UNIVERSITY OF CALIFORNIA
RIVERSIDE

Novel Structure Similarity-Based Methods for Identifying Drug-Like Compounds

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Yi Qun Cao

August 2010

Acknowledgements

I would like to express my gratitude to my thesis committee Dr. Thomas Girke, Dr. Tao Jiang and Dr. Stefano Lonardi. Especially I would like to thank my supervisor Dr. Thomas Girke for his guidance and continuous support during my graduate studies.

I am also very thankful to my CS and ChemGen IGERT classmates for their help to familiarize myself with the student life. They were always there to assist me with any personal or academic issue. Especially I want to thank the members of Dr. Girke's lab for their help in many aspects of my research: Li-Chang Cheng, Josh Lauricha, Alex Levchuk, Kevin Horan, Anna Charisi, Julian Krause, Tyler Backman and Rebecca Sun.

There are not enough words to describe my appreciation and my love for my parents and my wife Lei Wang. They are my family who stood next to me during all the difficult moments and shared with me all the happy ones. They are always going to be a significant part of my life.

Finally, I would like to thank the ChemGen IGERT program for supporting my graduate studies, expecially Dr. Julia Bailey-Serres, Sean Cutler and Natasha Raikhel.

ABSTRACT OF THE DISSERTATION

Novel Structure Similarity-Based Methods for Identifying Drug-Like Compounds

by

Yi Qun Cao

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, August 2010
Dr. Tao Jiang, Chairperson

The prediction of biologically active compounds is of great importance for high-throughput screening (HTS) approaches in drug discovery and chemical genomics. Many computational methods in this area focus on measuring the structural similarities between chemical structures. However, traditional similarity measures are often too rigid or they consider only global similarities between structures. This study introduces two new alternative search approaches that overcome most of these limitations. First, the maximum common substructure (MCS) approach provides a more promising and flexible alternative for predicting bioactive compounds. A new backtracking algorithm for MCS is proposed here and compared to global similarity measurements. Our algorithm provides high flexibility in the matching process and is very efficient in identifying local structural similarities. To apply the MCS-based similarity measure in predictive models of biological activity of compounds, the concept of basis compounds is introduced to enable researchers to easily combine the MCS-based and traditional similarity measures with modern machine learning techniques. Our experiments on real compound datasets demonstrate that MCS complements the well-known atom pair descriptor-based similarity measure. By combining these two measures, we propose an SVM-based algorithm for predicting the biological activities of chemical compounds with high specificity and sensitivity.

In similarity search and clustering applications of very large compound sets, most methods are limited in efficiency and scalability and cannot handle today's large compound datasets with several million entries. This is particularly true for MCS-based methods and the computation complexity renders MCS infeasible for large compound dataset. The second main topic of this study addresses this time performance issue by introducing a new method for greatly accelerating similarity search and clustering of very large compound sets using embedding and indexing techniques. The method, which can be used with MCS-based as well as traditional similarity measures, embeds compounds in a high-dimensional Euclidean space and searches this space using an efficient index-aware nearest neighbor search method based on Locality Sensitive Hashing. The method can also be used to accelerate cluster analysis of large compound sets. When applied to similarity search in compound datasets as large as PubChem, we found that the method was 40-200 times faster than sequential search methods, while maintaining comparable recall rates. It also made MCS-based similarity search tractable for large compound datasets. When applied to the clustering of such compound datasets, it helped to reduce the computation time from several months to only a few days.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

One of the fundamental problems in drug discovery and chemical genomics is to identify biologically active compounds with desirable properties in the chemical space of available small molecules. Recent advancements in compound synthesis have greatly expanded the size of the accessible chemical space. However, our capability to screen an ever-increasing set of compound libraries has only been slightly increased in the past years (Dobson, 2004b). Besides, High Throughput Screening (HTS) - the most widely used tool to probe the chemical space - has a less-than-ideal effectiveness that hinders the discovery process of novel drugs.

Because of this situation, there is a burgeoning demand for computational methods to mine the available small molecule space. Digitized compound information such as structures and biological activity information of millions of small molecules are now collected and published in open-access databases, like PubChem (Austin et al., 2004), ChemBank (Seiler et al., 2008), NCI (Ihlenfeldt et al., 2002), ChemMine (Girke et al., 2005a), ChemDB (Chen et al., 2007) and ZINC (Irwin and Shoichet, 2005). Software tools have been developed to facilitate discovery process in areas ranging from drug discovery, chemical biology and chemical genomics to medicinal chemistry (Oprea, 2002; Strausberg and Schreiber, 2003; Savchuk et al., 2004; Haggarty, 2005; Oprea et al., 2007). As an example, *in silico virtual screening* has been used in combination with traditional *in vitro* screening methods to jointly improve the turnaround time and success rate of the drug discovery process (Abt et al., 2001; Engels and Venkatarangan, 2001).

This dissertation addresses several problems in *in silico* virtual screening, and more specifically, *ligand-based* virtual screening. Unlike *receptor-based* virtual screening, which depends on known knowledge about the structure of the protein target and often also the binding domain, ligand-based virtual screening does not require knowledge of the protein target. Instead, it uses structure and activity information of small molecules to conduct drug design studies by screening databases of structures of available or potentially available chemicals.

The fundamental principle behind ligand-based virtual screening is the *similar property principle*, which states that similar compounds have similar properties (Johnson and Maggiora, 1990a). This places the methods to measure *chemical similarity* between chemical compounds as the foundation of all ligand-based virtual screening techniques. A variety of structure similarity search methods are available (reviewed by Willett et al., 1998; Nikolova and Jaworska, 2004). However, almost all of them suffer from one or both of the following two problems:

1. They are often too rigid to identify weak similarity features or too relaxed to identify local similarities between structures or those with large size differences.

2. In addition, it is often very costly to perform similarity searches using these similarity measures, either because of the intrinsic complexity in computing the similarity measures, or because of the need to sequentially compare the query structure to all database structures, or a combination of both factors.

This dissertation investigates Maximum Common Substructure (MCS) as a promising alternative metric for chemical similarity measure and applies it in similarity search and biological activity prediction. Although it is widely accepted that MCS has the advantages of being intuitive, quantitative, flexible and sensitive to structure similarity at a local level, its intractable computational complexity has plagued its applicability and usefulness. This dissertation describes an efficient and flexible algorithm for computing MCS-based chemical similarity. Using this algorithm, it addresses several questions not answered in previous studies. These include a quantitative study of the effectiveness of MCS-based similarity in terms of identifying novel biologically active structures not captured by traditional methods, a framework to utilize MCS-based chemical similarity in building statistical prediction models for biological activities, and a method to combine MCS-based and other similarity measures in such models.

