 5HT _{1D} _Agonist, 5HT _{1D} _Antagonist, 5HT _{1F} _Agonist, 5HT _{2A} _Antagonist, 5HT _{2B} _Antagonist, 5HT _{2C} _Agonist, 5HT _{2C} _Antagonist, 5HT _{4R} _Agonist, 5HT _{4R} _Antagonist, AA _{1R} _Agonist, AA _{1R} _Antagonist, AA _{2AR} _Antagonist, AA _{2BR} _Antagonist, ACM ₁ _Agonist, ACM ₂ _Antagonist, ACM ₃ _Antagonist, ADA _{1A} _Antagonist, ADA _{1B} _Antagonist, ADA _{1D} _Antagonist, ADA _{2A} _Agonist, ADA _{2A} _Antagonist, ADA _{2B} _Agonist, ADA _{2B} _Antagonist, ADA _{2C} _Agonist, ADA _{2C} _Antagonist, ADRB ₁ _Agonist, ADRB ₁ _Antagonist, ADRB ₂ _Agonist, ADRB ₂ _Antagonist, ADRB ₃ _Agonist, ADRB ₃ _Antagonist, AG _{2R} _Antagonist, BKRB ₁ _Antagonist, BKRB ₂ _Antagonist, CCKAR_Antagonist, CLTR ₁ _Antagonist, DRD ₁ _Antagonist, DRD ₂ _Agonist, DRD ₂ _Antagonist, DRD ₃ _Antagonist, DRD ₄ _Antagonist, EDNRA_Antagonist, EDNRB_Antagonist, GASR_Antagonist, HRH ₂ _Antagonist, HRH ₃ _Antagonist, LSHR_Antagonist, LT _{4R1} _Antagonist, LT _{4R2} _Antagonist, MTR _{1A} _Agonist, MTR _{1B} _Agonist, MTR _{1L} _Agonist, NK _{1R} _Antagonist, NK _{2R} _Antagonist, NK _{3R} _Antagonist, OPRD_Agonist, OPRK_Agonist, OPRM_Agonist, OXYR_Antagonist, PE _{2R1} _Antagonist, PE _{2R2} _Antagonist, PE _{2R3} _Antagonist, PE _{2R4} _Antagonist, TA _{2R} _Antagonist, V _{1AR} _Antagonist, V _{1BR} _Antagonist, V _{2R} _Antagonist
CAPST	CDK ₂ , CHK ₁ , PTP _{1B} , UROKINASE
DUD-E	COX ₂ , DHFR, EGFR, FGFR ₁ , FXA, P ₃₈ , PDGFRB, SRC, AA _{2AR}
NRLiSt_BDB	AR_Agonist, AR_Antagonist, ER_Alpha_Agonist, ER_Alpha_Antagonist, ER_Beta_Agonist, FXR_Alpha_Agonist, GR_Agonist, GR_Antagonist, LXR_Alpha_Agonist, LXR_Beta_Agonist, MR_Antagonist, PPAR_Alpha_Agonist, PPAR_Beta_Agonist, PPAR_Gamma_Agonist, PR_Agonist, PR_Antagonist, PXR_Agonist, RAR_Alpha_Agonist, RAR_Beta_Agonist, RAR_Gamma_Agonist, RXR_Alpha_Agonist, RXR_Alpha_Antagonist, RXR_Gamma_Agonist, VDR_Agonist
MUV	466, 548, 600, 644, 652, 689, 692, 712, 713, 733, 737, 810, 832, 846, 852, 858, 859

Table 3.1: Input data used for the experiments. The column entitled 'Dataset' contains the name of each dataset, and the column entitled 'Targets used' contains the name of the targets used during the experiments for each dataset. Note that per-target points in the result plots shown below are arranged in the same order as in this table.

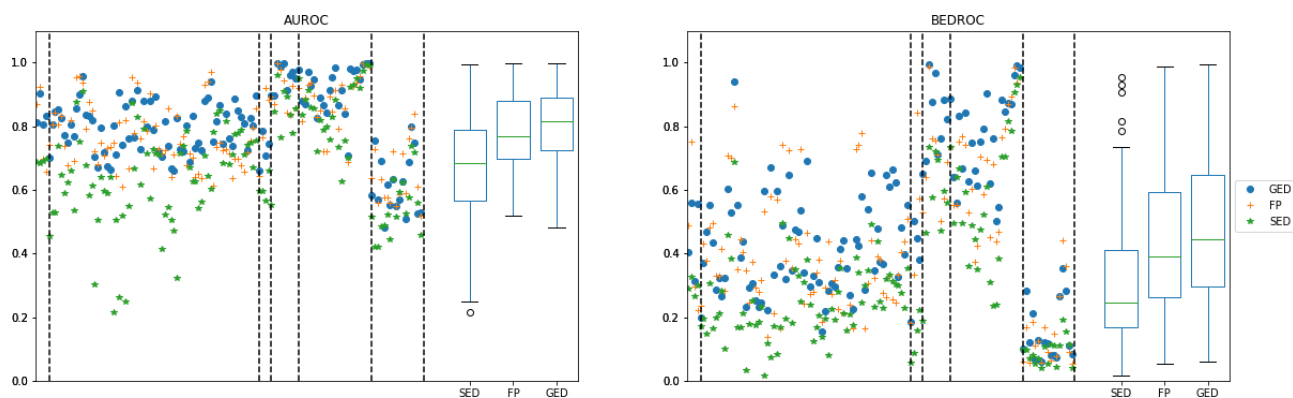


Figure 3.4: AUC and BEDROC ($\alpha = 20$) for all available targets in the LBVS benchmarking platform. The scattered values on the left of both subplots represent the median value from 10 predefined random-built splits, using different colors and shapes per similarity method. Vertical segmented lines mark the edge between different datasets (from left to right: ULS-UDS, GLL&GDD, CAPST, DUD-E, NRISt_BDB, MUV). The box-and-whisker plots on the right of both subplots show the distribution of the resulting values for each similarity method. The boxes show the first and third quartile, the line is the median value (second quartile), and the whiskers extend from the boxes to show the range of the data (outliers are included if there are any).

BEDROC results, especially in the last two targets (OPRM_Agonist and PE₂R₃_Antagonist). Probably the most noticeable advantage of the GED-based method, in this case, is its stability. The stability is represented as the box and whiskers length. Shorter lengths indicate that several results are closer to each other, and therefore the method seems more reliable.

For the GLL&GDD dataset, the performance of the GED-based method is significantly better than the FP-based and SED-based methods. The median, first quartile, and third quartile are better in both the AUC and BEDROC results. The most significant difference in performance is in the “PE₂R” group targets, located close to the right in the plot (PE₂R₁_Antagonist, PE₂R₂_Antagonist, PE₂R₃_Antagonist, and PE₂R₄_Antagonist).

For the CAPST dataset, the performance is slightly better with the FP-based method rather than the GED-based and SED-based methods. This difference in performance can be seen in the AUC and BEDROC results, and it is more noticeable in the last two targets (PTP₁B

and UROKINASE). Nevertheless, the stability of results is not very reliable, especially in BEDROC, where performance goes from very low to very high values.

The advantage of the GED-based method is possibly most significant for the DUD-E dataset. Even the second quartile of the GED-based method is higher than the third quartile of the FP-based and SED-based methods. This superiority is valid for the AUC and BEDROC values. Furthermore, the GED-based method is relatively stable, particularly for the AUC values. The GED-based advantage is more noticeable in such targets as P₃₈, SRC, FXA, and FGFR₁ located between the center and the right of the plot.

With the NRLiSt_BDB dataset, the GED-based method again gets significantly better results than FP-based and SED-based methods. This difference can be seen in the AUC and BEDROC results, and it is evident in the first, second (median), and third quartiles. The stability is similar for all methods, with the FP-based method being slightly better in the AUC values and the SED-based method being slightly better in the BEDROC values. The most significant difference in performance for the GED-based method is for LXR_Alpha_Agonist, LXR_Beta_Agonist, PPAR_Alpha_Agonist, and PPAR_Gamma_Agonist, near the center of the plot.

The overall results for the MUV dataset are the lowest of all the experiments. This difference is present in the three similarity methods. Nevertheless, the GED-based method performs slightly better than the FP-based and the SED-based methods in the “early recognition” BEDROC results. The difference is more significant for such targets as 466, 548, and 600 on the left of the plot. Moreover, the FP-based method performs slightly better than others in the AUC results, particularly in such targets as 652, 852, and 737.

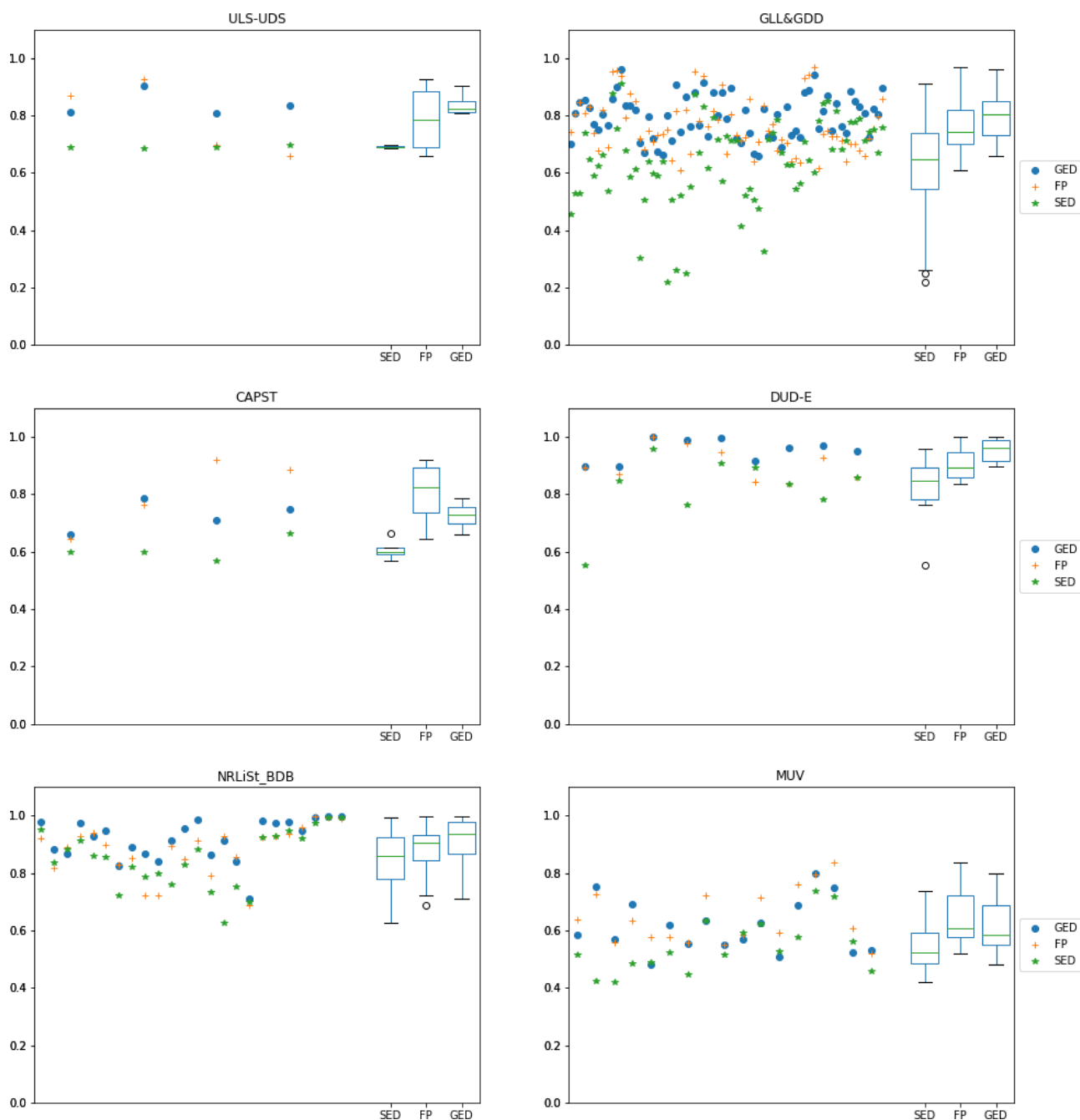


Figure 3.5: AUC results for all available targets in the LBVS benchmarking platform separated by dataset. Each scattered value on the left of each subplot represents the median value of 10 predefined random-built splits. Different colors and shapes are used for each similarity method. Box-and-whisker plots on the right of each subplot show the distribution of results.

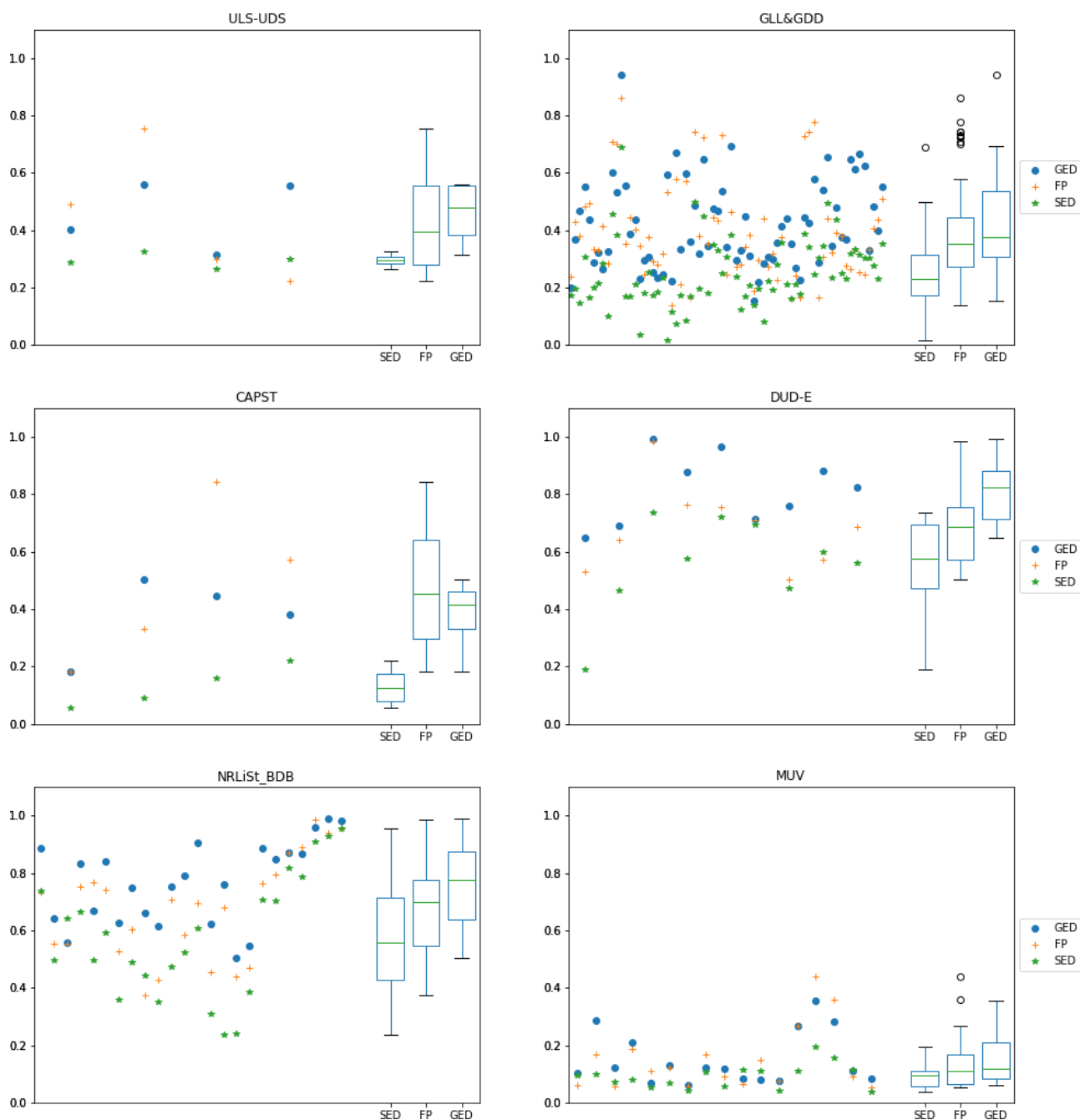


Figure 3.6: BEDROC ($\alpha = 20$) results for all available targets in the LBVS benchmarking platform separated by dataset. Each scattered value on the left of each subplot represents the median value of 10 predefined random-built splits. Different colors and shapes are used for each similarity method. Box-and-whisker plots on the right of each subplot show the distribution of results.

	Friedman test (AUC)	Friedman test (BEDROC)
ULS-UDS	0.173774	0.173774
GLL&GDD	2.97804E-14*	5.30798E-15*
DUD-E	0.000911882*	0.000300185*
NRLiSt_BDB	7.48518E-05*	5.77775E-08*
MUV	0.00102573*	0.00714619*
CAPST	0.0497871*	0.0497871*
All datasets	7.46387E-24*	1.89219E-28*

Table 3.2: P-values for the Friedman test (AUC and BEDROC) comparing the three similarity methods (GED-based, FP-based, and SED-based) simultaneously. The test is done per dataset, and all datasets are combined in the last row. Here, a confidence level of $\alpha = 0.05$ is used, so p-values lower than α indicate statistically significant differences, which are marked with an asterisk (*) in the table.

3.3.3 STATISTICAL TESTS

Statistical tests were carried out to determine whether the differences in performance are statistically significant. In other words, these tests assess whether some methods are consistently better than others.