To address the scalability issue of similarity search in chemical databases, this dissertation demonstrates a method to accelerate nearest neighbor search. This method is shown to be applicable to a wide variety of chemical similarity measures including the MCS-based chemical similarity. Due to the intrinsic computational complexity of the MCS problem, the time to perform MCS-based similarity search and cluster analysis is unmanageable in compound databases with several thousand entries. Many chemical similarity measures having highly desirable features also share the same problem thanks to their computationally expensive metric functions. With a large chemical database with millions of entries, almost all chemical similarity measures will become unscalable in similarity search applications and cluster analysis because of the need to sequentially compare the query structure to all database structures. At the cost of not guaranteeing the exact accurate results, we propose a generic acceleration scheme that utilizes a fast and accurate indexing scheme to provide high-quality results with high efficiency. We demonstrate that this index-aware similarity search method scales elegantly with the size of the database, the computational complexity of the underlying similarity metric function, and the intrinsic dimensionality implied by this function. This dissertation also introduces a fast algorithm for clustering large compound datasets by taking advantage of the enhanced similarity search speed.

1.2 Chemical Similarity Concepts

Similar property principle plays an important role in modern approaches of virtual screening, structure-based drug design and property analysis and prediction of chemical compounds.

Because of the importance of this principle in drug discovery research, many similarity measures have been proposed to quantify similarity between chemical structures and to predict their bioactivity potential.

Substructure and *superstructure* relationships are among the most intuitive and most commonly used similarity measures. They consider two structures to be similar if one (the substructure) is fully contained in the other (the superstructure). A structure that is putatively associated with certain properties of interest or certain types of chemical reactions can be used as query to identify in databases all compounds that share this substructure (or superstructure) and possibly its properties or reactions. Despite the intuitiveness, this type of similarity measures has several drawbacks. First, the utilized matching strategy is very rigid and has a high false negative rate. Second, substructure searching is a knowledge-based approach, in which every utilized query substructure needs to be well defined. If this is not the case or a substructure model is not entirely correct, the search results will be of poor quality and of very limited predictive value. Most importantly, this search type does not generate any quantitative similarity measure, which makes it difficult to rank the search results in a meaningful manner.

Structural descriptor-based similarity measures are another type of commonly used approaches in measuring chemical similarity in structural similarity search and bioactivity predictions. Unlike substructure and superstructure methods, structural descriptors represent chemical structures in a way so that their similarity can be easily quantified. Usually, this is achieved by identifying presence and absence of structural features in chemical structures and encoding them using a fixed-size or variable-size vectors. These structural features can be generated from a predefined reference set or directly enumerated from the structures. The resulting representations in the form of sets or vectors, often referred as *structural descriptors* or *molecule fingerprints*, are used in combination with a similarity coefficients such as the *Tanimoto coefficient* to quantify chemical similarity. This type of similarity measure represents a much less stringent and far more flexible approach than substructure or superstructure searches.

Since there is flexibility in choosing structural features in structural descriptor-based similarity measures, many divergent types of descriptors have been proposed and are widely employed in various software applications. Examples of structural descriptor-based methods include the various widely-used fingerprint methods, atom pair (AP), atom sequence (AS) and many others (Carhart et al., 1985; Johnson and Maggiora, 1990b; Dean, 1995; Willett et al., 1998; Chen and Reynolds, 2002; Sheridan and Kearsley, 2002; Girke et al., 2005b). Combined with a wide range of similarity coefficients to choose from, the family of structural descriptor-based similarity measures are highly diverse and cover different levels of complexity and effectiveness. However, due the simplicity of structural features in all methods of this type, they are unable to identify local similarities between structures or those with large size differences. Their false negative rates increase vastly when it comes to the identification of weaker similarities. In addition, the diversity of methods also complicates the design of a generic mechanism to optimize similarity search and cluster analysis applications.

Chapter 2 gives an in-depth overview of several chemical similarity measures used in this study, including substructure, superstructure, atom pair descriptor-based methods, and *PubChem fingerprints*. Several examples are presented to highlight drawbacks of each of these methods.

1.3 Maximum Common Substructure-based Similarity

One of the major contribution of this dissertation is a comprehensive study of MCS as an alternative chemical similarity measure. MCS-based chemical similarity considers 2D topological representations of compound structures. For a pair of chemical compounds, the MCS is the largest substructure that appears in both structures. The chemical similarity value between them is then quantified using the size of the MCS and optionally a similarity coefficient.

Chapter 3 formally introduces the MCS-based chemical similarity and the MCS problem. It reviews several relevant previous studies, and introduces a new MCS algorithm that is designed for flexible matching of chemical structures. This is followed by a quantitative evaluation of the effectiveness of MCS-based similarity measure in similarity search applications and a comparative study of several structural descriptor-based methods. This answers the questions of whether MCS-based chemical similarity provides comparable or better performance in identifying biologically active compounds, and whether using it in addition to traditional methods can help capture novel biologically active compounds that are missed if only traditional methods are used.

1.4 Similarity Measures and Machine Learning Models

Chapter 4 expands the usage of MCS as an alternative chemical similarity measure to building statistical prediction models for biological activities. It demonstrates a generic scheme to easily apply virtually any chemical similarity in machine learning models. This scheme enables new applications of MCS-based and many other similarity measures that are otherwise difficult to be employed in these models. It also allows easy integration of multiple similarity measures in a single model to potentially increase the predictive power.

Most machine learning models require the data to be in a feature space with explicit finite dimensions. Although many traditional chemical similarity measures satisfy this requirement, the MCS-based method does not. Kernel-based methods provide more flexible alternatives, but the computational complexity of the MCS problem renders a combination of such methods and the MCS-based similarity measure is currently impractical for large compound databases.

This chapter introduces a novel concept of *basis compounds* that allows us to easily utilize the MCS-based similarity measure in modern machine learning models with manageable computational complexity. The basis compounds method maps each compound into a vector in an explicit feature space with limited dimensions. MCS-based similarity information is encoded into each vector. Many machine learning methods can then be applied to these vectors without modification to build prediction models. These models implicitly make use of the MCS-based similarity. This is the first report of using MCS-based similarity in predicting biological activity. As a natural extension, it is also demonstrated that multiple complementary similarity measures can be integrated into one *hybrid* predictive model using the basis compounds method.

This chapter ends with a detailed quantitative evaluation of the prediction models based on different similarity measures. The study on hybrid predictive models reconfirm our answer to the question of whether using MCS-based similarity in addition to traditional methods can help capture novel biologically active compounds that are missed if only traditional methods are used.

1.5 High-Fidelity Embedding of Chemical Structures

The basis compound concept is interesting and powerful because it allows to map structure information into vector representations that encode, *losefully*, similarity information based on arbitrary similarity measures. Despite the loss of information, numeric representation brings flexibility and enables many possibilities.