First, we used a comprehensive Friedman test⁵⁸ for each dataset, as presented in Table 3.2. The last row shows the results of applying the same test to all the targets combined. The p-value for the comprehensive Friedman test was extremely low in most cases (a confidence level of $\alpha = 0.05$ is used), indicating statistically significant differences between different methods. Hence, a more in-depth analysis might be helpful.

The second step in the statistical tests consists of a pairwise Wilcoxon signed-rank test¹⁷⁹ to determine which pairs of methods exhibit a statistically significant difference. Table 3.3 shows the results of this test applied to all the datasets combined and including the AUC and BEDROC values.

The same pairwise Wilcoxon signed-rank test was performed again but this time applied to each dataset separately. The results of the tests are presented in Tables 3.4 (AUC values)

Wilcoxon test (AUC)				Wilcoxon test (BEDROC)			
	SED	FP	GED	SED	FP	GED	
SED		1.20841E-13*	7.4432E-21*		2.08456E-16*	1.18024E-21*	
FP	1.20841E-13*		0.000748788*	2.08456E-16*		0.000585085*	
GED	7.4432E-21*	0.000748788*		1.18024E-21*	0.000585085*		

Table 3.3: P-values for the pairwise Wilcoxon signed-rank test (AUC and BEDROC) comparing the three similarity methods (GED-based, FP-based, and SED-based). The test is applied to all targets in the datasets combined. Here, a confidence level of $\alpha = 0.05$ is used, so p-values lower than α indicate statistically significant differences, which are marked with an asterisk (*) in the table.

ULS-UDS				GLL&GDD			
	SED	FP	GED	SED	FP	GED	
SED		0.273322	0.0678892		4.26037E-08*	3.34327E-12*	
FP	0.273322		0.465209	4.26037E-08*		0.0018207*	
GED	0.0678892	0.465209		3.34327E-12*	0.0018207*		

CAPST				MUV			
	SED	FP	GED	SED	FP	GED	
SED		0.0678892	0.0678892		0.000351533*	0.00748178*	
FP	0.0678892		0.465209	0.000351533*		0.0615039	
GED	0.0678892	0.465209		0.00748178*	0.0615039		

DUD-E				NRLiSt_BDB			
	SED	FP	GED	SED	FP	GED	
SED		0.109745	0.00768579*		0.0520334	3.02696E-05*	
FP	0.109745		0.00768579*	0.0520334		0.00167307*	
GED	0.00768579*	0.00768579*		3.02696E-05*	0.00167307*		

Table 3.4: P-values for the pairwise Wilcoxon signed-rank test (AUC) comparing the three similarity methods (GED-based, FP-based, and SED-based). The test is done using all targets separated by datasets. Here, a confidence level of $\alpha = 0.05$ is used, so p-values lower than α indicate statistically significant differences, which are marked with an asterisk (*) in the table.

and 3.5 (BEDROC values).

Pairwise comparison tables show very low p-values in most cases (the lower the p-value, the better), and the difference between one method and the other was statistically significant. This difference is valid for AUC and BEDROC results and with all datasets except ULS-UDS and CAPST. Therefore, for these two datasets, the slight differences in performance are not regarded as statistically significant. Note that statistically significant differences may not always mean practically meaningful differences⁸⁷.

Finally, Table 3.6 shows the drawing and the ErG representation of three sample molecules. As an example, we selected the first two active molecules (ligands) and the first inactive molecule

ULS-UDS			
	SED	FP	GED
SED		0.273322	0.0678892
FP	0.273322		1
GED	0.0678892	1	

GLL&GDD			
	SED	FP	GED
SED		1.41571E-09*	7.73521E-13*
FP	1.41571E-09*		0.125128
GED	7.73521E-13*	0.125128	

CAPST			
	SED	FP	GED
SED		0.0678892	0.0678892
FP	0.0678892		0.465209
GED	0.0678892	0.465209	

MUV			
	SED	FP	GED
SED		0.00988213*	0.00359936*
FP	0.00988213*		0.758312
GED	0.00359936*	0.758312	

DUD-E			
	SED	FP	GED
SED		0.015156*	0.00768579*
FP	0.015156*		0.00768579*
GED	0.00768579*	0.00768579*	

NRLiSt_BDB			
	SED	FP	GED
SED		0.000318217*	3.43006E-05*
FP	0.000318217*		0.000284994*
GED	3.43006E-05*	0.000284994*	

Table 3.5: P-values for the pairwise Wilcoxon signed-rank test (BEDROC) comparing the three similarity methods (GED-based, FP-based, and SED-based). The test is done using all targets separated by datasets. Here, a confidence level of $\alpha = 0.05$ is used, so p-values lower than α indicate statistically significant differences, which are marked with an asterisk (*) in the table.

(decoy) from the target VDR_Agonist in the NRLiSt_BDB dataset. Table 3.7 shows the distances between these molecules using the FP-based, SED-based, and GED-based similarity methods. The range of FP-based and SED-based distances is $[0, 1]$, whereas the range of the GED-based is $[0, \text{Inf}]$. Hence, we cannot compare the values directly from different methods. Nevertheless, it is noticeable that, for each method, the distance value computed between two ligand compounds is lower than the distance computed between a ligand compound and a decoy compound. This assertion is the basis of correctly classifying the ligand and decoy molecules to be used in the process of virtual screening.

3.4 DISCUSSION

This chapter presented a molecular similarity measure that uses graph edit distance to compare the representation of molecules by extended reduced graphs effectively. This method works as an alternative to the fingerprint-based similarity method used in the original paper on ErGs by Stiefl *et al.* and an alternative to the string edit distance-based method used in the paper by Harper *et al.*

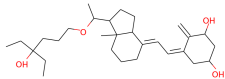
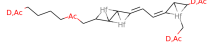
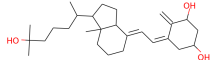
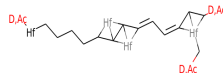
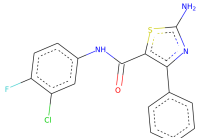
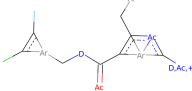
Mol ID	Molecule	ErG	Properties
1			Mol. Name: ZINCo4474609 Dataset: NRLiSt_BDB Target: VDR_Agonist Activity: Ligand
2			Mol. Name: ZINCo3924790 Dataset: NRLiSt_BDB Target: VDR_Agonist Activity: Ligand
3			Mol. Name: ZINCo0091842 Dataset: NRLiSt_BDB Target: VDR_Agonist Activity: Decoy

Table 3.6: Three sample molecules from the target VDR_Agonist in the NRLiSt_BDB dataset.

	Mol ID 2 Ligand	Mol ID 3 Decoy
Mol ID 1 Ligand	SED: 0.06 FPD: 0.40 GED: 0.49	SED: 0.28 FPD: 0.92 GED: 3.05

Table 3.7: Distances between molecules shown in Table 3.6 computed using the FP-based, SED-based, and GED-based similarity methods.

The experiments performed used several samples collected from publicly available datasets like DUD-E and MUV. To overcome the common problems of reproducibility when different methods are compared, we used a benchmarking platform proposed by Skoda and Hok-sza. The platform includes, among other features, various screening and statistical tools and provides fully reproducible outcomes.

Results show that the GED-based method performed better in 5 out of 6 methods according to the “early recognition” BEDROC values; moreover, GED-based performed better in 4 out of 6 methods according to AUC values. Nevertheless, these differences are statistically significant in four out of six datasets because ULS-UDS and CAPST differences are not significant according to the pairwise Wilcoxon signed-rank test.

To compute the GED, we used the edit costs proposed by Harper *et al.*, which experts assigned to manage relationships between the different node and edge types. The next chapter will be focused on automatically learning the edit costs on nodes and edges in several scenarios using various toxicological endpoints. This learning process will be similar to that carried out by Birchall *et al.*¹⁸, in which they optimized the edit values proposed by Harper *et al.* for their method.

We haven't got the money, so we've got to think.

Ernest Rutherford.

4

Learning The Edit Costs Of The Graph Edit Distance Applied to Ligand-Based Virtual Screening Applications

4.1 CHAPTER INTRODUCTION

In the previous chapter, we used the edit costs proposed by Harper *et al.*, assigned by experts considering the different node and edge types. This chapter presents a method for optimizing those edit costs based on minimizing the distance between molecules classified correctly and maximizing the distance between molecules classified incorrectly.

This chapter is inspired by a similar work carried out by Birchall *et al.*¹⁸. In that work, the authors optimize the transformation costs of a String Edit Distance-based method to compare molecules using reduced graphs. In contrast, our work optimizes the edit costs of a Graph Edit Distance-based method to compare molecules using ErG.

The outline of this chapter is as follows. First, materials and methods are presented and explained in detail, including the datasets, the GED methodology, and the optimization process; second, we present the computational results; third, the chapter is concluded with a final discussion.

4.2 SPECIFIC MATERIALS AND METHODS

4.2.1 DATASETS

Datasets used in this chapter are trimmed versions of the datasets presented in section 2.1; they were trimmed to reduce the computational time needed for the optimization process. The resulting subsets include the first 100 active and 100 inactive molecules for each target as formatted in the LBVS benchmarking platform. In some cases, available active molecules are less than 100; for those cases, all available active molecules and the same number of inactive molecules are used. Later, we split each set by half to have a *train* and a *test* subset used

independently; the former to optimize the transformation costs and the latter to evaluate the recognition ratio with unknown data.

4.2.2 MOLECULAR REPRESENTATION AND COMPARISON.

GED COMPUTATION

Several GED computational methods have been proposed during the last three decades, which can be classified into two groups: those returning the exact value for the GED in exponential time proportional to the number of nodes²⁰, and those returning an approximation of the GED in polynomial time^{150,146}. These two groups of GED computational methods have been widely studied^{42,172}.

As in the previous chapter, here, we compute the approximation of the GED in polynomial time using an in-house implementation of the bipartite graph matching method proposed by Serratos¹⁵⁰; it was programmed in C++ and Python languages.

GED COSTS

In the previous chapter, we used the edit costs proposed by Harper *et al.*⁸⁰ with minor changes to fit the ErG features. The node and edge descriptions are shown in Table 2.3, and the specific costs proposed by Harper *et al.* are exposed in Tables 4.1 and 4.2. Note that the insertion and deletion costs applied to a given node are constants. Moreover, substitutions are symmetric, which means that the substitution of node type A to B is assigned the exact cost as the substitution of type B to A to guarantee the symmetry property for the GED.

Substitution Costs for Nodes															
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[0, 1]	[0, 2]	[0, 3]	[1, 2]	[1, 3]	[2, 3]	[0, 1, 2]
[0]	0	2	2	2	2	2	2	3	1	1	1	2	2	2	1
[1]	2	0	2	2	2	2	2	3	1	2	2	1	1	2	1
[2]	2	2	0	2	2	2	2	3	2	1	2	1	2	1	1
[3]	2	2	2	0	2	2	2	3	2	2	1	2	1	1	2
[4]	2	2	2	2	0	2	2	3	2	2	2	2	2	2	2
[5]	2	2	2	2	2	0	2	3	2	2	2	2	2	2	2
[6]	2	2	2	2	2	2	0	3	2	2	2	2	2	2	2
[7]	3	3	3	3	3	3	3	0	3	3	3	3	3	3	3
[0, 1]	1	1	2	2	2	2	2	3	0	2	2	2	2	2	2
[0, 2]	1	2	1	2	2	2	2	3	2	0	2	2	2	2	2
[0, 3]	1	2	2	1	2	2	2	3	2	2	0	2	2	2	2
[1, 2]	2	1	1	2	2	2	2	3	2	2	2	0	2	2	2
[1, 3]	2	1	2	1	2	2	2	3	2	2	2	2	0	2	2
[2, 3]	2	2	1	1	2	2	2	3	2	2	2	2	2	0	2
[0, 1, 2]	1	1	1	2	2	2	2	3	2	2	2	2	2	2	0
Insertion/Deletion Costs for Nodes															
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[0, 1]	[0, 2]	[0, 3]	[1, 2]	[1, 3]	[2, 3]	[0, 1, 2]
insert	2	2	2	2	2	2	1	1	2	2	2	2	2	2	2
delete	2	2	2	2	2	2	1	1	2	2	2	2	2	2	2

Table 4.1: Substitution, Insertion and Deletion costs for Nodes based on those proposed by Harper *et al.*⁸⁰

Substitution Costs for Edges			
	-	=	≡
-	0	3	3
=	3	0	3
≡	3	3	0
Insertion/Deletion Costs for Edges			
	-	=	≡
insert	0	1	1
delete	0	1	1

Table 4.2: Substitution, Insertion and Deletion costs for Edges based on those proposed by Harper *et al.*⁸⁰

4.2.3 METHOD EVALUATION

As introduced in section 2.4, to evaluate the methodology in this chapter, we compute the number of errors or miss-classifications when trying to predict the bioactivity of molecules concerning a particular protein target. This evaluation metric was selected since it is the same used for the model optimization during the training process of a genetic algorithm. The genetic algorithm is optimized over 127 different targets extracted from the datasets mentioned before.

The set of molecules corresponding to each target was split by half to have “*train*” and “*test*” subsets. The “*train*” subset is used to optimize the transformation costs for the GED computation. The “*test*” subset is used to evaluate the recognition ratio with unknown data for the model.

OBJECTIVE FUNCTION.

Figure 4.1 shows the objective function we use together with a genetic algorithm^{71,162} to track the evolution of the learning process. The objective function is divided into three main steps. First, we compare each molecule with all the others and create a matrix of distances. Second, for each row in the matrix (which represents the distances computed for one molecule, the “query molecule”, with all of the others), we find the lowest distance D , which is considered as the “closest molecule” to the query one. Third, we use the “closest molecule” distance D and a log loss function¹⁴³, adding up the quantity $\exp(-D)$ as follows: if the “query molecule” and the “closest molecule” are from the same bioactivity classes, the objective function is decreased. On the contrary, if the “query molecule” and the “closest molecule” are from different bioactivity classes, the objective function increases. The log loss uses the error magnitude

in the prediction (how much it varies from the ground truth) to give a more continuous view over the model's behavior, slowly increasing or decreasing the objective function output values. Note that during the construction of the matrix of distances, if the molecules belong to the same bioactivity class, then the lower the distance, the better. On the other hand, if the molecules belong to different bioactivity classes, the higher the distance, the better.

The learning algorithm attempts to minimize the objective function to have as many correct classifications as possible. This minimization occurs since correct classifications reduce the resulting value and wrong classifications increase it, as explained before. The main goal of this process is to measure the performance of the objective function on each iteration and tune the values to be used as edit costs during the next iteration.

COMPUTATIONAL EXPENSE

The experiments reported in this chapter were performed using a quad-core, 2.0 GHz Intel Core i7 laptop with 8 GB RAM (operating system version Ubuntu 18.04). The most time-consuming step is the training of the edit costs employing the genetic algorithm; this process is computationally demanding due to the difficulty of finding costs that work better for all the molecules evaluated against the same target. In one experiment performed for one target only, the computation time was about 6 hours; the same task was executed for all 127 targets in the database. Fortunately, we executed all the optimizations in parallel since optimizing one target is entirely independent of others.

```
input: learning_set ,  
        edit_costs  
output: F  
  
begin  
    # step 1 : compute the matrix of distances  
     $\forall G_i$  in learning_set  
         $\forall G_j$  in learning_set  
            Dist(i, j) = GED( $G_i$ ,  $G_j$ , edit_costs)  
        end $\forall$   
    end $\forall$   
  
    # initialize the resulting value  
    F = 0  
  
    # for each row in the matrix of distances  
     $\forall G_i$  in learning_set  
        # step 2 : find the closest molecule and its index  
        D =  $\min_{\forall j} \{Dist(i, j)\}$   
        K =  $\operatorname{argmin}_{\forall j} \{Dist(i, j)\}$   
  
        # step 3 : add up or subtract depending on classification  
        if class( $G_K$ ) = class( $G_i$ )  
            F = F -  $e^{-D}$   
        else  
            F = F +  $e^{-D}$   
        endif  
    end $\forall$   
  
end
```

Figure 4.1: Objective function.

4.3 RESULTS

The aim of the practical experiments is twofold. First, we want to compare the recognition ratio deduced using the learned edit costs to the recognition ratio deduced using the edit costs proposed by Harper *et al.*⁸⁰ Second, we want to analyze if the learned costs are congruent with the chemical knowledge given by Harper *et al.* To that aim, we performed four experiments learning one different edit cost on each. The four edit costs were selected considering the most frequent node and edge attributes.

- **Experiment 1:** insertion and deletion costs corresponding to the carbon link node assigned to the “[6]” attribute in Table 2.2.
- **Experiment 2:** substitution cost between the carbon link node (“[6]” attribute in Table 2.2) and the aromatic ring system node (“[5]” attribute in Table 2.2).
- **Experiment 3:** insertion and deletion costs corresponding to the single bond edge assigned to the “-” attribute in Table 2.2.
- **Experiment 4:** substitution cost between the single bond edge (“-” attribute in Table 2.2) and the double bond edge (“=” attribute in Table 2.2).

Analysis of the experiments. Figure 4.2 shows an example of the learning behavior of a single target; the continuous blue line represents the objective function, which decreases after every learning iteration until convergence. The red segmented line represents the number of miss-classifications over the training set, and the green points represent the number of miss-classifications over the test set. Both training and test miss-classification values should decrease. However, it is not always the case since it depends on several factors, including the

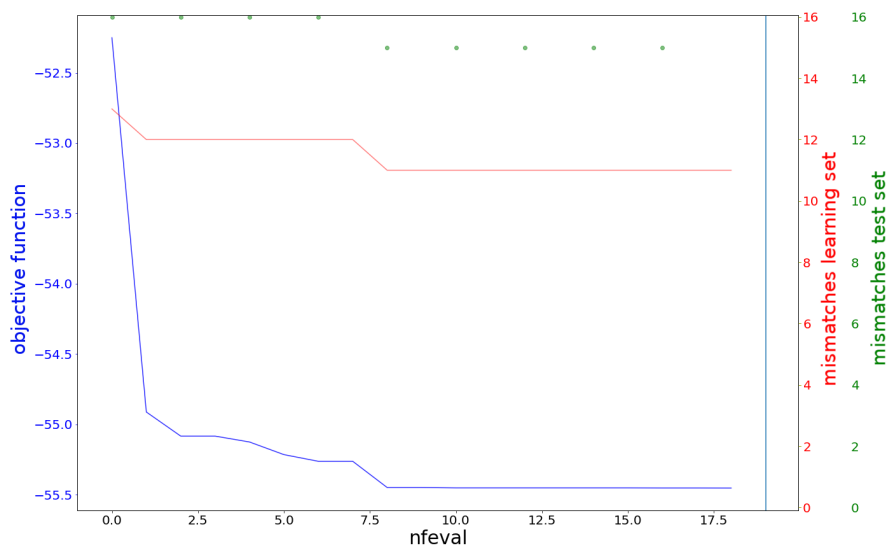


Figure 4.2: Training evolution curves for the FXA target in the DUD-E dataset.

training set's size, the number of variables being learned, the tolerance for convergence, and the over-fitting.

Figure 4.3 shows the number of errors in the classification process for **Experiment 4** over the test set and for all 127 targets, using two different edit cost configurations, the edit costs proposed by Harper *et al.*⁸⁰, and the edit costs we have learned. It is important to note that, since the figure depicts the number of miss-classifications, the lower the values, the better. These results show how the learned edit costs present a slightly improved behavior compared to the edit costs proposed by Harper *et al.* The improvement is noticed in the maximum and the third quartile being part of the box-and-whisker plots. All other quartile values are the same for both methods.

Figure 4.4 shows values for each dataset separately to have a better picture of the behavior using the learned costs. This figure shows the number of errors in the classification for two datasets in each experiment (we do not include all the results per experiment for space

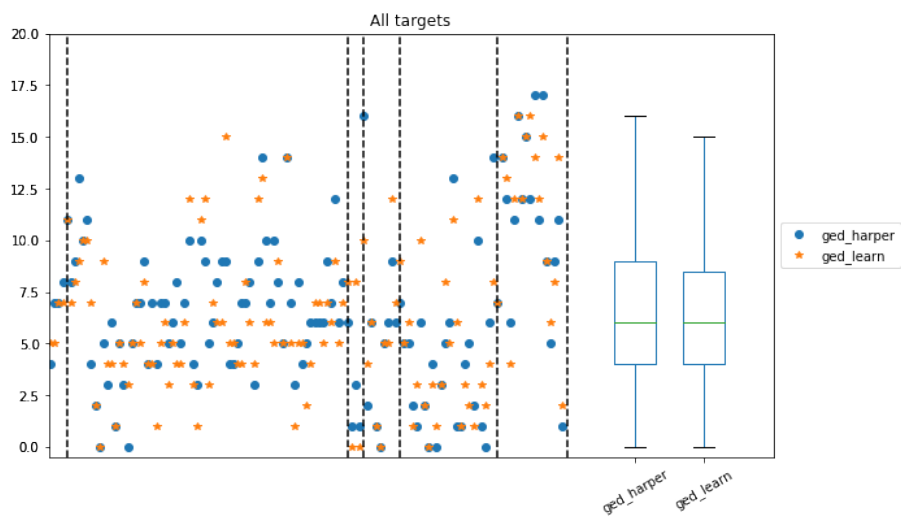


Figure 4.3: The number of miss-classifications in Experiment 4 using the test set over the 127 targets available in the six datasets combined. The scattered values on the left of the plot represent the number of classification errors. Different colors and shapes represent different sets of edit costs. Vertical segmented lines mark the limit between different datasets (from left to right: ULS-UDS, GLL&GDD, CAPST, DUD-E, NRLiSt_BDB, MUV). The box-and-whisker plots on the right show the distribution of the resulting values. The boxes show the first and third quartile, the line in the middle of the box is the median value (second quartile), and the whiskers extend from the boxes to show the range of the data (outliers are not included).

reasons, nevertheless, other results can be found as supplementary material). Note that each row represents an experiment and each sub-figure in the row represents a different dataset. As in Figure 4.3, here we show the number of miss-classifications, and box-and-whisker plots are located on the right of each subplot to illustrate the distribution of the values.

For the first experiment, we used the GLL&GDD and DUD-E datasets. The results using the learned edit costs are slightly better. The improvement obtained using the learned costs is observed in the third quartile and the maximum value for the GLL&GDD dataset and the median and maximum value for the DUD-E dataset. For the GLL&GDD, as the third quartile is reduced, notice how the GED using the learned costs provides more stable results. The stability is represented as the box and whiskers length. Shorter lengths indicate that several results are closer to each other, and therefore the method seems more reliable. For the other datasets in this experiment, results were similar, obtaining lower or equal median values using the learned edit costs compared to the edit costs from Harper.

For the second experiment, we used the ULS-UDS and GLL&GDD datasets. In this case, again, the results using the learned edit costs are significantly better. The improvement using the learned costs is exhibited in every aspect for the ULS-UDS dataset and most aspects for the GLL&GDD dataset, except for the minimum values in both cases, which are zero. For the other datasets in this experiment, median values using the learned and Harper's edit costs were the same except for CAPST, which obtained a better median value using Harper's ones.

For the third experiment, we used the CAPST and ULS-UDS datasets. In this case, only the results for ULS-UDS using the learned edit costs are better. This improvement is noticeable for all values in the box plot, including the three quartiles and the minimum and maximum. On the other hand, for the CAPST dataset, the values are equal for each target using

the learned edit costs and the edit costs proposed by Harper *et al.*⁸⁰ For the other datasets in this experiment, median values using the learned and Harper's edit costs were the same in every case.

Finally, for the fourth experiment, we used the NRLiSt_BDB and MUV datasets. In this experiment, the results using the learned edit costs are slightly better. The improvement obtained by using the learned costs is exhibited in the median and maximum values for the NRLiSt_BDB dataset. For the MUV, the improvement is more noticeable in the first quartile, minimum and maximum values. Nevertheless, the third quartile is better for the NRLiSt_BDB dataset using the costs proposed by Harper *et al.*⁸⁰ For the other datasets in this experiment, results were similar, obtaining lower or equal median values using the learned edit costs compared to the edit costs from Harper, except for CAPST, which obtained a better median value using Harper's ones.

Overall, the usage of GED together with ErGs using the learned edit costs obtained better recognition ratio results in most cases. During the experiment, we only learned one edit cost per case. We chose this straightforward and explicit approach to show the validity of our method. Our learning methodology can be applied to learn several edit costs at a time, either sequentially or in parallel. Learning a more significant number of edit costs might increase the recognition ratio values than those presented in this paper.

Table 4.3 shows the edit costs proposed by Harper *et al.*⁸⁰ and the learned edit costs obtained for each experiment. We can see how the learned edit costs of the first experiment tend to be lower than Harper's edit costs in most datasets. It means that, in general, inserting or deleting a link node should imply a lower cost than expected by Harper *et al.*⁸⁰ On the other hand, for the third experiment, learned edit costs tend to be slightly higher than Harper's edit

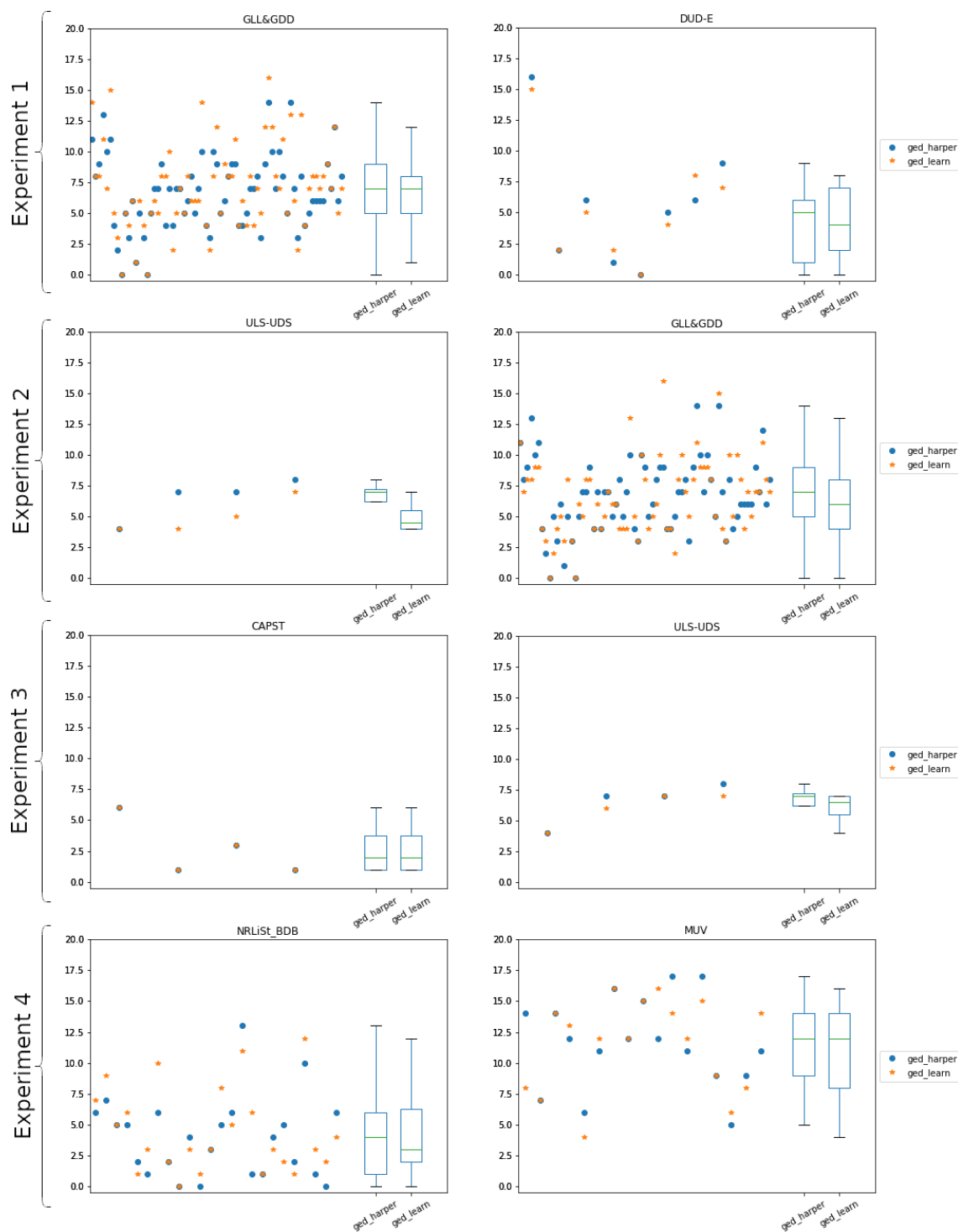


Figure 4.4: The number of miss-classifications for all available targets in the LBVS benchmarking platform separated per dataset and experiment. The scattered values on the left of each subplot represent the number of classification errors (the lower the values, the better) using different colors and shapes depending on the edit costs used. Box-and-whisker plots on the right of each subplot show the distribution of the resulting values for each experiment.

	Harper <i>et al</i>	CAPST	DUD-E	GLL&GDD	MUV	NRLiSt_BDB	ULS-UDS
Experiment 1	1	0.000002	0.005	0.014	0.490	0.012	0.115
Experiment 2	2	0.013	0.145	0.333	0.867	0.104	0.500
Experiment 3	0	0.004	0.001	0.003	0.327	0.003	0.011
Experiment 4	3	0.017	0.186	0.206	1.005	0.024	0.607

Table 4.3: Harper’s and learned costs (average values from all targets) per experiment.

costs, which means that inserting or deleting a single bond edge should imply a higher cost than Harper *et al.* expected. Furthermore, learned edit costs of second and fourth experiments tend to have a higher value than first and third experiments. It might imply that the substitution of a link node for an aromatic ring node or the substitution of a single bond edge for a double bond edge should carry a higher cost than inserting or deleting a link node or a single bond edge. This information is helpful to increase our knowledge about the structure-activity relationship within the molecules.

4.3.1 COMPARISON WITH THE PREVIOUS CHAPTER

In order to make a direct comparison of results obtained in this chapter with those from the previous chapter 3, we show in Figure 4.5 a plot with the AUC ROC values computed using Harper’s and learned transformation costs applied to the same subset of molecules used in this section.

The figure shows the superiority of both GED-based methods compared with the SED-based and the FP-based method. The difference between the GED using the edit costs proposed by Harper *et al.* and the edit costs we have learned is small. This mild difference in Experiment 3 is likely due to the usage of only one edit costs to be optimized, namely, the insertion/deletion cost corresponding to the single bond edge (“-” attribute in Table 2.2). Nevertheless, it is possible to notice how the minimum value and the median value are slightly

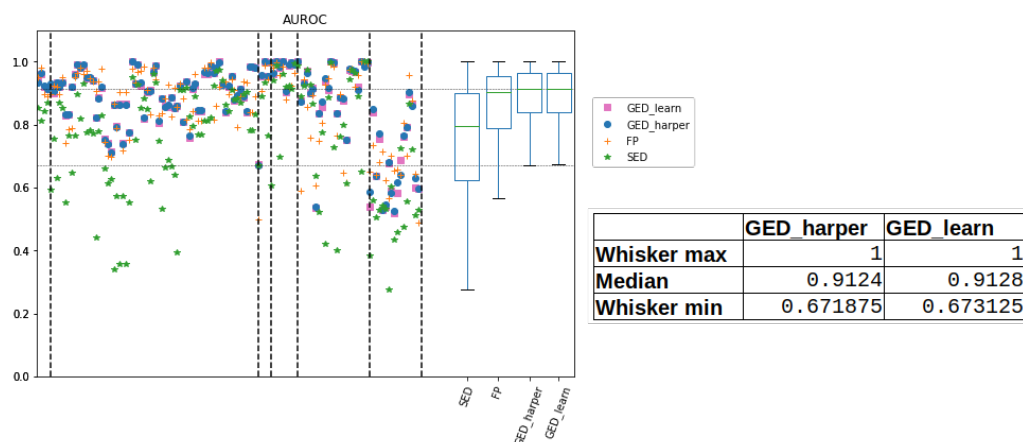


Figure 4.5: AUROC values comparison between chapter 3 and chapter 4 over the 127 targets.

better when using the optimized version of the edit costs.

4.4 DISCUSSION

In the previous chapter, a molecular similarity measure was presented, which uses graph edit distance to compare molecules' representation by extended reduced graphs effectively. At that moment, the edit costs for the different node and edge operations were assigned using expert knowledge. In this chapter, we used a learning algorithm to learn the edit costs automatically. Significant performance improvements were shown when using the learned costs in most experiments for the 127 targets present in the six datasets. All datasets used are publicly available and were formatted as part of the benchmarking platform proposed by Skoda and Hoksza.

Results show that the learned edit costs performed as good or better in most of the targets present in the six datasets, compared to the edit costs proposed by Harper *et al.*⁸⁰ Moreover, the learned costs may also give some ideas related to the structure-activity relationship present

within the activity classes.

The optimization process usually is computationally demanding, sometimes taking several hours to be completed. The next chapter will use a methodology inspired by natural language processing techniques to infer the transformation costs much faster.

*God not only plays dice, he throws them in the corner where
you can't see them.*

Stephen Hawking.

5

NLP tools to swiftly infer GED transformation costs for virtual screening

5.1 CHAPTER INTRODUCTION

Graphs are frequently used in real-world applications, and multiple complex systems are naturally shaped as networks, making them ideal to be studied by employing their graph representations. Some examples of such systems being: social networks where people follow and interact with each other⁵⁷, neural networks which mimic the way that neuron cells operate inside the animal brains³⁹, biological networks like Protein-Protein interactions^{6,166}, and linguistics or co-occurrence networks used in the field of text mining³⁵.

In certain straightforward cases, graph structures include only single nodes or sequences of nodes one after the other. However, in many applications, the data obtained from the systems is presented in much more complex structures like trees, cyclic graphs, or acyclic graphs. In those complex cases, network data can get considerably challenging to work with, so to process it effectively, the main challenge is to find the most convenient network data representation. In other words, finding how to represent networks clearly and concisely is a critical step to conduct efficiently any advanced analysis⁴⁴.

We can broadly separate Graph analytics tasks into four categories: node classification¹⁵, link prediction⁹⁷, clustering⁴⁷, and visualization⁹⁹. When working with any of these tasks, applying mathematical and statistical methods directly on the graphs is limited and challenging, also applying machine learning methods. This difficulty is probably because common graph data structures such as adjacency matrices and lists are plagued with sparsity, and sparse representations are natural setbacks for data analysis and machine learning applications, making them harder to generalize in statistical learning.