This dissertation extends this idea by introducing *high-fidelity embedding*, a mechanism to derive vector representations in Euclidean space for chemical structures using a chosen similarity measure. This mechanism allows high-fidelity reproduction of similarity values between a pair of chemical structures using the Euclidean distance between the corresponding vector pair. In other words, the distance between any pair of vectors highly resembles the dissimilarity between the pair of the corresponding chemical compounds using the chosen similarity measure. By minimizing the loss of information, high-fidelity embedding introduces a high degree of flexibility and enables many possibilities.

Chapter 5 describes several important existing and potential useful applications of embedding chemical structures into Euclidean space. It also outlines why existing methods fail to achieve high-fidelity embedding of large compound datasets. This is followed by an in-depth description of a new embedding method with focus on high-quality results and scalability. To evaluate the efficiency and embedding accuracy of the method, it is tested using large-size chemical databases and various similarity measures.

1.6 Accelerating Structure Similarity Search and Clustering

Of the many possibilities revealed by high-fidelity embedding, this dissertation is particularly interested in using it in combination with spatial indexing to accelerate structure similarity searching and clustering of compound databases.

While Chapter 3 introduces MCS as a promising alternative chemical similarity measure, in practice it remains challenging to apply MCS-based similarity measure in similarity search and cluster analysis applications due to its intractable computational complexity. This is a problem shared with many other similarity measures in medium size databases with thousands of entries. A majority of available similarity measures would run into similar scalability problems with large databases containing millions of entries.

Chapter 6 describes a novel approach to accomplish fast compound similarity searching and clustering by integrating high-fidelity embedding with a locality sensitive hashing technique. It demonstrates that when the vector representations generated by high-fidelity embedding are indexed properly, similarity searches can be faithfully reproduced by a highly efficient nearest neighbor search in the embedding space. The chapter also extends the benefit of this technique from similarity search to cluster analysis. This acceleration scheme is put to test in building an online high-performance similarity search engine and in clustering a very large compound database of nearly 20 million compounds.

1.7 A Toolkit for Studying Similarity Measures

Several highly-reusable and user- and developer-friendly software packages and online services have been created along the course of this dissertation study. Chapter 7 will briefly introduce the following:

ChemmineR is a software package designed for R programming environment (Cao et al, 2008). It provides support for 2D structural similarity comparisons between compounds and similarity searching against compound databases. Both features use atom pair descriptor for scoring chemical similarity. The package also provides various functions for clustering entire compound libraries and visualizing clustering results and compound structures. All functions and data objects are well integrated into the existing infrastructure of the R environment. This package has been accepted to be distributed as part of bioconductor releases (Gentleman et al., 2004).

`libap` provides a highly efficient implementation of the most important functions in ChemmineR. Similar to ChemmineR, it supports calculations of atom pair descriptor, similarity comparisons between compounds, and similarity searching against compound databases. Written in C++ for the highest efficiency, this package provides *bindings* to various scripting languages. For example, it includes a companion package for ChemmineR that seamlessly boosts the performance of ChemmineR. In addition, it allows to perform similarity searches directly in the PostgreSQL database engine using standard SQL.

`libmcs` is an MCS solver written in ANSI C. It serves as a reference implementation of our new MCS algorithm introduced in Chapter 3. Bindings are available in several program languages. Besides being widely used in internal projects and building an online MCS solver, it has also been downloaded and used by several third party research organizations.

The geometric embedding toolkit is based on techniques introduced in Chapter 5. It offers the capability to embed a compound set into a high-dimensional Euclidean space. Being a generic method neutral to the choice of similarity measure, it is bundled with support for atom pair descriptor-, PubChem fingerprint- and MCS-based similarity measures, and allows extensions in the form of supporting more similarity metrics to be easily created. The package can also perform automatic evaluation by running random queries and compare the results generated by exact search and embedding-assisted searches.

The EI web application serves as a PubChem gateway with a similarity search functionality with subsecond response time, and up-to-date cluster analysis for each compound in the PubChem Compound collection. This application uses the method introduced in Chapter 6 to build a highly efficient and accurate similarity search engine. The subsecond response time has made it a promising alternative to PubChem's built-in similarity search function, which is more than twenty times slower. Using the new clustering method demonstrated in Chapter 6 of this dissertation, this application is capable of maintaining pre-computed clustering results for the whole PubChem Compound collection.

The ChemMine platform is a web application that intends to build a foundation for sharing and collaboration of cheminformatics tools. By providing a simple publishing and integration platform of reusable functional modules, it aims at reducing the gap between method developers and end users, simplifying method development, publishing and consumption, and facilitating comparative study of cheminformatics methods.

1.8 Publications

This dissertation encompasses five publications with four other authors: Thomas Girke, Tao Jiang, Anna Charisi, Li-Chang Cheng, all from the University of California Riverside. The complete list of publications includes:

- **A maximum common substructure-based algorithm for searching and predicting drug-like compounds**. Yiqun Cao, Tao Jiang, Thomas Girke. ISMB 2008. Also appeared on Bioinformatics 24(13).
- **ChemmineR: a compound mining framework in R.** Yiqun Cao, Anna Charisi, Li-Chang Cheng, Tao Jiang, Thomas Girke. Bioinformatics 24(15).
- **Accelerated Similarity Searching and Clustering of Large Compound Sets by Geometric Embedding and Locality Sensitive Hashing.** Yiqun Cao, Tao Jiang, Thomas Girke. Bioinformatics 26(7).
- **EI: A Gateway for PubChem Compound Database.** Yiqun Cao, Tao Jiang, Thomas Girke. In preparation.
- **ChemMine: A Tool-oriented Approach to Compound Mining Database.** Yiqun Cao, Lei Wang, Tao Jiang, Thomas Girke. In preparation.

Chapter 2

Background

Many similarity measures have been proposed to quantify similarity between chemical structures. After introducing typical ways to represent chemical structure in computer systems, this chapter gives an in-depth overview of a few chemical similarity measures relevant to or used in this study. This includes substructure, superstructure, and atom pair-, and PubChem fingerprint-based methods. Each method has its strengths and weaknesses, which will also be discussed in this chapter. This chapter does not serve as a comprehensive review of chemical similarity methods. For such purpose, one can consult the reviews on this topic by Willett et al. (1998); Nikolova and Jaworska (2004).

2.1 Substructure and Superstructure

Substructure and superstructure relationships are among the most commonly used similarity measures. Given two chemical structures A and B, if structure A is fully contained in structure B, then A is a substructure of B, while B is a superstructure of A. According to the above similar property principle, A and B may share properties that are related to their common substructure. Therefore, a substructure that is putatively associated with certain properties of interest can be used as query to identify in databases all compounds that share this substructure (or superstructure) and possibly its activities.

This type of similarity measures have several drawbacks. First, the utilized matching strategy is very rigid and has a high false negative rate. Second, substructure searching is a knowledge-based approach, in which every utilized query substructure needs to be well defined. If this is not the case or a substructure model is not entirely correct, the search results will be of poor quality and of very limited predictive value. Most importantly, this search type does not generate any quantitative similarity measure, which makes it difficult to rank the search results in a meaningful manner.