One solution to tackle the sparsity problem is to apply some dimensionality reduction

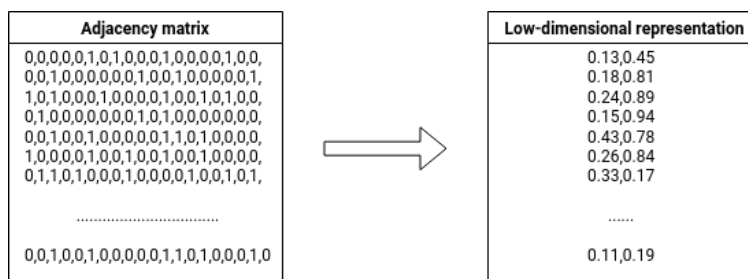


Figure 5.1: Example of conversion from an adjacency matrix to a low-dimensional continuous vector space.

technique, converting the sparse graph data into a low-dimensional representation known as graph embedding. Methods using this low-dimensional vector space are recently becoming widely popular^{1,165,176}.

Some popular matrix decomposition techniques are singular value decomposition (SVD)⁷³ and multidimensional scaling (MDS)¹⁰⁷, but those techniques have a running time of Matrix Multiplication, which is around n^3 (see Figure 5.1). This running time is not a big problem for many applications; however, it might be unfeasible for graph representations involving thousands or even millions of nodes. On top of that, those matrix decomposition techniques must be recomputed as nodes change or new nodes are added to the graph. To overcome those challenges, we could use techniques coming from machine learning to compute the low-dimensional embeddings; some examples of using machine learning for graph representation are Deepwalk¹³⁰ and Node2vec⁷⁷ (see Figure 5.2).

Graph embeddings should catch the graph's topology, vertex-to-vertex relationship, and other information about the subgraphs and vertices. We can divide embeddings into two broad groups: vertex embeddings and graph embeddings. In vertex embeddings, each vertex has its independent vector representation, which is helpful to obtain specific information or make predictions on the vertex level, e.g., studying the role or specific atoms inside a molecular

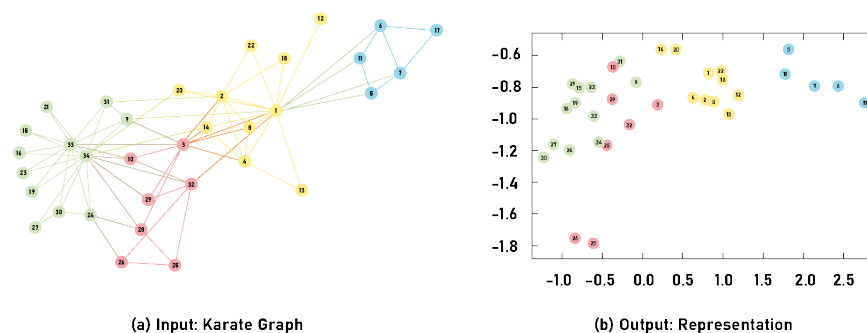


Figure 5.2: Illustration of the 2-dimensional embeddings obtained from the graph on the left. The embeddings can be plotted and clustered based on metrics such as the Euclidean distance.

structure. On the other hand, graph embeddings capture the information from the whole graph in a single vector, which is helpful to obtain information or make predictions on the graph level, e.g., studying the toxicity of whole molecular structures.

Before discussing embeddings, it is worth talking about Word2vec and the skip-gram model¹¹² since they are the base for this work's graph embedding method. Word2vec is also an embedding methodology, but instead of transforming graphs into vectors, it transforms words into vectors. After applying Word2vec to an input text (also known as dictionary) and get the embeddings as an outcome, a simple mathematical function like the cosine similarity between the embedding vectors should indicate the level of semantic similarity between the words represented by those vectors. Therefore, low cosine distance should represent two similar or related words such as orange and apple, and high cosine distance should represent not very related words such as car and closet.

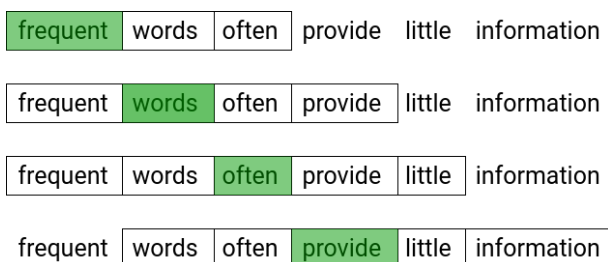


Figure 5.3: The green-box words are given to the network as targets; they are optimized to predict the words in the neighborhood (white-box words). In this example, we consider words located up to two places away from the input word.

5.1.1 WORD2VEC

Word2vec uses a neural network with one hidden layer to train the input dictionary and build the embeddings; this particular case of a neural network is called skip-gram. It is specifically designed to learn the likelihood of words appearing in the proximity of other words into a sentence; those nearby words are also known as “neighbor words”.

The skip-gram neural network is used only during the training phase; once this phase is finished, we can obtain the embedding vectors from the trained values assigned to the artificial neurons placed in the middle hidden layer. Figure 5.3 shows an example with a short text where the boxes represent the training neighborhood words predicted from the green-highlighted word used as an input. The main goal for the skip-gram model is that two similar words should have similar neighborhoods, and therefore, two similar words should also have similar embedding vectors.

Figure 5.4 shows the main building blocks of a skip-gram neural network: the input layer, one middle hidden layer, and the output layer. As for the input, the network receives the words encoded as one-hot vectors (see the vector in the input layer in Figure 5.4). Those one-hot vectors for each word should have the same length as the word dictionary (the number of

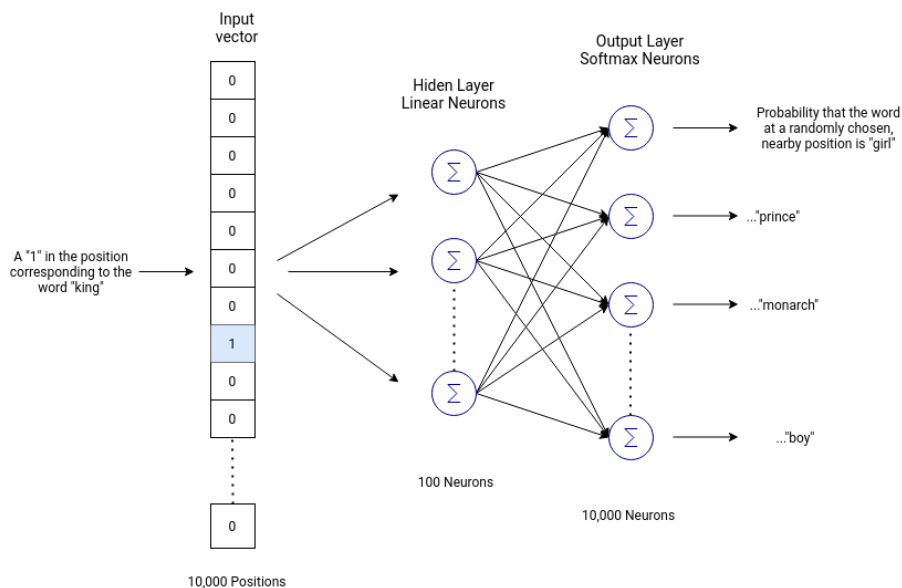


Figure 5.4: Components of a skip-gram neural network.

words in the text corpus used to train the network). All the values in a one-hot vector should be zero except the one indicating the word it encodes in the same order as they appear in the dictionary, as explained in Figure 5.5. The hidden layer does not have any activation function, and the final training values represent the word's embedding. Finally, the output layer uses a Hierarchical Softmax^{113,116,118} as an activation function to approximate the probability distribution and speed up the training process. After training, this last layer indicates the neighborhood words predicted for a given input.

Some works helpfully analyze and explain the Word2vec working process to be more easily understood by non-experts^{142,72}, including tools used as a playground and learn by using the actual algorithms. Figure 5.6 shows the 2D impression of the embeddings learned from a simple training example using a small dictionary; this plot was made using one of the aforementioned tools¹⁴².

Vocabulary:
man, woman, boy, girl, prince,
princess, king, queen, monarch.

⇒

one-hot vectors

man	0	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
king	0	0	0	0	0	0	1	0	0
queen	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Figure 5.5: One-hot vectors construction for an example dictionary of words.

After discussing the working mechanism of the skip-gram model for word embeddings used in Word2vec, we can now move to graph embeddings and describe two related works, namely DeepWalk and Node2vec, which also make use of the skip-gram model.

When working with nodes, we also need an input set of phrases like the word dictionary mentioned in Word2vec. Since we do not have such dictionaries or phrases in graphs, we can build an analogy out of walks through the network as described in the following sections.

5.1.2 DEEPWALK

DeepWalk employs the use of random walks to generate the walk-dictionary. To build a random walk, we start in a selected node, move to a random neighbor, and keep moving randomly for a defined number of steps. In the end, the walk will be the combination of random nodes selected and the links or bonds between them.

The whole DeepWalk method can be broadly separated into three steps:

1. Sampling the graph: The idea is sampling the graph by using random walks; not many random walks per node are necessary, and they do not need to be very long, but indeed, the number of walks and their length is usually different depending on the size and type

wevi: word embedding visual inspector

Everything you need to know about this tool - Source code

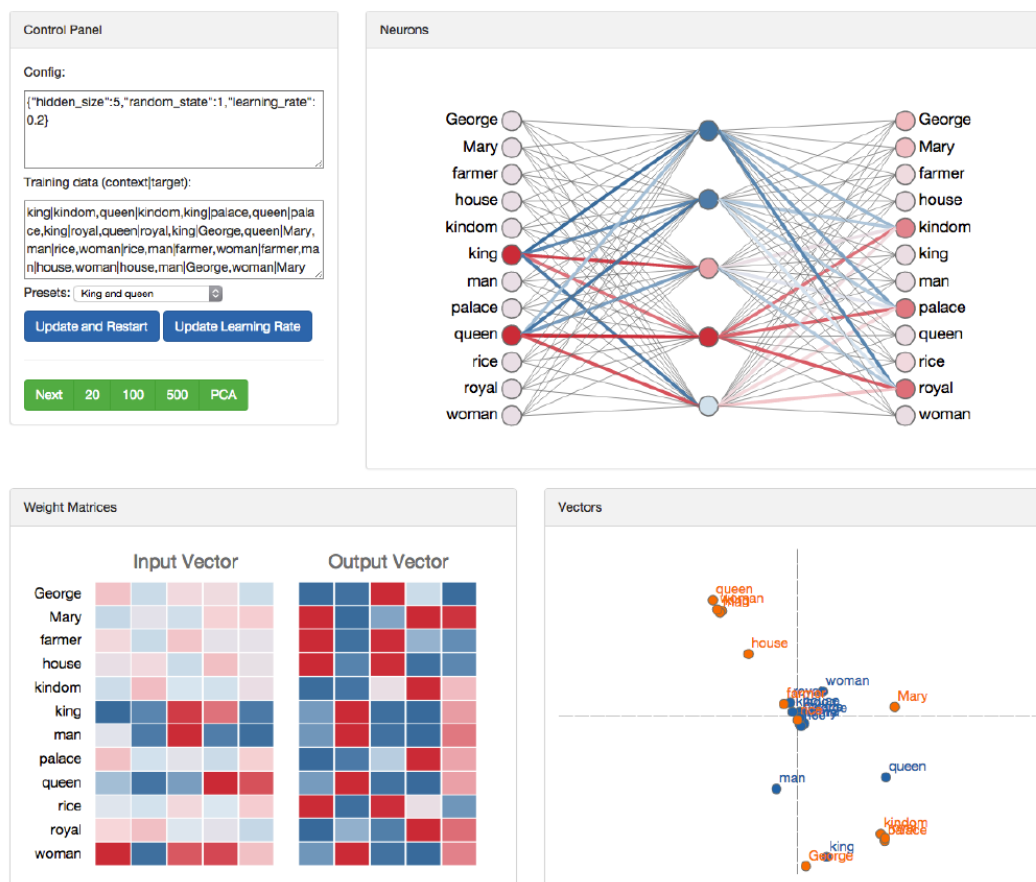


Figure 5.6: Wevi screenshot¹⁷⁸. The top left shows the control panel with the input dictionary and the learning settings. The top right shows the evolution of the neurons in the neural network during the training. The bottom left shows the weight matrices for the input and output vectors from each word in the dictionary. The bottom right shows the PCA distribution of the embedding vectors obtained from the words in the dictionary. (Source: <http://bit.ly/wevi-online>.)

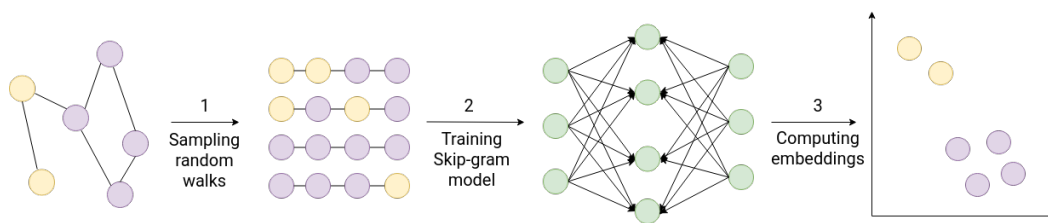


Figure 5.7: Phases of DeepWalk.

of the graphs used in the study.

2. Training the skip-gram: Similar Word2vec using the skip-gram to predict the probability of certain neighbor words by training the neural network with a collection of one-hot vectors, the skip-gram in DeepWalk can predict the probability of specific neighbor nodes by training the neural network also with a collection of one-hot vectors representing the nodes.
3. Computing the embedding vectors: In DeepWalk, and also in Word2vec, the embeddings are obtained as the outcome of the hidden layer once the training step is completed; the only difference is that Word2vec computes the embeddings for words in a dictionary whereas DeepWalk computes them for nodes in a graph.

5.1.3 NODE2VEC

Node2vec is another implementation of the skip-gram model applied to graph networks. As DeepWalk, Node2vec predicts the probability of specific neighbor nodes in the same window. One crucial difference between these two approaches is the sampling procedure used to generate the random walks; DeepWalk generates them randomly, whereas Node2vec includes two parameters, P and Q , to discern between the type of walk we prefer, either BFS

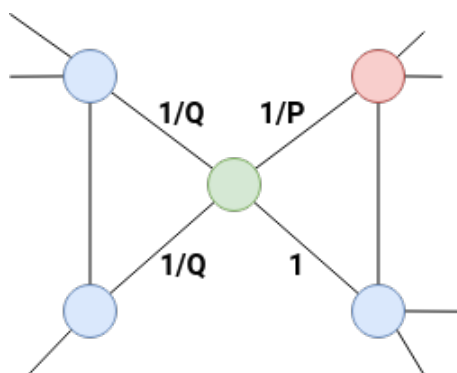


Figure 5.8: Probabilities of a random walk step in Node2vec after moving one step from the red to the green node. The probability of going back to the red node is $1/P$; the probability of going to a non-neighbor of the previous node (red) is $1/Q$; the probability of going to a neighbor of the previous node (red) is 1 .

or DFS. In BFS (Breadth-first Sampling), the neighborhood is restricted to neighbors of the source node. In DFS (Depth-first Sampling), the neighborhood comprises nodes sequentially sampled at increasing distances from the source node. Furthermore, parameter Q defines the random walk probability of discovering new parts of the graph far from the source node, and parameter P defines the random walk probability of returning to a previous node while exploring the network (see Figure 5.8). The other steps for the embedding construction are the same as for DeepWalk.

Mol2vec⁸³ is another work where the authors used a similar approach applying the skip-gram model for graphs, optimizing the model to learn the vector representations of molecular substructures. Like Word2vec, where closely related words are assigned to nearby embeddings in the vector space, in Mol2vec, the closely placed embeddings represent chemically related substructures.

The skip-gram model in Word2vec is trained with an input dictionary made up of sentences and words. Mol2vec also uses an input text where instead of “words”, they use Morgan fingerprints¹⁴⁰, and instead of “sentences”, they use compounds. Each compound is encoded

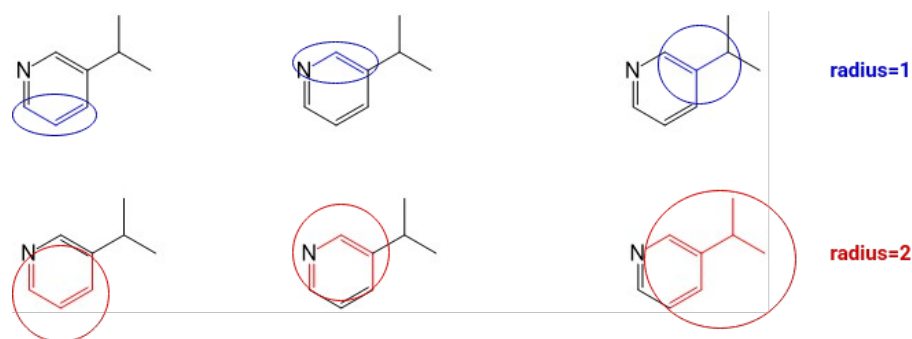


Figure 5.9: Morgan algorithm's substructure selection on heavy atoms for two different radii.

as one vector by summing the Morgan fingerprints of the individual substructures inside the molecule.

Morgan fingerprints (also known as extended connectivity fingerprints (ECFPs)) are commonly used to represent the chemical features of the molecules since they usually outperform other types of fingerprints in molecular activity prediction^{138,104,110,158}, as well as in similarity search and virtual screening tasks^{139,127}. The Morgan algorithm generates and assigns unique identifiers to all the possible substructures obtained from the heavy atoms of the molecule; these substructures are restrained to a predefined radio around the heavy atoms, as presented in Figure 5.9. The resulting identifiers (one for each heavy atom) are also known as Morgan identifiers; they are usually hashed and combined into a vector with a predefined length to obtain the compound Morgan fingerprint of the whole molecule.

In their work, after processing all the molecules and obtain the molecular “sentences”, the authors used them as typical text sentences to train a Word2vec model and obtain the embeddings. Later, the embeddings train different machine learning methods like random forest and deep neural networks and perform molecular analysis like substructures relationships, compound similarities, and supervised ML predictions.

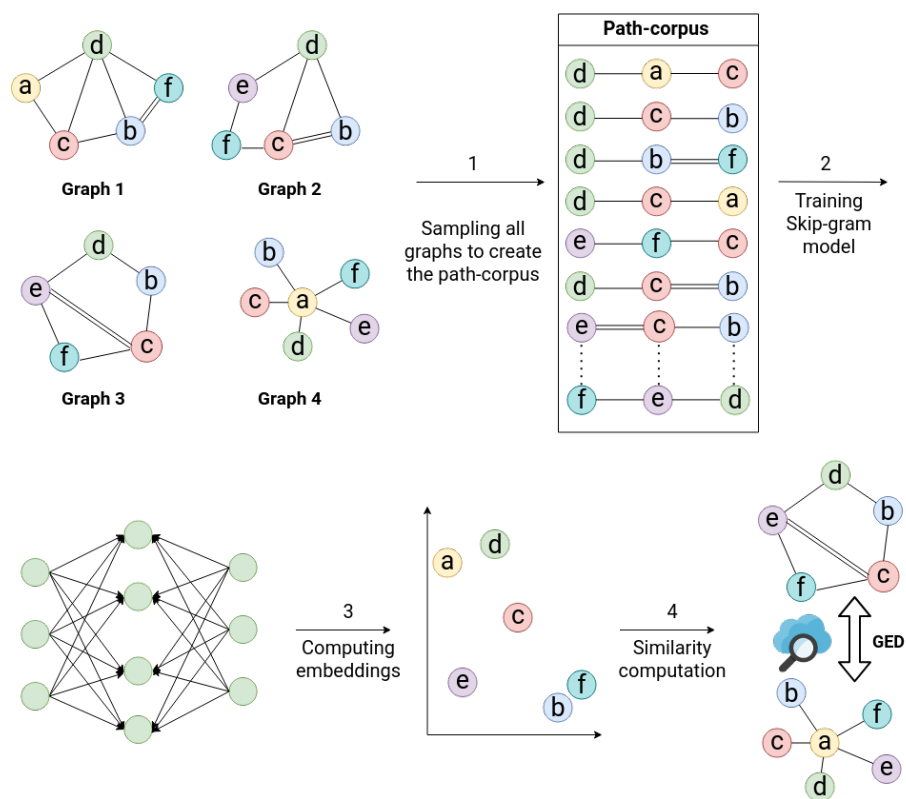


Figure 5.10: Phases of the approach used in this chapter.

This chapter uses the skip-gram model to learn the features and compute a relative distance between different nodes in a graph, similar to how it is done in DeepWalk and Node2vec. The graph we use is built with the representations of chemical substructures obtained with the ErG methodology; in other words, we are training the skip-gram with the molecular substructures from compounds, similarly to how it is done in Mol2vec. Although in our case, we use the spatial distances from the resulting embedding vectors as transformation cost values to compare molecular graphs through the GED methodology applied to chemical compounds for virtual screening. The entire process is presented in Figure 5.10.

Datasets	Molecules	Avg Degree	Avg Size	Min Size	Max size	Problem type
MAO	68	2.1	18.4	11	27	Classification
PTC	416	2.1	14.4	2	64	Classification
AIDS	2000	2.1	15.7	2	95	Classification
ACYCLIC	185	1.8	8.2	3	11	Regression

Table 5.1: Feature summary of the datasets.

5.2 SPECIFIC MATERIALS AND METHODS

5.2.1 DATASETS

In this chapter we used four public available datasets: MAO⁶³, PTC¹⁶⁹, AIDS¹³⁵ and Acyclic⁴⁰. Table 5.1 shows the general properties for each dataset, including the number of molecules, average degree, average size, minimum size, maximum size, and the type of problem. All these datasets are available on the GREYC research group's dataset repository⁷⁶, with whom we worked together to develop the experiments below.

A classification problem predicts the class of a molecule; the classification can be binary (each molecule can belong to the positive or negative) or multi-class class (each molecule can belong to one of several distinct classes). On the other hand, a regression problem consists of predicting a property of a chemical compound, which can take the value of any real number.

The MonoAmine Oxidase (MAO) dataset is a classification problem that predicts the inhibitory character of molecules on monoamine oxidase. The dataset consists of 68 molecules divided into two classes: 38 molecules inhibit monoamine oxidase (characteristic of an antidepressant drug), and 30 do not. These molecules are represented with cyclic and labeled graphs. However, the sets of cycles of each molecule do not differ much from one molecule to another.

The Predictive Toxicity Challenge (PTC) dataset is taken from the Predictive Toxicity Challenge¹⁶⁹. It proposes a classification problem: to predict the carcinogenicity of 416 molecules distributed over four classes of animals: female rats (FR, 351 molecules), male rats (MR, 344 molecules), female mice (FM, 349 molecules), and male mice (MM, 336 molecules). Each class of animals is separated into learning and testing sets. Each molecule is made up of heteroatoms and rings and is represented by cyclic and labeled molecular graphs.

The AIDS dataset is taken from¹³⁵ and is based on the AIDS Antiviral Screen Database of Active Compounds. It consists of 2000 diverse chemical compounds, some of them made up of disconnected molecules. These chemical compounds have been classified as active or inactive against HIV. They are divided into three groups: learning set with 250 compounds, validation set with 250 compounds, and test set with the remaining 1500 compounds.

The ACYCLIC dataset comprises 185 acyclic ether molecules used in regression problem experiments to predict their regular boiling point⁴⁰. These molecules are acyclic and include heteroatoms (atoms with a chemical element different from carbon) and are thus represented as acyclic labeled graphs. The boiling temperatures range from -23.7 °C to 250 °C.

5.2.2 MOLECULAR REPRESENTATION AND COMPARISON

This chapter employs the GED-based method, presented in detail in section 1.5.3, to compare molecules represented as the ErGs described by Stiefl *et al.*¹⁶¹, where node features represent pharmacophore-type node descriptions.

In chapter 3, we used the edit costs proposed by Harper *et al.*⁸⁰, and in chapter 4, we used a genetic algorithm to optimize the edit costs. Differently, in this chapter, we use a much less computationally expensive method, the skip-gram model, to learn the features and compute

a relative distance between different nodes in a graph. Then we use those relative distances as transformation costs for the GED.

GED COMPUTATION

As in chapters 3 and 4, here we compute the approximation of the GED in polynomial time using an in-house implementation of the bipartite graph matching method proposed by Serfatosa¹⁵⁰; it was programmed in C++ and Python languages.

GED TRANSFORMATION COSTS

During the graph comparison process, the GED-based method uses three different sets of edition costs:

1. The first set of costs uses a random distance between different types of nodes and, therefore, a random distance between molecules; this is used as a reference to confirm whether the distances obtained with the other sets behave randomly or not.
2. The second set of costs uses a fixed value when comparing different node types and zero if comparing the same type of nodes as presented in Tables 5.2 and 5.3; we refer to this set of costs as “Direct Costs”.
3. The third set of costs is the one obtained from the skip-gram model and its embeddings.

Substitution Costs for Nodes															
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[0, 1]	[0, 2]	[0, 3]	[1, 2]	[1, 3]	[2, 3]	[0, 1, 2]
[0]	0	3	3	3	3	3	3	3	3	3	3	3	3	3	3
[1]	3	0	3	3	3	3	3	3	3	3	3	3	3	3	3
[2]	3	3	0	3	3	3	3	3	3	3	3	3	3	3	3
[3]	3	3	3	0	3	3	3	3	3	3	3	3	3	3	3
[4]	3	3	3	3	0	3	3	3	3	3	3	3	3	3	3
[5]	3	3	3	3	3	0	3	3	3	3	3	3	3	3	3
[6]	3	3	3	3	3	3	0	3	3	3	3	3	3	3	3
[7]	3	3	3	3	3	3	3	0	3	3	3	3	3	3	3
[0, 1]	3	3	3	3	3	3	3	3	0	3	3	3	3	3	3
[0, 2]	3	3	3	3	3	3	3	3	3	0	3	3	3	3	3
[0, 3]	3	3	3	3	3	3	3	3	3	3	0	3	3	3	3
[1, 2]	3	3	3	3	3	3	3	3	3	3	3	0	3	3	3
[1, 3]	3	3	3	3	3	3	3	3	3	3	3	3	0	3	3
[2, 3]	3	3	3	3	3	3	3	3	3	3	3	3	3	0	3
[0, 1, 2]	3	3	3	3	3	3	3	3	3	3	3	3	3	3	0
Insertion/Deletion Costs for Nodes															
	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[0, 1]	[0, 2]	[0, 3]	[1, 2]	[1, 3]	[2, 3]	[0, 1, 2]
insert	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
delete	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 5.2: Direct Substitution, Insertion, and Deletion costs for Nodes.

Substitution Costs for Edges			
	-	=	≡
-	0	3	3
=	3	0	3
≡	3	3	0
Insertion/Deletion Costs for Edges			
	-	=	≡
insert	1	1	1
delete	1	1	1

Table 5.3: Direct Substitution, Insertion, and Deletion costs for Edges.

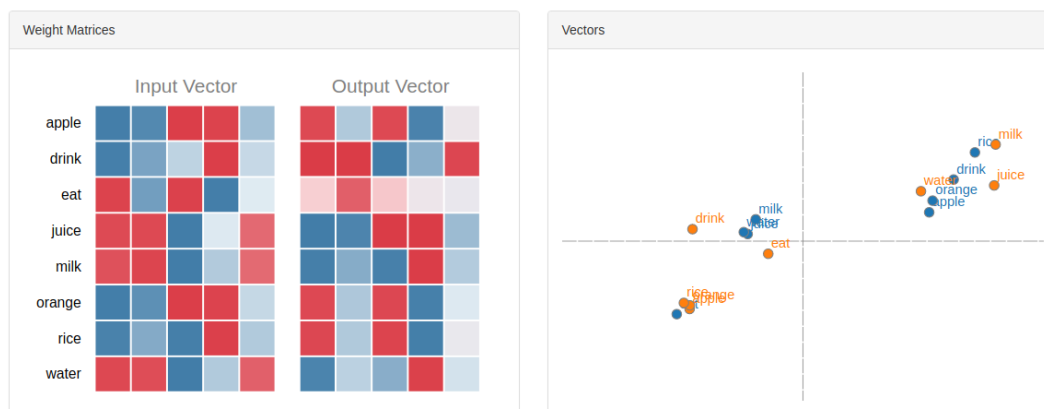


Figure 5.11: Wevi screenshot¹⁷⁸ using an example of a small dictionary and after only 200 iterations. Left shows the weight matrices for the input and output vectors. Right shows the PCA distribution of the embedding vectors. In the bottom left of the PCA plot, we can notice a cluster with the words “orange”, “rice”, and “apple” in orange and the word “eat” in blue; implying a semantic relationship between the fruits and the verb “eat” higher than with “milk” or “water” whose vectors are closer to the verb “drink”. (Source: <http://bit.ly/wevi-online>.)

5.2.3 EMBEDDING GENERATION

The Skip-gram model is a machine learning technique for natural language processing that learns word associations from an input text or dictionary, as described in section 5.1.1. While training, the neural network fixes a word and uses it to predict those nearby or neighbor words. Once trained, the model can identify synonyms and related words and even make suggestions to complete a partial sentence¹¹².

The objective function in the skip-gram neural network seeks to maximize the co-occurrence probability for each word within a specified window in a sentence. Due to this objective function, words appearing in similar contexts end up having similar embeddings; the distance between embeddings can be measured by the cosine similarity or the euclidean distance (see Figure 5.11).

The graph we use is built with the representations of chemical substructures obtained with

the ErG methodology; in other words, we will be training the skip-gram with the molecular substructures from compounds, similarly to how the authors do it in Mol2vec.

We use the skip-gram model in a similar way to Word2vec. However, instead of learning relative distances between words, we seek to learn the relative distance between nodes in ErGs to later use those distances as transformation costs for the GED methodology. Following, we will discuss the process to obtain the embeddings.

JOINED PATHS AS A TEXT CORPUS

Word2vec uses a corpus of text (word-dictionary) as input to train the skip-gram model; from this corpus and for each training step, the algorithm extracts a sentence of a certain length (window) and uses it to learn the word co-occurrence probability within that specific window. The algorithm then moves on to the following window and repeats the same procedure as exhibited in Figure 5.3.

As described in sections 5.1.2 and 5.1.3, Deepwalk and Node2vec employ random walks to generate a walk-dictionary, treating walks as the analogy of sentences in Word2vec; each walk will combine different nodes and the links or bonds between them.

We use an implementation based on Node2vec with a slight difference to create the walks. Node2vec was thought mainly to be applied on big graphs like social interaction networks; unlikely, we are working with small molecular graphs, then we create the walks out of shortest paths between all pairs of nodes. Next, we remove all paths shorter than a given value “n2v_walk_length” and remove duplicate paths or reverse duplicate paths (identical paths but opposite order). Later we use the remaining paths as sentences to feed into the skip-gram model, where they are treated similarly to the sentences in the text-corpus used by Word2vec;

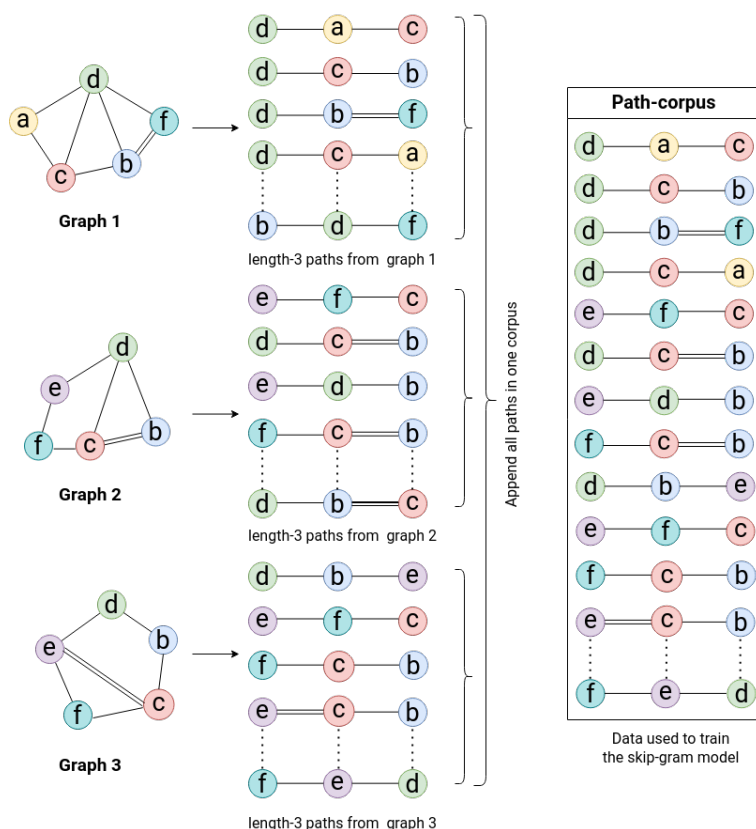


Figure 5.12: The process of joining together paths from different graphs.

we can call this corpus “path-corpus”. The path-creating process is exemplified in Figure 5.12.

VISUALIZING THE NODE EMBEDDINGS

Here we use the gensim¹³³ skip-gram implementation of Word2vec to compute the embeddings of the sentences we created using the shortest paths. This implementation is a shallow, two-layer neural network.

Figure 5.13 shows a 2D and 3D visual representation of the embeddings obtained in one of the experiments using the AIDS dataset.

We use the cosine similarities or euclidian distances among all the embeddings as transfor-

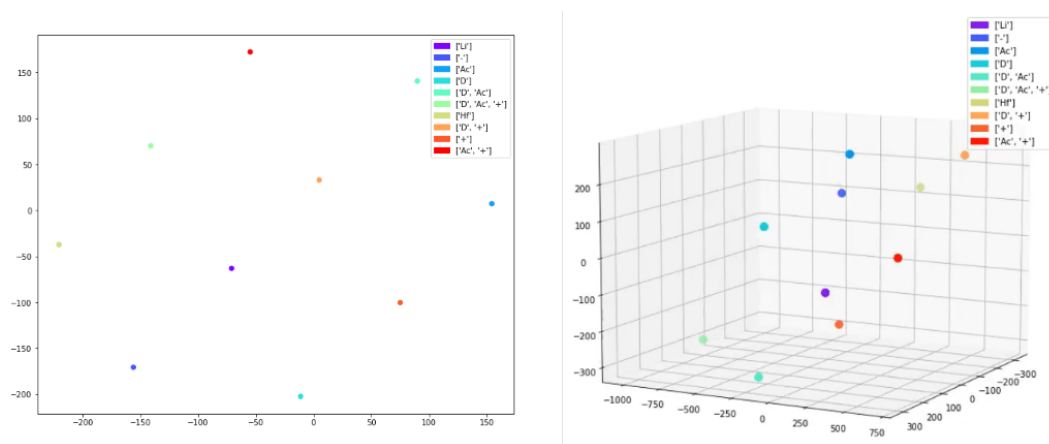


Figure 5.13: 2D and 3D T-SNE visualizations of embeddings from the AIDS dataset.

mation costs for the GED computation. In other words, those distances will substitute the edit costs proposed by Harper *et al.*⁸⁰ used in chapter 3; and will also substitute the optimized edit costs used in chapter 4.