2.2 Structure Descriptor-based Chemical Similarity

Structural descriptor-based methods are commonly used approaches for structural similarity searching and bioactivity predictions. These methods represent a chemical structure in its *structure descriptors*, which encode the presence and absence information of structure features in a binary vector or a list. Using such representations, similarity values between a pair of chemical structures can be easily quantified using a *similarity coefficient*.

This type of search represents a much less stringent approach than substructure searches. Typically, the search uses a full structure or substructure as query, and identifies structures in the database that are globally “similar” to the query structure. Eliminating the requirement of an exact match between the query and database structures, similarity searches based on structural descriptor-based methods is far more flexible and allows much higher level of tolerance in the quality of user query structures. Being quantitative methods, they can provide scores as a similarity measure for ranking the results from the most similar to the least.

2.2.1 Structure Descriptors

Based on how the structure features are enumerated, methods to generate structure descriptors can be roughly categorized into two groups. The first group of methods use a *fragment library* for enumeration. For each chemical structure the presence or absence of each fragment is determined. This information is then encoded into a binary string, often referred to as *structural keys* or *fingerprints*, as a compact representation of chemical structures. A representative example of this approach are the MACCS Structural Keys. MACCS contains a total of 166 *structural keys*, corresponding to various groups (e.g., aromatic-C-aromatic) that may appear in chemical structures. Using this fragment library of 166 entries, each chemical structure is represented as a 166-bit binary string. An obvious disadvantage of these methods is the bias introduced by the limited fragment library. Features that are crucial for reliably enriching compounds with a property of interest from a large database may be absent in the fragment library. Varying the fragment library may also result in inconsistent conclusions about whether two structures are similar. A second group of methods avoid the need for a fragment library and instead enumerate *all* substructure features that are present in a given chemical structure. Due to the exponential increase in the number of substructure features with the size of chemical structures, this enumeration is artificially limited to simple substructures. For example, the approach from Carhart et al. (1985) limits the enumeration to all atom pairs, and Young (1999) uses the atom triple features. Each compound is therefore converted into a list that encodes all substructure features identified by the enumeration process. Some methods, such as Daylight fingerprints (James et al., 1992), also compress the representation by folding the list into a fixed-length binary strings. Because of the exhaustive enumeration, this group of methods often offer a less lossy representation of chemical structures at the cost of higher computational complexity and a larger storage footprint.

The descriptors used in this dissertation are based on Carhart et al. (1985)'s original definitions of atom pairs and the fragment library-based PubChem fingerprint (National Center for Biotechnology Information, 2009).

2.2.2 Atom Pair Descriptors

Atom pairs are based on enumerating a simple type of substructures: pairs of atoms. Since introduced by Carhart et al. (1985), their performance has been very well studied (Willett et al., 1998; Sheridan and Kearsley, 2002; Sheridan et al., 1996; Kearsley et al., 1996) and deployed in public compound databases such as ChemMine (Girke et al., 2005b). Because Chen and Reynolds (2002) claims superior performance of atom pair descriptors when used with the Tanimoto coefficient, this method combination is used in this dissertation for various performance comparisons.

Carhart et al. (1985) define atom pairs as a substructure composed of two non-hydrogen atoms and an interatomic separation:

atom 1 description - separation - atom 2 description

The two atoms in an atom pair do not need to be directly connected by a bond, and the separation gives the number of the bonds on the shortest path connecting the two atoms. The description of each atom defines the atom type, the number of non-hydrogen atoms it is directly connected to via a chemical bond, and the number of π electrons that it has.

In our implementation, each descriptor is packed into a 32-bit binary string, which is conveniently handled as an *unsigned integer* internally. The least significant 13 bits stores the description of the heavy atom, among which the first 7 bits are used for atomic number, the next 3 bits are for the number of non-hydrogen neighboring atoms, and the last 3 bits are for the number of π electrons. The most significant 12 bits stores the description of the other atom. Compared to 13 bits used for the heavy atom, this representation uses only 6 bits for the atomic number. This can cause very little information loss because it only happens in rare cases when both atoms have atomic numbers larger than 63.

All unique atom pairs of a structure are then saved in a list together with their number of occurrences in this structure. Compared to Daylight fingerprints that folds fragment information into a fixed-length binary string, our approach keeps more structural information and can potentially increase the quality of similarity measure at the cost of higher memory space requirements and longer computation time. In our implementation, we even make a new copy of the unsigned integers encoding the atom pair for every recurrence. By storing the unsigned integers then in a sorted vector, we can greatly accelerate the calculation time of the Tanimoto similarity coefficient at the cost of a larger memory footprint. We consider this a justifiable trade-off for compound datasets as large as hundreds of thousands of entries, although for larger datasets with millions of entries, online processing based on this type of descriptor may require an optimization scheme as the one demonstrated in Chapter 6.

In general, atom pair descriptors are relatively simple to calculate from chemical structures. Our implementation computes atom pair descriptors directly from the common Structure Data Format (SDF). Unlike Carhart's original implementation, our implementation supports all atom types and has proved its robustness in processing the over 26 million compounds PubChem's Compound collection. It is part of the `libap` package described in detail later in Chapter 7.

2.2.3 PubChem Fingerprints

The PubChem fingerprints represent another type of structure descriptor. Unlike atom pair descriptors, the PubChem fingerprints employ a fragment library to generate an 881-bit binary substructure fingerprint for each chemical structure. Each bit of the fingerprint represents a boolean determination for the presence of an element count, a type of ring system, atom pairing, atom environment (nearest neighbors), etc. in a chemical structure.

The 881 substructure features in the fragment library used by PubChem fingerprint are highly diverse and can be categorized into the following seven categories:

1. **Hierarchic element counts** These features describe the presence or counts of individual chemical atoms for most atom types. For example, the fourth feature is whether there are over 32 hydrogen atoms in the structure.

2. **Ring counts** These features describe the presence or counts of chemical ring systems. The PubChem fingerprint uses Extended Smallest Set of Smallest Ring (ESSSR) to remove ambiguity in the ring counts.

3. **Simple atom pairs** These 64 features are simple pairs of atoms that are directly connected by a bond.

4. **Simple atom nearest neighbors** These features describe the presence of atom nearest neighbor patterns. For example, the 335th feature is whether there exists a carbon atom directly connected to four other carbon atoms.

5. **Detailed atom neighborhoods** These features are identical to the simple atom nearest neighbor features, but specify the bond types that connect the neighboring atoms.

6. **Simple structure patterns** These features describe simple substructure patterns consisting of four to six atoms.

7. **Complex structure patterns** These features describe more complex substructure patterns consisting of seven to eight atoms. All substructure patterns in this group have rings and branches.

Computation of PubChem fingerprint is relatively simple. Because all substructure features are simple patterns, one can create a simple implementation by taking advantage of existing substructure search functionality in software modules such as JChem. An open source implementation is made available by the National Center for Biotechnology Information (2009). This Java-based program can process one compound in a time scale of one-tenth of a second. Though much less efficient than atom pair descriptors, there is a lot of possibilities for optimization. For instance, each feature in the fragment library can be checked independently, and it is trivial to parallelize the fingerprint computation.

Compared to atom pair descriptors, the PubChem fingerprints are more compact and much easier to process and store. With 7 bits at the end to complete the last byte and a four-byte prefix, each fingerprint takes exact 115 bytes. This amounts to less than 3 GB for a library of 26 million compounds and the whole fingerprint database fits into the main memory of a workstation. In our tests, similarity searches using PubChem's fingerprints were seven times faster than the atom pair descriptor method.

2.2.4 Similarity Coefficients

To quantify the similarity value between a pair of chemical structures, structure descriptors are used in combinator with similarity coefficients. A similarity coefficient provides a quantitative measure of the degree of relatedness between a pair of structure descriptors. Some similarity coefficients measure the distance, or dissimilarity instead, and are referred to as *distance coefficients*.

There are many similarity coefficients defined and widely used in many research areas. In this dissertation, however, we will not provide a thorough review of these similarity coefficients. Readers are advised to consult various reviews on this topic Willett et al. (1998); Chen and Reynolds (2002); Haranczyk and Holliday (2008). Instead, we briefly introduce the Tanimoto coefficient, which is the major similarity coefficient used in this dissertation study. For measuring chemical similarity, the Tanimoto coefficient is one of the most commonly used methods and reported to outperform many other common alternatives (Chen and Reynolds, 2002).

Generally, a structure descriptor for a chemical structure, either represented as a fingerprint binary string, a list of structure fragments or a vector of indexed values, can be described as a vector of attributes

$$X = \{x_1, x_2, ...x_n\}, \tag{2.1}$$

where x_i is the value for the ith attribute. Each attribute may corresponds to a bit in the fingerprint binary string, or the absence information of a substructure feature, or a value related to some structure features such as the number of hydrogen bonds. We consider the simple case when the attribute values are integer values. Then the Tanimoto coefficient between two vectors X_A and X_B is defined as

$$S_{A,B} = \frac{\sum_{i=1}^{n} \min(x_{A,i}, x_{B,i})}{\sum_{i=1}^{n} x_{A,i} + \sum_{i=1}^{n} x_{B,i} - \sum_{i=1}^{n} \min(x_{A,i}, x_{B,i})} \tag{2.2}$$

If the attribute values are further limited to binary values, then the above equation can be simplified as the *binary form*:

$$S_{A,B} = \frac{c}{a + b - c} \tag{2.3}$$

where a is the number of unique “on” attributes in X_A, b is the number of unique “on” attributes in X_B, and c is the number of shared “on” attributes in X_A and X_B.

In this dissertation, we use atom pair descriptors with the Tanimoto coefficient as given in Equation 2.2. In this case, each attribute counts the occurrence of one atom pair fragment in a chemical structure. Though there are contradictory claim regarding whether including the occurrence information may improve the effectiveness of similarity measure for similarity searching (Flower, 1998; Brown and Martin, 1996), we use the chosen form of Tanimoto coefficient because Chen and Reynolds (2002) have demonstrated the superiority of counting the occurrences of atom pair descriptors.

2.2.5 Drawbacks of Structure Descriptor-based Similarity

The popularity of structure descriptor-based similarity can be attributed to it being computationally simple and effective in practice. However, generally all of the methods in this category share severe limitations. Most importantly, though sensitive to full-structure similarity, they are unable to identify local similarity between structures or those with large size differences, which is depicted in Figure 2.1. In this example, a researcher would be interested in retrieving the database structure using the query structure because the former contains a substructure highly similar to the query structure. However, a full-structure similarity search using structure descriptor-based similarity would fail to identify the database structure because the overall global similarity between the two structures is not significant. It is further noted that a substructure or superstructure search would also fail.

In addition to being insensitive to local similarity, structure descriptor-based similarities also have vastly increasing false negative rates when it comes to the identification of weaker similarities. This is because compounds of interest with a weaker similarity to the query are often undistinguishable from noise.

Last but not least, from a usability perspective, structure descriptor-based similarity does not provide assistance in illustrating the relatedness between similar structures. This is in contrary to the substructure and superstructure concepts, in which the counterpart of the contained structure can be highlighted in the containing structure to intuitively reveal the similarity. Due to missing visualization assistance, similarity search results based on structure descriptors can also be unintuitive without further investigation.

2.3 Summary

This chapter has discussed the basics of chemical similarity, from representing chemical structures, to structure descriptor types and quantifying similarity between descriptors. For computer-based analysis, chemical structures must be represented in computer-friendly formats. 2D topology is the most commonly used computerized representation of chemical structures because of its simplicity and effectiveness. Using such representations, many metric to measure the similarity between a pair of chemical structures have been proposed. We have reviewed substructure- and superstructure-based methods, which are intuitive and effective, but often too rigid and hard to use than structure descriptor-based methods, which are far more flexible. To illustrate the two types of structure descriptors, atom pair and PubChem fingerprints are introduced in more details as two representative descriptors. To measure chemical similarity, a descriptor must be used with a similarity coefficient. Here we described the Tanimoto coefficient as the most important similarity measure in cheminformatics. Together, the structure descriptors and Tanimoto coefficient are very efficient for measuring similarity, searching chemical compound datasets and performing cluster analysis.

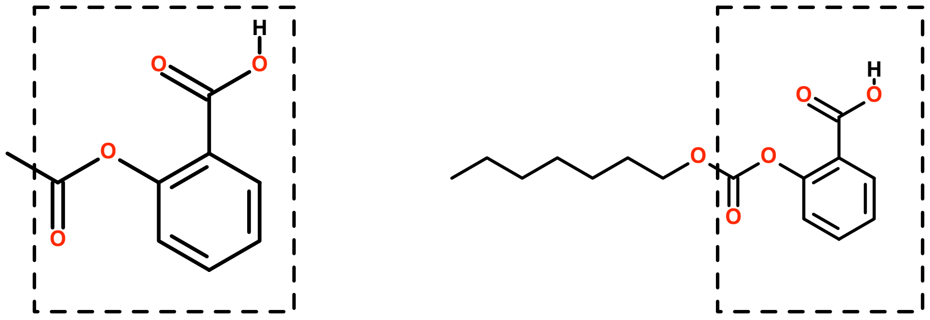


Figure 2.1: Local similarity between compounds. The two structures share a common substructure (dashed boxes). The size difference will result in insignificant scores in 2D fragment-based similarity measures.