DIMENSIONALITY EVOLUTION AND STABILITY

The quality of word embedding increases by using a higher number of dimensions; nevertheless, the quality stops increasing after a certain point; it either finds a plateau or starts decreasing¹¹². Therefore, it is a good practice to analyze the dataset to find a coherent number of dimensions.

We ran experiments and measure the stability by relative square error (RSE) between the cosine distances among all the embeddings. The cosine distances among embeddings are the values we use as transformation costs for the GED computation; thus, we compute those distances while increasing the number of dimensions and later measure the stability like:

$$\sum |(C_n - C_{n-1})|^2$$

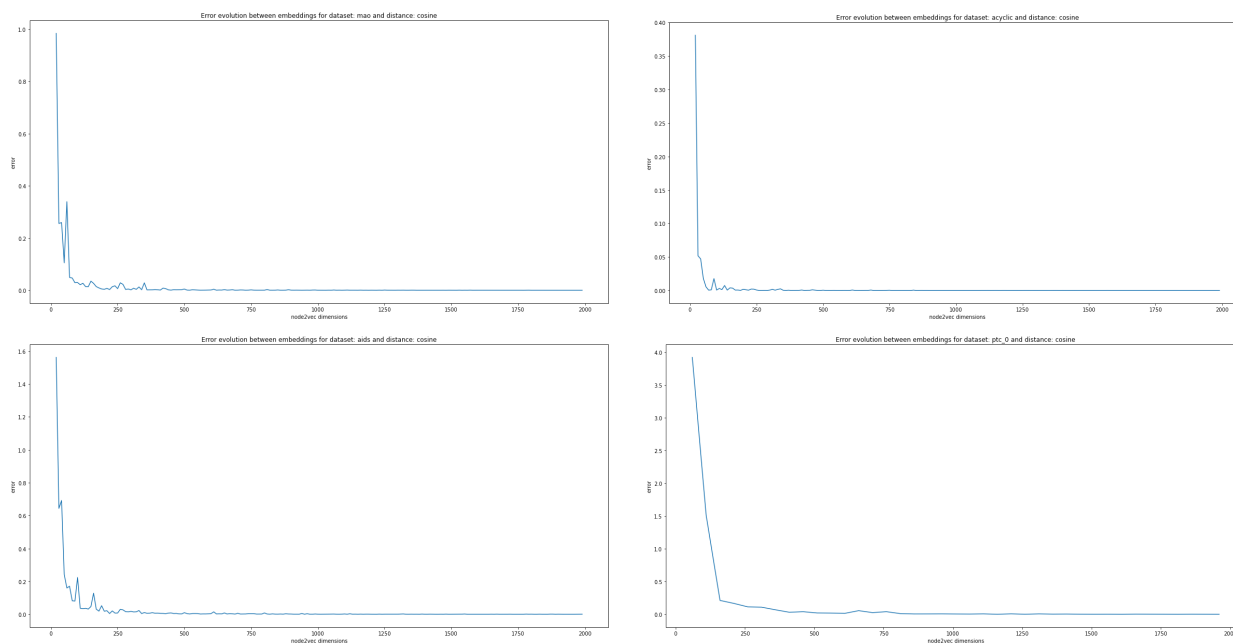


Figure 5.14: Stability evolution of embeddings with different datasets.

where C_n is the matrix of cosine distances among all embeddings computed using n dimensions, and C_{n-1} is the same computed using $n - 1$ dimensions starting from $n = 10$ until $n = 2000$.

Figure 5.14 shows the evolution of cosine distances among embeddings with different datasets. The distances were relatively stable for all cases when dimensions $n = 1000$; hence, we use that value for the experiments below.

OTHER SETTINGS TO COMPUTE THE EMBEDDINGS

Word2vec can be trained in two ways: one option is the continuous bag of words (CBOW) when the objective function attempts to predict a word from a context regardless of the order of words in the input context (thus the bag-of-words assumption). The other option is the

Skip-gram when the context is predicted according to the input word and assigns a higher weight to closer words inside the input context. We use the Skip-gram model only because we want to learn the graph's structure, giving more significance to nodes closer to each other.

Other parameters in the skip-gram algorithm like "window size" were explored to find the best settings. The "window size" indicates the size of the context used by the algorithm for the training stage; we use a value of 3 in our experiments.

Any non-frequent type of node not appearing in the training data will be treated as "UNKNOWN" for the rest of the experiments, assigning a transformation cost equal to "INFINITY". The goal is to ignore those ErGs having rare nodes since Word2vec cannot get meaningful embeddings for rare words.

COMPUTATIONAL EXPENSE

The experiments reported in this chapter were performed using an octa-core, 3.5 GHz Intel Core i7 laptop with 16 GB RAM (operating system version Ubuntu 18.04). Compared with chapters 3 and 4, the computations performed here are significantly light and quick. The average time to compute the embeddings for one dataset is not more than a couple of seconds. For instance, the computation of the embeddings and cosine distances for the AIDS dataset is about 4 seconds; the ACYCLIC dataset is about 400 milliseconds; the MAO dataset is about 370 milliseconds, and for any of the PTC sub-sets is about 1.5 seconds. Those execution times are almost insignificant compared to the optimization time needed for the genetic algorithm used in chapter 4.

5.2.4 METHOD EVALUATION

As we discussed before in section 5.2.2, in this chapter, we compare three different sets of edit costs to confirm whether the information obtained through the embedding process is significantly better than other methods. The three versions of edit costs used to compute the GED are:

- “random-costs”, used as a reference to compare the behavior of our method with that of a random distance generator.
- “direct-costs”, based on a set of fixed values as presented in Tables 5.2 and 5.3.
- and our method, “embedding-costs”, obtained through the skip-gram model.

To evaluate the predictive accuracy of the different edit cost sets, we use a version of the “*k*-nearest neighbors” algorithm (*k*-NN) depending on the type of task we are working with, being either a classification or a regression problem.

For **classification** tasks, namely MAO, PTC and, AIDS, we use the *k*-NN classification model to perform the predictions; later, we use a simple counting of correct and incorrect outputs to evaluate the quality of results. For **regression** tasks, namely ACYCLIC, we use a *k*-NN regression model², where *k* is the number of neighbors taken into account to perform the prediction. Later, we use the Residual Sum of Squares (RSS), defined as the sum of squared prediction errors made for each molecule, to evaluate the quality of results.

Datasets available on the GREYC repository⁷⁶ are often conveniently separated into different splits to be used independently as train and test sets. We use the test set to measure the methods’ prediction accuracy with a specific set of edition costs. The train set is used to op-

optimize the hyper-values, e.g., the k value optimized using a 5-fold cross-validation procedure for all values between 1 and 25.

5.3 RESULTS

The experiments carried out in this chapter are aimed to directly compare the predictive accuracy among three different sets of edition costs. We tested the three sets with an in-house implementation of the bipartite graph matching method used to compute the approximation of the GED in polynomial time.

The pipeline for each experiment is always the same:

1. select one of the databases mentioned in section 5.2.1.
2. transform the molecular graphs in the dataset into ErGs.
3. create the path-corpus by joining all the paths available in the ErGs as described in section 5.2.3.
4. compute the embeddings via the skip-gram model while increasing the number n of dimensions.
5. verify the stability of the embeddings when $n = 1000$ and use those embeddings for the next steps.
6. measure the spatial distance among all the embeddings and use them as transformation costs for the GED computation
7. use the GED and the transformation costs to compute the distances among all the molecules and create a molecular distance matrix.

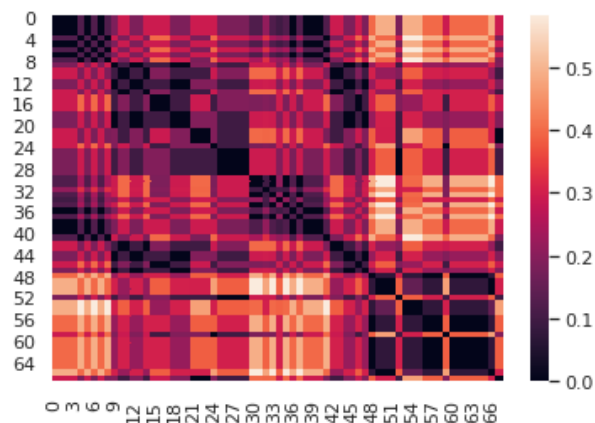


Figure 5.15: Heatmap using GED values with MAO dataset.

8. evaluate the accuracy of the distance matrix using the methodology described in section 5.2.4.
9. analyze the outcome of the evaluation.

Analysis of the experiments.

Figure 5.15 shows the heatmap of an example molecular distance matrix obtained with the MAO dataset and the transformation costs obtained from the embeddings. In this example, molecules were organized by activity, placing all active molecules at the beginning and all inactive molecules at the end. The figure shows two different groups indicating how distance values tend to be lower for molecules belonging to the same activity group and higher otherwise.

Figure 5.16 shows the bar plots with the values obtained in the experiments using the four subsets in the PTC dataset and the three different sets of transformation costs. The GED classification accuracy is higher for all subsets using the transformation costs computed from the embeddings (`ged_embs`). For all subsets, the accuracy obtained by using the direct values

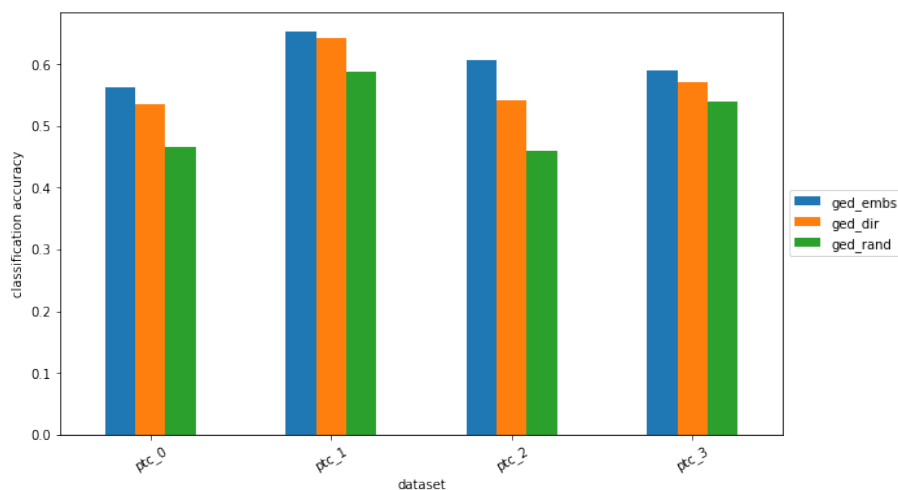


Figure 5.16: Results for PTC_0, PTC_1, PTC_2, and PTC_3 datasets.

(ged_dir) is lower than the embeddings; however, it is higher than the one achieved using the random values as transformation costs (ged_rand).

Figure 5.17 (a) shows the bar plots with the values obtained using the MAO and AIDS datasets. With MAO, the classification accuracy obtained with ged_embs and ged_dir is the same, both being better than ged_rand. A similar situation occurs for AIDS, where ged_embs and ged_dir are very alike; nevertheless, they are not the same (see Table 5.4).

Figure 5.17 (b) shows the bar-plot with the values obtained using the ACYCLIC dataset. In this experiment, the regression error RSS computed with ged_embs is lower than that computed with ged_dir and ged_rand. Since this is a regression dataset, the measure is based on the differential error; therefore, lower values are better.

Table 5.4 includes the actual values obtained for the experiments using the seven datasets and the three versions of the transformation costs. Here we can confirm that results are as good or better for all cases when using the transformation costs based on embeddings, rather than using the transformation costs based on direct or random values.

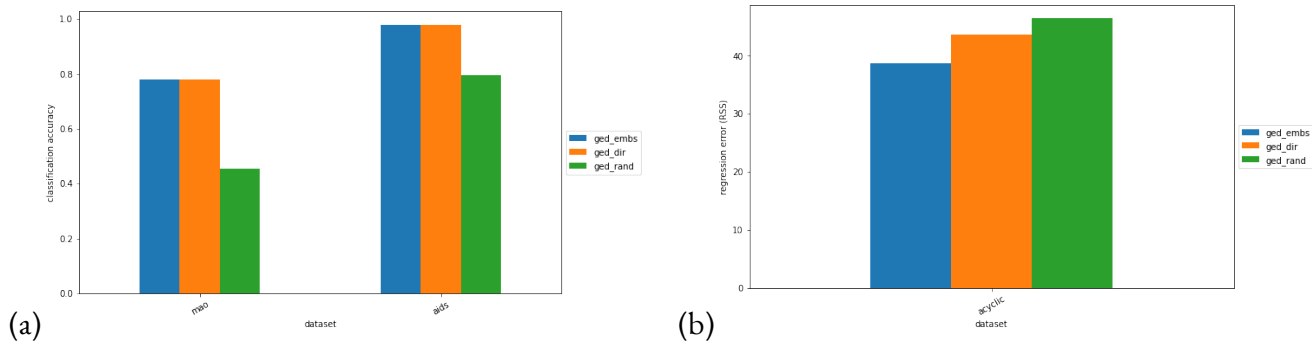


Figure 5.17: (a) Results for MAO and AIDS datasets. (b) Results for the ACYCLIC dataset.

dataset	ged_embs	ged_dir	ged_rand
ptc_o	0.5616438356164384	0.5342465753424658	0.4657534246575342
ptc_1	0.6521739130434783	0.6413043478260869	0.5869565217391305
ptc_2	0.6065573770491803	0.5409836065573771	0.45901639344262296
ptc_3	0.59	0.57	0.54
mao	0.779412	0.779412	0.45588235294117646
aids	0.9793333333333333	0.978	0.796
acyclic	38.627896439574194	43.60018615234525	46.3911403146047

Table 5.4: Actual values for all the experiments using the seven datasets and the three sets of transformation costs.

5.4 DISCUSSION

This chapter continues our global aim to combine graph-reduction, graph-distance, and machine learning techniques to compare molecular structures and reveal inherent relationships among them.

Chapter 3 presented a molecular similarity measure based on GED and ErGs to compare the representation of molecules based on their pharmacophore-type node descriptions. In that chapter, we used the edit costs proposed by Harper *et al.*⁸⁰, fixed values assigned by experts to manage relationships between the different node and edge types. Later, in chapter 4, we aimed to automatically learn the edit costs on nodes and edges through a genetic algorithm, which resulted in an improved version of the algorithm at the expense of a long and heavy optimization process. This chapter uses a faster ML method based on NLP to learn the features and compute relative distances between different nodes and edges in a graph; those relative distances were later used as transformation costs for the GED methodology.

All datasets in this chapter are formatted and publicly available on the GREYC research group's dataset repository, with whom we were working together to develop this section's experiments.

GEDs computed with the embedding-based transformation costs are equal or better than the other two proposed sets of transformation costs. Since embeddings overtake the fixed values, we can hypothesize an intrinsic knowledge relationship between the embeddings and the pharmacophore-type of nodes in the ErGs. Additionally, we show that results do not follow a random distribution behavior; this is achieved by comparing all results with a random pairwise distance between ErGs. Finally, as in the previous chapter, the optimized transfor-

mation costs may hint at the structure-activity relationship in the different activity classes.

If you torture the data long enough, it will confess.

Ronald Coase.

6

General conclusion

This thesis focuses on studying QSAR models to find significant correlations between molecular structure and molecular bioactivity, assuming that structurally similar molecules are likely to have similar properties. This assumption makes molecular similarity methods frequently used to select suitable candidates in the drug discovery industry and in ligand-based virtual screening applications, where the activity of compounds is estimated from the bioactivity based on similar molecules.

In our work, we favored the use of graph reduction, graph theory, and optimization tools. It is not the first time those tools are used in life sciences for molecular analysis, QSAR models, and drug discovery. Chapter 1 presents a state of the art and literature review on molecular similarity and virtual screening in chemoinformatics. Most of the works mentioned in that chapter are based on fingerprints, which are substructure descriptors generating a vector encoding information about the molecule. Fingerprints are particularly suitable for machine learning or statistical analysis, often made up of binary digits (1s and 0s) representing the presence or absence (in some cases, the frequency) of certain substructures or features in the molecule. Fingerprints are fast to compute, use a small space in memory, and are relatively easy to compare. However, experience shows that finding the best type of fingerprint for a specific application (e.g., virtual screening) depends strongly on the data set. Moreover, fingerprints include essential information about the molecule's composition but contain limited information about its structure and shape.

Chapter 2 shows and explains the general materials and methods used throughout this work, emphasizing the datasets, the graph reduction process, the GED methodology, the optimization tools, and the evaluation methods.

Chapter 3 introduces a molecular similarity measure to compare the representation of

molecules as extended reduced graphs utilizing the graph edit distance. The goal of that chapter is to present an alternative to the string edit distance-based and fingerprint-based similarity methods available, including the original one used in the paper on ErGs. The experiments in that chapter used several samples from publicly available datasets and an open-source benchmarking platform. The platform includes, among other features, various screening and statistical tools and provides fully reproducible outcomes. Results show that the GED-based method performed better in 5 out of 6 methods according to the “early recognition” BEDROC values; moreover, GED-based performed better in 4 out of 6 methods according to AUC values. The transformation costs used in that chapter were proposed by Harper *et al.*, fixed values experts assigned to manage relationships between the node and edge types.