Chapter 3

Maximum Common Substructure-based Similarity

This chapter formally introduces the Maximum Common Substructure (MCS)-based chemical similarity. Having been considered as a promising alternative to more commonly used chemical similarity measures for almost two decades, MCS has only been investigated in a few cheminformatic studies. Most of these studies are also based on the general Maximum Common Subgraph problem and do not adapt to the special characteristics of chemical structures and needs of measuring chemical similarity. This chapter describes a new MCS algorithm that is designed for flexible matching of chemical structures. Using this algorithm, we perform a quantitative study of the MCS-based chemical similarity by comparing it against two more commonly used chemical similarity measures.

3.1 Maximum Common Substructure

The Maximum Common Substructure or MCS of two compounds is the largest substructure that appears in both structures. Conceptually, a substructure is not necessarily connected and can consist of several disjoint connected substructures. Practically, we often limit the substructure to be a connected one or to be a set of limited number of disjoint connected ones. In this dissertation, one substructure is said to be *larger* than another if the former one consists of more atoms than the latter. This differs from studies on Maximum Common Edge Substructure (MCES) which uses the number of edges as the size of a substructure.

The compound MCS problem can be modeled using the well-studied problem of Maximum Common Subgraph. Here we include a formal definition of Maximum Common Induced Subgraph, which is used in this study to model compound MCS.

A pair of graphs are said to be *isomorphic* if there is a one-to-one correspondence between their vertices. Moreover, this correspondence has the property that an edge between two vertices exists in one graph if and only if an edge exists between the corresponding two vertices in the other graph. A graph G_s is said to be an *induced subgraph* of a graph G if all vertices of graph G_s belong to the set of vertices of graph G, and there is an edge between two vertices in graph G_s if and only if there is an edge between those two vertices in graph G. For example, in Figure 3.1, graph 3.1c is an induced subgraph of graph 3.1a. A *common induced subgraph* between the graphs G_1 and G_2 are two graphs G_{s1} and G_{s2}, such that G_{s1} is an induced subgraph of G_1, G_{s2} is an induced subgraph of G_2, and G_{s1} and G_{s2} are isomorphic.

In Figure 3.1, graph 3.1c is an induced subgraph of both graphs 3.1a and 3.1b, and is a common induced subgraph of graphs 3.1a and 3.1b. The largest common induced subgraph for a pair of graphs is referred to as the *maximum common induced subgraph* (MCIS) between them. In this dissertation, we model Maximum Common Substructure using MCIS.

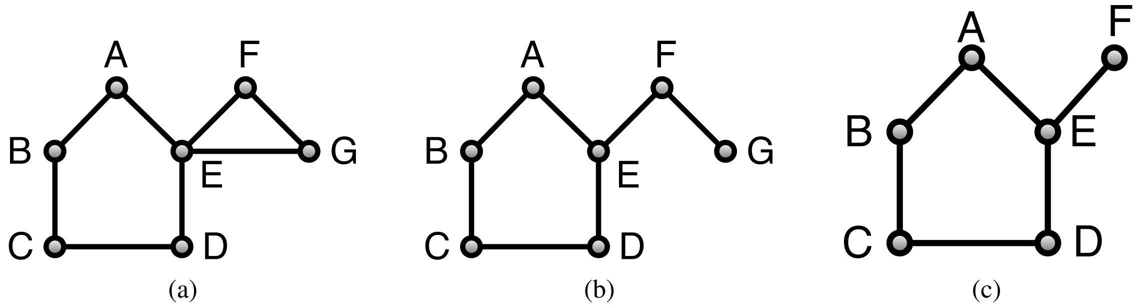


Figure 3.1: Induced subgraph, common induced subgraph and MCIS. Graph 3.1c is an induced subgraph of graphs 3.1a and 3.1b. Therefore, it is the common induced subgraph of the two graphs. It is also the MCIS between them.

3.2 Previous Studies

The MCS problem has been well studied as a general graph matching problem, and has found applications in many areas (Bunke, 2000). Garey and Johnson (1979) showed that the MCS problem is NP-complete. However, many algorithms have been proposed to solve the problem either optimally (McGregor, 1982; Cordella et al., 1998), sub-optimally (Wilson and Hancock, 1997; Wang et al., 1997; Luo and Hancock, 2001) or with error-tolerance (Tsai and Fu, 1979; Dumay et al., 1992; Berretti et al., 2001). A comprehensive review of different MCS algorithms is available in Conte et al. (2004). However, most of these algorithms are focused on general graphs from the pattern recognition area. These methods do not meet the specific needs for efficient graph representations in the chemical compound area, which are sparse, small in size and with bounded degrees. Also, most of the algorithms convert input graphs to association graphs and convert the MCS problem to a clique detection problem. These conversions make it much harder to perform flexible matching effectively. For example, allowing a bromine atom to be matched to a chlorine atom when they are attached to an aromatic ring, but not elsewhere, is much harder to do with association graphs.

Surprisingly, after Cone et al. (1977) proposed to use MCS as a similarity measure, the approach has received much less attention than other similarity measure strategies. This is mainly due to the intractable computational complexity of the MCS problem. Hagadone (1992) has built an MCS-based chemical structure search program for 2D structure drug discovery databases. More recently, Raymond et al. (2002a) applied several heuristic strategies that are based on specific properties of chemical structures to improve the efficiency of the MCS-based similarity search algorithm (Raymond et al., 2002b). The most recent work on the MCS problem is from Yan et al. (2005). The result is restricted to the design of an efficient feature database and does not include a structure comparison step.

3.3 Computing Maximum Common Subgraphs

3.3.1 Compatibility Graph-based Algorithms

Many MCS algorithms convert the MCS problem into the maximum clique problem by introducing *association graphs*, also known as *compatibility graphs* (Levi, 1973; Barrow and

Burstall, 1976; Cone et al., 1977). An association graph between the graphs G_1 and G_2 is a graph, in which the vertex set is the Cartesian product of the vertex set of G_1 and G_2, and the two vertices (u_i, v_i) and (u_j, v_j) are adjacent if, and only if, u_i and u_j are adjacent to G_1, and v_i and v_j are adjacent to G_2, or if u_i and u_j are not adjacent to G_1, and v_i and v_j are not adjacent to G_2. For example, in Figure 3.2 the graph 3.2c is the association graph of the graphs 3.2a and 3.2b. In this association graph, the vertices (A, a) and (B, b) are adjacent, because the vertices A and B are adjacent in graph 3.2a and the vertices a and b are adjacent in graph 3.2b. Similarly, the vertices (A, b) and (B, c) are not adjacent in the association graph, because the vertices A and B are adjacent in graph 3.2a, but the vertices b and c are not adjacent in graph 3.2b.

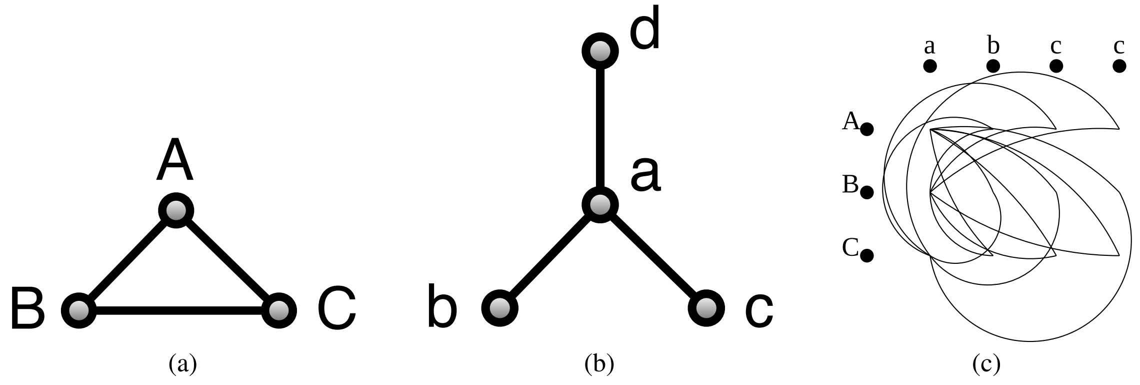


Figure 3.2: Association graph. Graph 3.2c is the association graph of the graphs 3.2a and 3.2b.

An edge between (u_i, v_i) and (u_j, v_j) in the association graph implies that matching u_i to v_i and u_j to v_j is compatible when searching for common induced subgraphs between G_1 and G_2. In other words, if vertices (u_i, v_i) and (u_j, v_j) are adjacent in the association graph, matching u_i to v_i and matching u_j to v_j do not conflict with each other. For example, there is an edge between the vertices (A, a) and (B, b) in the association graph 3.2c. Therefore, matching A in graph 3.2a to a in graph 3.2b, and matching B in graph 3.2a to b in graph 3.2b, does not conflict with the process of finding the common induced subgraph between the graphs 3.2a and 3.2b. However, there is no edge between the vertices (A, b) and (B, c), thus, it is not valid to match vertex A in graph 3.2a to b in graph 3.2b, and to match at the same time vertex B in graph 3.2a to c in graph 3.2b. The reason that these two vertex matchings conflict with each other is that the vertices A and B are adjacent to each other in the graph 3.2a, while the vertices b and c are not adjacent in graph 3.2b. The definition of induced subgraph requires that if the vertices A and B are adjacent in graph 3.2a, then the vertices b and c must also be adjacent in graph 3.2b in order to match A to b and B to c in the process of finding a common induced subgraph between the graphs 3.2a and 3.2b.

Because an edge in the association graph, implies compatibility between two vertex matchings in searching for common induced subgraphs between G_1 and G_2, a clique in the association graph corresponds to a set of compatible vertex matchings, and a maximum clique in the association graph corresponds to a maximum common induced subgraph between G_1 and G_2. The original problem of searching MCIS between two graphs has been converted into the maximum clique problem. For example, one of the maximum cliques in graph 3.2c is $\{(A, a), (B, b)\}$, and it corresponds to the induced subgraph of graph 3.2a containing vertices A and B and an edge between them, which is an MCIS of the graphs 3.2a and 3.2b. These strategies provide a sufficient solution for utilizing the clique detection algorithms for MCS-based approaches. Using this approach, we can take advantage of the availability of different types of clique detection algorithms. However, due to the nature of chemical structures, the conversion will result in an association graph with a large and dense structure (Raymond et al., 2002a). In addition, such a conversion will complicate flexible matching approaches for finding common substructures. For instance, although it is possible to limit the solution to connected subgraphs (Koch, 2001), using association graphs it is not easy to further relax the constraints on their connectedness, such as limiting the solution to contain at most three connected components. Another example is matching different atoms under certain conditions, such as matching bromine and chlorine only when they are both attached to an aromatic ring. This level of flexibility in subgraph matching is also difficult to achieve with association graphs.

3.3.2 Backtracking Algorithm

To avoid the difficulty in the association graph based approaches, we propose a new backtracking algorithm for the MCS problem, which operates directly on the chemical structure graph. This algorithm is based on the VF algorithm, which is designed for the graph and subgraph isomorphism problems (Cordella et al., 2001). The pseudocode of our algorithm is included in the Supplementary Material Section.

To illustrate the basic idea of our backtracking algorithm, one can consider the problem of finding the MCS of the graphs 3.2a and 3.2b in Figure 3.2. A common subgraph of two graphs can be represented by the vertex correspondences between the isomorphic subgraphs. For example, the subgraph of the graph 3.2a consisting of the vertices A and B and the edge

(A, B) and the subgraph of the graph 3.2b consisting of the vertices a and b and the edge (a, b) are isomorphic. This common subgraph can be represented by matching A to a and B to b, and it is denoted by a set of vertex correspondences as $\{A : a, B : b\}$. The backtracking algorithm searches *all* possible combinations of vertex correspondences. It organizes these combinations in a *search tree*. Each node of the search tree is a set of correspondences. By moving down the tree, this set is expanded. Therefore, the set of correspondences at the parent node is always a subset of the set at any child node (see Figure 3.3). For example, the root of the tree is an empty set $\{\}$. The leftmost child of the root, $\{a : A\}$, has one correspondence, which matches vertex a to vertex A. The leaf nodes correspond to *maximal* common subgraphs, which cannot be further expanded. The final solution to the MCS problem corresponds to one of these leaf nodes. The backtracking algorithm searches this tree in a depth-first fashion, and will return the leaf node that contains the largest set of vertex correspondences as the solution to the MCS problem. In this example, the leftmost leaf node ($\{a : A, b : B\}$) may be returned as the MCS of the graphs 3.2a and 3.2b.

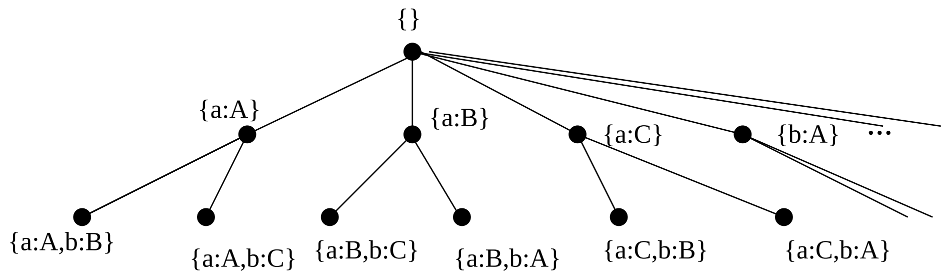


Figure 3.3: Search tree of a backtracking algorithm in search of the MCS between of the graphs 3.2a and 3.2b.