Chapter 4 presents an optimization procedure based on a genetic algorithm to automatically learn the edit costs for nodes and edges in several scenarios using various toxicological endpoints. The optimization improves the recognition ratio when classifying molecules represented as ErGs according to their biological activity. A significant improvement is obtained using the learned costs in most experiments for the 127 targets present in the six datasets. All datasets used are publicly available and formatted as part of a benchmarking platform. Results show that the learned edit costs performed as good or better in most of the targets present in the six datasets, compared to the edit costs proposed by Harper *et al.*

Additionally to the better performance, the learned edit costs may also give some hints about the structure-activity relationship in the activity classes. This method’s downside is the need for a long and computationally heavy optimization process, sometimes taking several hours to be completed.

Finally, chapter 5 uses a methodology inspired by natural language processing techniques,

the skip-gram model, to infer the transformation costs much faster than the previous chapter. This process analyzes relationships between node features as words in a text and computes relative distances between different nodes and edges in a graph. The relative distances are then used as transformation costs for the GED methodology. All datasets in this chapter are formatted and publicly available similarly to those in the previous chapters.

GEDs computed with the embedding-based transformation costs are equal or better than the other sets of transformation costs proposed for evaluation. Results obtained using the embeddings do not follow a random distribution behavior; this is confirmed by comparing all results with a random pairwise distance between ErGs. Furthermore, the transformation costs obtained from the embeddings may hint at the structure-activity relationship in the different activity classes, which could be understood as an intrinsic knowledge relationship between the embeddings and the pharmacophore-type of nodes in the ErGs.

From a global view, this thesis aims to combine graph-reduction, graph-distance, and machine learning techniques to compare molecular structures and reveal inherent relationships among them to be used in LBVS applications. The results proved promising and consistent, with our methods performing better than others based on reduced graphs or array representations. We use various screening and statistical tools, including the ligand-based virtual screening benchmarking platform and the RDKit open-source cheminformatics software.

Overall, it is shown that extended reduced graphs, graph edit distance, and optimization tools are a suitable combination of methods. This combination has numerous applications and identifies bioactivity similarities in structurally diverse groups of molecules, thus suggesting that they might be beneficial in scaffold-hopping applications. Moreover, from the results in chapters 4 and 5, we can conclude that optimizing the transformation costs performed on

specific activity classes can improve retrieval accuracy and yield some property hints about the underlying structure-activity relationship among molecules.

6.1 PROSPECTS AND PERSPECTIVES

After performing diverse experiments with tools coming from several fields, this work opens different perspectives to go further in the attempt to solve the problems related to molecular bioactivity prediction:

First, reduced graphs try to generalize the molecular structure while retaining the topology information. They proved helpful in many situations, but the reduction scheme is crucial depending on the type of application. For instance, in applications searching for Markush structures, a simple ring/non-ring scheme might suffice; on the other hand, applications to identify the structure-activity relationship, more complex schemes are required to identify the pharmacophoric groups. We conceive as an option the use of an optimization method to learn the sub-structures reduced in each specific scheme according to the type of molecular property to be predicted.

Second, as mentioned in chapter 3, this work does not envisage the use of stereochemical information for molecules, and it constitutes essential information for specific chemical properties. It would be interesting to include the 3D location of each atom as a quantitative label and then use a reference of the neighbors' location as an extension of the costs in the graph edit distance methodology.

Third, life sciences are advancing at a rate probably faster than most branches of science, and the same can be said for machine learning; Combining those two research fields can yield outstanding outcomes. Thus, similarly to how we used a shallow, two-layer neural network

in chapter 5 as the skip-gram implementation to infer the transformation costs of the GED. It might be a good idea to use a Graph neural network combined with an objective function based on the GED as a tool to infer better transformation costs yielding improved property hints about the structure-activity relationship among molecules.

References

- [1] Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., & Smola, A. J. (2013). Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web* (pp. 37–48).
- [2] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3), 175–185.
- [3] Anstreicher, K. M. (2003). Recent advances in the solution of quadratic assignment problems. *Mathematical Programming*, 97(1-2), 27–42.
- [4] Asgari, E. & Mofrad, M. R. (2015). Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11), e0141287.
- [5] Bajorath, J. (2001). Selected concepts and investigations in compound classification, molecular descriptor analysis, and virtual screening. *J. Chem. Inf. Comput. Sci.*, 41(2), 233–245.
- [6] Baldi, P. & Pollastri, G. (2003). The principled design of large-scale recursive neural network architectures—dag-rnns and the protein structure prediction problem. *Journal of Machine Learning Research*, 4(Sep), 575–602.
- [7] Baranski, A. (2012). The atomic mass unit, the avogadro constant, and the mole: a way to understanding. *Journal of Chemical Education*, 89(1), 97–102.
- [8] Barker, E. J., Buttar, D., Cosgrove, D. A., Gardiner, E. J., Kitts, P., Willett, P., & Gillet, V. J. (2006a). Scaffold hopping using clique detection applied to reduced graphs. *J. Chem. Inf. Model.*, 46(2), 503–511.
- [9] Barker, E. J., Buttar, D., Cosgrove, D. A., Gardiner, E. J., Kitts, P., Willett, P., & Gillet, V. J. (2006b). Scaffold hopping using clique detection applied to reduced graphs. *Journal of chemical information and modeling*, 46(2), 503–511.

- [10] Barker, E. J., Gardiner, E. J., Gillet, V. J., Kitts, P., & Morris, J. (2003). Further development of reduced graphs for identifying bioactive compounds. *J. Chem. Inf. Comput. Sci.*, 43(2), 346–356.
- [11] Barnard, J. M. (1993). Substructure searching methods: Old and new. *J. Chem. Inf. Comput. Sci.*, 33(4), 532–538.
- [12] Bender, A. & Glen, R. C. (2004). Molecular similarity: a key technique in mol. inf. *Org. Biomol. Chem.*, 2(22), 3204–3218.
- [13] Beno, B. R. & Mason, J. S. (2001). The design of combinatorial libraries using properties and 3d pharmacophore fingerprints. *Drug Discovery Today*, 6(5), 251–258.
- [14] Berretti, S., Del Bimbo, A., & Vicario, E. (2001). Efficient matching and indexing of graph models in content-based retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10), 1089–1105.
- [15] Bhagat, S., Cormode, G., & Muthukrishnan, S. (2011). Node classification in social networks. In *Social network data analytics* (pp. 115–148). Springer.
- [16] Biggs, N., Lloyd, E. K., & Wilson, R. J. (1986). *Graph Theory, 1736-1936*. Oxford University Press.
- [17] Birchall, K. & Gillet, V. J. (2010). Reduced graphs and their applications in chemoinformatics. In *Chemoinformatics and Computational Chemical Biology* (pp. 197–212). Springer.
- [18] Birchall, K., Gillet, V. J., Harper, G., & Pickett, S. D. (2006). Training similarity measures for specific activities: application to reduced graphs. *J. Chem. Inf. Model.*, 46(2), 577–586.
- [19] Bleicher, K. H., Böhm, H.-J., Müller, K., & Alanine, A. I. (2003). Hit and lead generation: beyond high-throughput screening. *Nature reviews Drug discovery*, 2(5), 369–378.
- [20] Blumenthal, D. B. & Gamper, J. (2018). On the exact computation of the graph edit distance. *Pattern Recognit. Lett.*, 0(0), 1–12.
- [21] Bondi, A. v. (1964). van der waals volumes and radii. *The Journal of physical chemistry*, 68(3), 441–451.

- [22] Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., & Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1), i47–i56.
- [23] Böttcher, C. J. F. & Bordewijk, P. (1978). *Theory of electric polarization*, volume 2. Elsevier Science Limited.
- [24] Bourgeois, F. & Lassalle, J.-C. (1971). An extension of the munkres algorithm for the assignment problem to rectangular matrices. *Communications of the ACM*, 14(12), 802–804.
- [25] Brants, T., Popat, A. C., Xu, P., Och, F. J., & Dean, J. (2007). Large language models in machine translation.
- [26] Brown, N. & Jacoby, E. (2006). On scaffolds and hopping in medicinal chemistry. *Mini reviews in medicinal chemistry*, 6(11), 1217–1229.
- [27] Brown, R. D. & Martin, Y. C. (1996). Use of structure- activity data to compare structure-based clustering methods and descriptors for use in compound selection. *Journal of chemical information and computer sciences*, 36(3), 572–584.
- [28] Brown, R. D. & Martin, Y. C. (1997). The information content of 2d and 3d structural descriptors relevant to ligand-receptor binding. *Journal of Chemical Information and Computer Sciences*, 37(1), 1–9.
- [29] Bruice, P. Y. (2003). Organic chemistry 4th edition.
- [30] Bunke, H. (1998). Error-tolerant graph matching: a formal framework and algorithms. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)* (pp. 1–14): Springer.
- [31] Bunke, H. & Allermann, G. (1983). Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4), 245–253.
- [32] Burkard, R., Dell’Amico, M., & Martello, S. (2012). *Assignment problems: revised reprint*. SIAM.
- [33] Butler, K. T., Davies, D. W., Cartwright, H., Isayev, O., & Walsh, A. (2018). Machine learning for molecular and materials science. *Nature*, 559(7715), 547–555.

- [34] Caetano, T. S., McAuley, J. J., Cheng, L., Le, Q. V., & Smola, A. J. (2009). Learning graph matching. *IEEE transactions on pattern analysis and machine intelligence*, 31(6), 1048–1058.
- [35] Cancho, R. F. I. & Solé, R. V. (2001). The small world of human language. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1482), 2261–2265.
- [36] Carhart, R. E., Smith, D. H., & Venkataraghavan, R. (1985). Atom pairs as molecular features in structure-activity studies: definition and applications. *J. Chem. Inf. Comput. Sci.*, 25(2), 64–73.
- [37] Cavasotto, C. N., Orry, W., & Andrew, J. (2007). Ligand docking and structure-based virtual screening in drug discovery. *Current topics in medicinal chemistry*, 7(10), 1006–1014.
- [38] Cereto-Massagué, A., Ojeda, M. J., Valls, C., Mulero, M., Garcia-Vallvé, S., & Pujadas, G. (2015). Molecular fingerprint similarity search in virtual screening. *Methods*, 71, 58–63.
- [39] Chen, Y.-Y., Lin, Y.-H., Kung, C.-C., Chung, M.-H., Yen, I., et al. (2019). Design and implementation of cloud analytics-assisted smart power meters considering advanced artificial intelligence as edge analytics in demand-side management for smart homes. *Sensors*, 19(9), 2047.
- [40] Cherqaoui, D., Villemin, D., Mesbah, A., Cense, J.-M., & Kvasnicka, V. (1994). Use of a neural network to determine the normal boiling points of acyclic ethers, peroxides, acetals and their sulfur analogues. *Journal of the Chemical Society, Faraday Transactions*, 90(14), 2015–2019.
- [41] Comer, J. & Tam, K. (2001). Lipophilicity profiles: theory and measurement.
- [42] Conte, D., Foggia, P., Sansone, C., & Vento, M. (2004). Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03), 265–298.
- [43] Cour, T., Srinivasan, P., & Shi, J. (2006). Balanced graph matching. *Advances in neural information processing systems*, 19, 313–320.
- [44] Cui, P., Wang, X., Pei, J., & Zhu, W. (2018). A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, 31(5), 833–852.

- [45] Daylight (1997). Daylight chemical information systems. <https://www.daylight.com/>. Accessed: 2019-12-01.
- [46] DeKock, R. L. & Gray, H. B. (1989). *Chemical structure and bonding*. University Science Books.
- [47] Ding, C. H., He, X., Zha, H., Gu, M., & Simon, H. D. (2001). A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings 2001 IEEE international conference on data mining* (pp. 107–114): IEEE.
- [48] Dror, O., Schneidman-Duhovny, D., Inbar, Y., Nussinov, R., & Wolfson, H. J. (2009). Novel approach for efficient pharmacophore-based virtual screening: method and applications. *Journal of chemical information and modeling*, 49(10), 2333–2343.
- [49] Du, D.-Z., Pardalos, P. M., & Wu, W. (2009). History of optimization.
- [50] Dudek, A. Z., Arodz, T., & Gálvez, J. (2006). Computational methods in developing quantitative structure-activity relationships (qsar): a review. *Combinatorial chemistry & high throughput screening*, 9(3), 213–228.
- [51] Durant, J. L., Leland, B. A., Henry, D. R., & Nourse, J. G. (2002). Reoptimization of mdl keys for use in drug discovery. *Journal of chemical information and computer sciences*, 42(6), 1273–1280.
- [52] Edmonds, J. & Karp, R. M. (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2), 248–264.
- [53] Elton, D. C., Boukouvalas, Z., Fuge, M. D., & Chung, P. W. (2019). Deep learning for molecular design—a review of the state of the art. *Molecular Systems Design & Engineering*, 4(4), 828–849.
- [54] Eshera, M. A. & Fu, K.-S. (1984). A graph distance measure for image analysis. *IEEE transactions on systems, man, and cybernetics*, (3), 398–408.
- [55] Fisanick, W., Lipkus, A. H., & Rusinko, A. (1994). Similarity Searching on CAS Registry Substances. 2. 2D Structural Similarity. *J. Chem. Inf. Comput. Sci.*, 34(1), 130–140.
- [56] Foggia, P., Percannella, G., & Vento, M. (2014). Graph matching and learning in pattern recognition in the last 10 years. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(01), 1450001.

- [57] Freeman, L. C. (2000). Visualizing social networks. *Journal of social structure*, 1(1), 4.
- [58] Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J. Am. Stat. Assoc.*, 32(200), 675–701.
- [59] Fröhlich, H., Wegner, J. K., Sieker, F., & Zell, A. (2005). Optimal assignment kernels for attributed molecular graphs. In *Proceedings of the 22nd international conference on Machine learning* (pp. 225–232).
- [60] Gao, X., Xiao, B., Tao, D., & Li, X. (2010). A survey of graph edit distance. *Pattern Analysis and applications*, 13(1), 113–129.
- [61] Garcia-Hernandez, C., Fernández, A., & Serratos, F. (2019). Ligand-based virtual screening using graph edit distance as molecular similarity measure. *Journal of chemical information and modeling*.
- [62] Gatica, E. A. & Cavasotto, C. N. (2011). Ligand and decoy sets for docking to G protein-coupled receptors. *J. Chem. Inf. Model.*, 52(1), 1–6.
- [63] Gaüzere, B., Brun, L., & Villemin, D. (2012). Two new graphs kernels in chemoinformatics. *Pattern Recognition Letters*, 33(15), 2038–2047.
- [64] Gavezzotti, A. (1983). The calculation of molecular volumes and the use of volume analysis in the investigation of structured media and of solid-state organic reactivity. *Journal of the American Chemical Society*, 105(16), 5220–5225.
- [65] Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.
- [66] Gershell, L. J. & Atkins, J. H. (2003). A brief history of novel drug discovery technologies. *Nature Reviews Drug Discovery*, 2(4), 321–327.
- [67] Gillet, V. J., Downs, G. M., Holliday, J. D., Lynch, M. F., & Dethlefsen, W. (1991). Computer storage and retrieval of generic chemical structures in patents. 13. reduced graph generation. *J. Chem. Inf. Comput. Sci.*, 31(2), 260–270.
- [68] Gillet, V. J., Downs, G. M., Ling, A., Lynch, M. F., Venkataram, P., Wood, J. V., & Dethlefsen, W. (1987). Computer storage and retrieval of generic chemical structures in patents. 8. reduced chemical graphs and their applications in generic chemical structure retrieval. *Journal of chemical information and computer sciences*, 27(3), 126–137.
- [69] Gillet, V. J., Willett, P., & Bradshaw, J. (2003). Similarity searching using reduced graphs. *J. Chem. Inf. Comput. Sci.*, 43(2), 338–345.

- [70] Gobbi, A. & Poppinger, D. (1998). Genetic optimization of combinatorial libraries. *Biotechnology and bioengineering*, 61(1), 47–54.
- [71] Goldberg, D. E. & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning*, 3(2), 95–99.
- [72] Goldberg, Y. & Levy, O. (2014). word2vec explained: deriving mikolov et al.’s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.
- [73] Golub, G. H. & Reinsch, C. (1971). Singular value decomposition and least squares solutions. In *Linear Algebra* (pp. 134–151). Springer.
- [74] Gori, M., Monfardini, G., & Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2 (pp. 729–734): IEEE.
- [75] Goyal, P. & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151, 78–94.
- [76] GREYC, online (2021). Greyc research group repository. <http://iapr-tcl5.greyc.fr/links.html#Benchmarking%20and%20data%20sets>. Accessed: 2021-02-15.
- [77] Grover, A. & Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 855–864).
- [78] Güner, O. F. (2000). *Pharmacophore perception, development, and use in drug design*, volume 2. Internat’l University Line.
- [79] Harchaoui, Z. & Bach, F. (2007). Image classification with segmentation graph kernels. In *2007 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8): IEEE.
- [80] Harper, G., Bravi, G. S., Pickett, S. D., Hussain, J., & Green, D. V. S. (2004). The reduced graph descriptor in virtual screening and data-driven clustering of high-throughput screening data. *J. Chem. Inf. Comput. Sci.*, 44(6), 2145–2156.
- [81] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2), 100–107.
- [82] Heikamp, K. & Bajorath, J. (2013). The future of virtual compound screening. *Chem. Biol. Drug Des.*, 81(1), 33–40.

- [83] Jaeger, S., Fulle, S., & Turk, S. (2018). Mol2vec: unsupervised machine learning approach with chemical intuition. *Journal of chemical information and modeling*, 58(1), 27–35.
- [84] Johnson, M. A. & Maggiora, G. M. (1990). *Concepts and applications of molecular similarity*. Wiley.
- [85] Kang, Y. K. & Jhon, M. S. (1982). Additivity of atomic static polarizabilities and dispersion coefficients. *Theoretica chimica acta*, 61(1), 41–48.
- [86] Kearsley, S. K., Sallamack, S., Fluder, E. M., Andose, J. D., Mosley, R. T., & Sheridan, R. P. (1996). Chemical similarity using physiochemical property descriptors. *J. Chem. Inf. Comput. Sci.*, 36(1), 118–127.
- [87] Kenny, P. W. & Montanari, C. A. (2013). Inflation of correlation in the pursuit of drug-likeness. *J. Comput.-Aided Mol. Des.*, 27(1), 1–13.
- [88] Kirchmair, J., Distinto, S., Markt, P., Schuster, D., Spitzer, G. M., Liedl, K. R., & Wolber, G. (2009). How to optimize shape-based virtual screening: choosing the right query and including chemical information. *J. Chem. Inf. Model.*, 49(3), 678–692.
- [89] Kirkpatrick, P. & Ellis, C. (2004). Chemical space.
- [90] Koza, J. R., Bennett, F. H., Andre, D., & Keane, M. A. (1996). Automated design of both the topology and sizing of analog electrical circuits using genetic programming. In *Artificial Intelligence in Design'96* (pp. 151–170). Springer.
- [91] Kroemer, R. T. (2007). Structure-based drug design: docking and scoring. *Current protein and peptide science*, 8(4), 312–328.
- [92] Kubinyi, H., Mannhold, R., & Timmerman, H. (2008). *Virtual screening for bioactive molecules*, volume 10. John Wiley & Sons.
- [93] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2), 83–97.
- [94] Lagarde, N., Ben Nasr, N., Jérémie, A., Guillemain, H., Laville, V., Labib, T., Zagury, J.-F., & Montes, M. (2014). Nrlist bdb, the manually curated nuclear receptors ligands and structures benchmarking database. *J. Med. Chem.*, 57(7), 3117–3125.
- [95] Lajiness, M. (1990). Molecular similarity-based methods for selecting compounds for screening. In *Computational chemical graph theory* (pp. 299–316): Nova Science Publishers, Inc.

- [96] Lee, B. & Richards, F. M. (1971). The interpretation of protein structures: estimation of static accessibility. *Journal of molecular biology*, 55(3), 379–IN4.
- [97] Liben-Nowell, D. & Kleinberg, J. (2007). The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7), 1019–1031.
- [98] Livingstone, D. J. (2000). The characterization of chemical structures using molecular properties. a survey. *J. Chem. Inf. Comput. Sci.*, 40(2), 195–209.
- [99] Maaten, L. v. d. & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov), 2579–2605.
- [100] Macarron, R., Banks, M. N., Bojanic, D., Burns, D. J., Cirovic, D. A., Garyantes, T., Green, D. V., Hertzberg, R. P., Janzen, W. P., Paslay, J. W., et al. (2011). Impact of high-throughput screening in biomedical research. *Nature reviews Drug discovery*, 10(3), 188–195.
- [101] MACCS, K. (1984). Mdl information systems. *Inc.: San Leandro, CA*.
- [102] Martin, Y. C. (2010). *Quantitative drug design: a critical introduction*. CRC Press.
- [103] Matter, H. (1997). Selecting optimally diverse compounds from structure databases: a validation study of two-dimensional and three-dimensional molecular descriptors. *Journal of Medicinal Chemistry*, 40(8), 1219–1229.
- [104] Mayr, A., Klambauer, G., Unterthiner, T., & Hochreiter, S. (2016). Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3, 80.
- [105] McGregor, M. J. & Pallai, P. V. (1997). Clustering of large databases of compounds: using the mdl “keys” as structural descriptors. *J. Chem. Inf. Comput. Sci.*, 37(3), 443–448.
- [106] McNaught, A. D., Wilkinson, A., et al. (1997). *Compendium of chemical terminology*, volume 1669. Blackwell Science Oxford.
- [107] Mead, A. (1992). Review of the development of multidimensional scaling methods. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 41(1), 27–39.
- [108] Melville, J. L., Burke, E. K., & Hirst, J. D. (2009). Machine learning in virtual screening. *Comb. Chem. High Throughput Screening*, 12(4), 332–343.

- [109] Menard, P. R., Mason, J. S., Morize, I., & Bauerschmidt, S. (1998). Chemistry space metrics in diversity analysis, library design, and compound selection. *J. Chem. Inf. Comput. Sci.*, 38(6), 1204–1213.
- [110] Merget, B., Turk, S., Eid, S., Rippmann, F., & Fulle, S. (2017). Profiling prediction of kinase inhibitors: toward the virtual assay. *Journal of medicinal chemistry*, 60(1), 474–485.
- [111] Messmer, B. T. & Bunke, H. (1998). A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE transactions on pattern analysis and machine intelligence*, 20(5), 493–504.
- [112] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- [113] Mikolov, T., Deoras, A., Povey, D., Burget, L., & Černocký, J. (2011). Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding* (pp. 196–201): IEEE.
- [114] Miller, K. J. (1990). Additivity methods in molecular polarizability. *Journal of the American Chemical Society*, 112(23), 8533–8542.
- [115] Mitchell, M. (1996). An introduction to genetic algorithms mit press. *Cambridge, Massachusetts. London, England*, 1996.
- [116] Mnih, A. & Hinton, G. E. (2008). A scalable hierarchical distributed language model. *Advances in neural information processing systems*, 21, 1081–1088.
- [117] Morgan, H. L. (1965). The generation of a unique machine description for chemical structures—a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, 5(2), 107–113.
- [118] Morin, F. & Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Aistats*, volume 5 (pp. 246–252): Citeseer.
- [119] Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1), 32–38.
- [120] Myers, R., Wison, R., & Hancock, E. R. (2000). Bayesian graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6), 628–635.

- [121] Mysinger, M. M., Carchia, M., Irwin, J. J., & Shoichet, B. K. (2012). Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking. *J. Med. Chem.*, 55(14), 6582–6594.
- [122] Nikolova, N. & Jaworska, J. (2003). Approaches to measure chemical similarity—a review. *QSAR & Combinatorial Science*, 22(9-10), 1006–1026.
- [123] Nilakantan, R., Bauman, N., Dixon, J. S., & Venkataraghavan, R. (1987). Topological torsion: a new molecular descriptor for sar applications. comparison with other descriptors. *J. Chem. Inf. Comput. Sci.*, 27(2), 82–85.
- [124] No, K. T., Cho, K. H., Jhon, M. S., & Scheraga, H. A. (1993). An empirical method to calculate average molecular polarizabilities from the dependence of effective atomic polarizabilities on net atomic charge. *Journal of the American Chemical Society*, 115(5), 2005–2014.
- [125] Noé, F., Tkatchenko, A., Müller, K.-R., & Clementi, C. (2020). Machine learning for molecular simulation. *Annual review of physical chemistry*, 71, 361–390.
- [126] Oei, R. W., Hou, G., Liu, F., Zhong, J., Zhang, J., An, Z., Xu, L., & Yang, Y. (2019). Convolutional neural network for cell classification using microscope images of intracellular actin networks. *PloS one*, 14(3), e0213626.
- [127] O’Boyle, N. M. & Sayle, R. A. (2016). Comparing structural fingerprints using a literature-based similarity benchmark. *Journal of cheminformatics*, 8(1), 1–14.
- [128] Paul, S. M., Mytelka, D. S., Dunwiddie, C. T., Persinger, C. C., Munos, B. H., Lindborg, S. R., & Schacht, A. L. (2010). How to improve r&d productivity: the pharmaceutical industry’s grand challenge. *Nature reviews Drug discovery*, 9(3), 203–214.
- [129] Pearlman, R. S. & Smith, K. M. (1999). Metric validation and the receptor-relevant subspace concept. *J. Chem. Inf. Comput. Sci.*, 39(1), 28–35.
- [130] Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 701–710).
- [131] Polishchuk, P. G., Madzhidov, T. I., & Varnek, A. (2013). Estimation of the size of drug-like chemical space based on gdb-17 data. *Journal of computer-aided molecular design*, 27(8), 675–679.

- [132] RDKit, online (2013). Rdkit: Cheminformatics and machine learning software. <http://www.rdkit.org>.
- [133] Rehurek, R. & Sojka, P. (2010). Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks*: Citeseer.
- [134] Reymond, J.-L. (2015). The chemical space project. *Accounts of chemical research*, 48(3), 722–730.
- [135] Riesen, K. & Bunke, H. (2008). Iam graph database repository for graph based pattern recognition and machine learning. In *Joint LAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)* (pp. 287–297): Springer.
- [136] Riesen, K. & Bunke, H. (2009). Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision computing*, 27(7), 950–959.
- [137] Riesen, K., Neuhaus, M., & Bunke, H. (2007). Bipartite graph matching for computing the edit distance of graphs. In *International Workshop on Graph-Based Representations in Pattern Recognition* (pp. 1–12): Springer.
- [138] Riniker, S., Fechner, N., & Landrum, G. A. (2013). Heterogeneous classifier fusion for ligand-based virtual screening: or, how decision making by committee can be a good thing. *Journal of chemical information and modeling*, 53(11), 2829–2836.
- [139] Riniker, S. & Landrum, G. A. (2013). Open-source platform to benchmark fingerprints for ligand-based virtual screening. *J. Cheminf.*, 5(1), 26.
- [140] Rogers, D. & Hahn, M. (2010). Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5), 742–754.
- [141] Rohrer, S. G. & Baumann, K. (2009). Maximum unbiased validation (muv) data sets for virtual screening based on pubchem bioactivity data. *J. Chem. Inf. Model.*, 49(2), 169–184.
- [142] Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.
- [143] Rosasco, L., Vito, E. D., Caponnetto, A., Piana, M., & Verri, A. (2004). Are loss functions all the same? *Neural Computation*, 16(5), 1063–1076.

- [144] Sanders, M. P., Barbosa, A. J., Zarzycka, B., Nicolaes, G. A., Klomp, J. P., de Vlieg, J., & Del Rio, A. (2012). Comparative analysis of pharmacophore screening tools. *J. Chem. Inf. Model.*, 52(6), 1607–1620.
- [145] Sanfeliu, A. & Fu, K.-S. (1983). A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, (3), 353–362.
- [146] Santacruz, P. & Serratos, F. (2018). Error-tolerant graph matching in linear computational cost using an initial small partial matching. *Pattern Recognit. Lett.*, 0(0), 1–10.
- [147] Schneider, G., Clément-Chomienne, O., Hilfiger, L., Schneider, P., Kirsch, S., Böhm, H.-J., & Neidhart, W. (2000). Virtual screening for bioactive molecules by evolutionary de novo design. *Angew. Chem. Int. Ed.*, 39(22), 4130–4133.
- [148] Schnur, D. (1999). Design and diversity analysis of large combinatorial libraries using cell-based methods. *J. Chem. Inf. Comput. Sci.*, 39(1), 36–45.
- [149] Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Židek, A., Nelson, A. W., Bridgland, A., et al. (2019). Protein structure prediction using multiple deep neural networks in the 13th critical assessment of protein structure prediction (caspr13). *Proteins: Structure, Function, and Bioinformatics*, 87(12), 1141–1148.
- [150] Serratos, F. (2014). Fast computation of bipartite graph matching. *Pattern Recognit. Lett.*, 45, 244–250.
- [151] Serratos, F. (2019). Graph edit distance: restrictions to be a metric. *Pattern Recognition*, 90, 250–256.
- [152] Serratos, F., Alquézar, R., & Sanfeliu, A. (2000). Efficient algorithms for matching attributed graphs and function-described graphs. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 2 (pp. 867–872): IEEE.
- [153] Sheridan, R. P. & Kearsley, S. K. (2002). Why do we need so many chemical similarity search methods? *Drug Discovery Today*, 7(17), 903–911.
- [154] Shoichet, B. K. (2004). Virtual screening of chemical libraries. *Nature*, 432(7019), 862–865.

- [155] Skoda, P. & Hoksza, D. (2017). Benchmarking platform for ligand-based virtual screening. In *Proceedings - 2016 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2016* (pp. 1220–1227).
- [156] Snyman, J. A. (2005). *Practical mathematical optimization*. Springer.
- [157] Solé-Ribalta, A., Serratos, F., & Sanfeliu, A. (2012). On the graph edit distance cost: properties and applications. *International Journal of Pattern Recognition and Artificial Intelligence*, 26(05), 1260004.
- [158] Sorgenfrei, F. A., Fulle, S., & Merget, B. (2018). Kinome-wide profiling prediction of small molecules. *ChemMedChem*, 13(6), 495–499.
- [159] Stahura, F. L. & Bajorath, J. (2004). Virtual screening methods that complement hts. *Combinatorial chemistry & high throughput screening*, 7(4), 259–269.
- [160] Stauffer, M., Tschachtli, T., Fischer, A., & Riesen, K. (2017). A survey on applications of bipartite graph edit distance. In *International Workshop on Graph-Based Representations in Pattern Recognition* (pp. 242–252): Springer.
- [161] Stiefl, N., Watson, I. A., Baumann, K., & Zaliani, A. (2006). Erg: 2d pharmacophore descriptions for scaffold hopping. *J. Chem. Inf. Model.*, 46(1), 208–220.
- [162] Storn, R. & Price, K. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4), 341–359.
- [163] Sun, H. (2008). Pharmacophore-based virtual screening. *Curr. Med. Chem.*, 15(10), 1018–1024.
- [164] Takahashi, Y., Sukekawa, M., & Sasaki, S. (1992). Automatic identification of molecular similarity using reduced-graph representation of chemical structure. *J. Chem. Inf. Model.*, 32(6), 639–643.
- [165] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web* (pp. 1067–1077).
- [166] Theocharidis, A., Van Dongen, S., Enright, A. J., & Freeman, T. C. (2009). Network visualization and analysis of gene expression data using biolayout express 3d. *Nature protocols*, 4(10), 1535.

- [167] Todeschini, R. & Consonni, V. (2008). *Handbook of molecular descriptors*, volume 11. John Wiley & Sons.
- [168] Todeschini, R. & Consonni, V. (2009). *Molecular descriptors for chemoinformatics: volume I: alphabetical listing/volume II: appendices, references*, volume 41. John Wiley & Sons.
- [169] Toivonen, H., Srinivasan, A., King, R. D., Kramer, S., & Helma, C. (2003). Statistical evaluation of the predictive toxicology challenge 2000–2001. *Bioinformatics*, 19(10), 1183–1193.
- [170] Truchon, J.-F. & Bayly, C. I. (2007). Evaluating virtual screening methods: good and bad metrics for the “early recognition” problem. *J. Chem. Inf. Model.*, 47(2), 488–508.
- [171] Vento, M. (2013). A one hour trip in the world of graphs, looking at the papers of the last ten years. In *International Workshop on Graph-Based Representations in Pattern Recognition* (pp. 1–10): Springer.
- [172] Vento, M. (2015). A long trip in the charming world of graphs for pattern recognition. *Pattern Recognition*, 48(2), 291–301.
- [173] Vikhar, P. A. (2016). Evolutionary algorithms: A critical review and its future prospects. In *2016 International conference on global trends in signal processing, information computing and communication (ICGTSPICC)* (pp. 261–265): IEEE.
- [174] Wagnière, G. H. (2007). *On chirality and the universal asymmetry: reflections on image and mirror image*. John Wiley & Sons.
- [175] Wallis, W. D., Shoubridge, P., Kraetz, M., & Ray, D. (2001). Graph distances using graph union. *Pattern Recognition Letters*, 22(6-7), 701–704.
- [176] Wang, D., Cui, P., & Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 1225–1234).
- [177] Wang, Y., Liu, Z., Muzzio, F., Drazer, G., & Callegari, G. (2018). A drop penetration method to measure powder blend wettability. *International journal of pharmaceutics*, 538(1-2), 112–118.
- [178] Wevi, online (2021). Greyc research group repository. <http://bit.ly/wevi-online>. Accessed: 2021-04-24.

- [179] Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biom. Bull.*, 1(6), 80–83.
- [180] Willett, J. (1987). *Similarity and clustering in chemical information systems*. John Wiley & Sons, Inc.
- [181] Willett, P. (2004). Evaluation of molecular similarity and mol. diversity methods using biological activity data. In *Chemoinformatics* (pp. 51–63). Springer.
- [182] Willett, P., Barnard, J. M., & Downs, G. M. (1998). Chemical similarity searching. *J. Chem. Inf. Comput. Sci.*, 38(6), 983–996.
- [183] Willett, P., Winterman, V., & Bawden, D. (1986). Implementation of nearest-neighbor searching in an online chemical structure search system. *Journal of Chemical Information and Computer Sciences*, 26(1), 36–41.
- [184] Xia, J., Tilahun, E. L., Reid, T.-E., Zhang, L., & Wang, X. S. (2015). Benchmarking methods and data sets for ligand enrichment assessment in virtual screening. *Methods*, 71, 146–157.
- [185] Xue, L. & Bajorath, J. (2000). Molecular descriptors in chemoinformatics, computational combinatorial chemistry, and virtual screening. *Comb. Chem. High Throughput Screening*, 3(5), 363–372.
- [186] Zou, Q., Chen, L., Huang, T., Zhang, Z., & Xu, Y. (2017). Machine learning and graph analytics in computational biomedicine. *Artif. Intell. Medicine*, 83, 1.



UNIVERSITAT
ROVIRA i VIRGILI