To speed up the computation, we introduce several strategies to reduce the search space. First, we limit the connectedness of the resultant MCS. Second, we use a heuristic based on the induced subgraph constraints to quickly remove branches with infeasible matches. Third, *branch and bound* strategies are employed to discard entire branches in the search tree. Finally, we order the search space such that the branch containing the solution can be searched as early as possible. This can improve the effectiveness of the branch and bound strategy, and it also helps when progressive optimization steps are used.

Connectedness of the Resultant Subgraphs

The MCS of two graphs may contain a number of disconnected structural fragments. This type of MCS is often not desirable when applied to similarity measure of chemical structures. For example, the MCS of two chemical structures in Figure 3.4 consists of five disconnected `C-O` fragments. When using the MCS as a similarity measure between chemical structures, it is often desired to only identify connected MCSs or MCSs consisting of only a limited number of disconnected fragments. Such a connectedness constraint can drastically reduce the search space in finding the MCS. To take advantage of this, our algorithm expands the current common subgraph by growing existing fragments if possible, and it keeps track of the connectedness of the current common subgraph. When it is required to start a new fragment in the common subgraph, the algorithm checks whether the number of disconnected fragments has reached its limit. If the limit has been reached, then the algorithm stops searching along the present branch and considers the common subgraph as the maximal one it can find along this branch of the search tree. In the pseudocode, this constraint is tested in the $order$ subroutine. When the connectedness limit is reached and there is no way to further expand the current common subgraph without introducing a new disconnected fragment, the $order$ subroutine will return a $None$ value to stop searching the current branch. At this point the current common subgraph found will replace the global solution if it provides an improvement to the previous one.

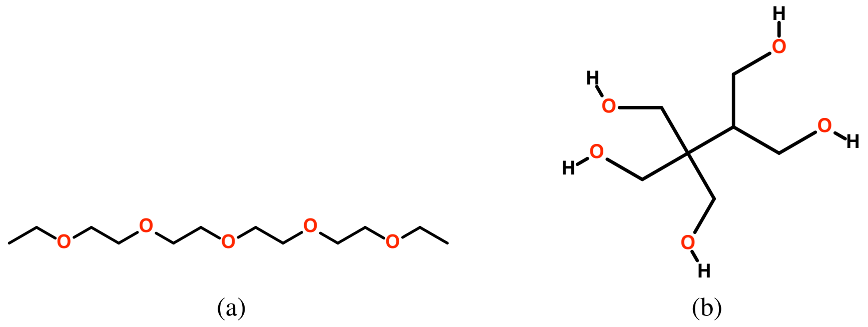


Figure 3.4: Two sample structures. The MCS of the two structures will be five disjoint `C-O` pairs.

Induced Subgraph Heuristic

A heuristic to quickly identify an infeasible set of correspondences can be derived from the definition of the induced subgraph. Consider the problem of finding the MCS between two graphs G_1 and G_2. Let u_1 and v_1 be unmatched in the vertices in G_1 and G_2, respectively. To check whether adding a correspondence, $u_1 : v_1$, to the current set of correspondences will lead to a feasible set of correspondences, the heuristic retrieves a set S_1 of matched neighbors of u_1 in G_1, and a set S_2 of matched neighbors of v_1 in G_2. If the elements in these two sets do not have a one-to-one correspondence in the current set of correspondences, then the correspondence $u_1 : v_1$ cannot be added. This heuristic is implemented in the $compatible$ routine in the pseudocode.

For example, for finding the MCS between the graph in Figure 3.2a and the one in Figure 3.2b, when the current set of correspondences is $\{b : A\}$, it is not feasible to add the correspondence $c : B$, because c is a neighbor of b in graph 3.2b while A and B are not neighbors in graph 3.2a.

Branch and Bound

Our algorithm employs the branch and bound strategy to discard branches of the search tree that cannot lead to an improvement of the candidate solution. An upper bound on the sizes of the common subgraphs at the leaf nodes of the present branch can be estimated. If this upper bound is worse than the candidate solution, then the branch can be discarded immediately and the algorithm backtracks to another branch.

The upper bound is estimated by using the above induced subgraph heuristic. At some point in traversing the search tree, let m be the size of the correspondence set, U be the set of the unmatched vertices in G_1 and V be the set of the unmatched vertices in G_2. For each vertex v in U, if v has not been marked as infeasible by previous search, we apply the above heuristic to test whether it is allowed to match to some vertex in V. If n vertices of U find potential matches in V, then an upper bound on the sizes of the common subgraphs at the leaf nodes of the present branch is $m + n$.

A tighter bound can be achieved by building a bipartite graph from these unmatched vertices. A bipartite $G = (U + V, E)$ is built. Initially E is an empty set. For u_1 from U and v_1 from V, if the correspondence $u_1 : v_1$ passes the induced subgraph heuristic test, then edge

(u_1, v_1) is added to E. To obtain a tighter upper bound, the size of the maximum matching in this bipartite is computed, and then added to m.

Ordering the Search Space

When using branch and bound, a proper order of the search space can help save a lot of computation time. We would like to search the branches that most likely contain the optimal or suboptimal solutions first, to potentially allow the candidate solution to be more quickly improved at an earlier stage. In this way, more branches can be discarded during the branch and bound steps.

A proper order of the search space also benefits *progressive optimization*. Progress optimization is a strategy that returns a suboptimal solution as early as possible to the user, and later progressively improves it over time. For example, progressive optimization can be applied to similarity searches that involve time-consuming computation of the similarity values between the query compound and compounds in the databases. To perform this search, a suboptimal solution is built using the *approximate* similarity values that can be computed faster. If the user is not satisfied with the result, the algorithm can improve it over time by progressively increasing the accuracy of similarity values. Our MCS algorithm can easily be adopted in progressive optimization-based similarity searches. This is achieved by suspending the MCS computation at any time, and using the candidate solution as an approximate result. To further refine the result, the MCS computation can be resumed on demand. Since ordering of search space in our MCS algorithm aims at more quickly improving the candidate solution at an earlier age, the result of progress optimization will be close to the optimal result at an earlier stage and it will stabilize sooner to an acceptable result.

At any point in traversing the search tree for computing the MCS between graphs G_1 and G_2, our algorithm chooses the next node to visit based on the following strategy. It finds the vertex (or vertices) in graph G_1 with the most neighbors in the current common subgraph. Among all the unvisited nodes in the search tree relevant to this vertex (or these vertices), it chooses the one that can improve the aforementioned upper bound the most. Experiments have shown that this ordering strategy can reduce the average computation time.

3.4 Implementation and Availability

The above algorithm has been implemented in C. The corresponding software has been made available as part of `libmcs` package, which is introduced in Chapter 7.

3.5 Evaluation

3.5.1 Datasets

To test the performance of the proposed methods and models, experiments on similarity search and bioactivity prediction have been performed on two public available compound datasets, the NCI AIDS Antiviral Screen dataset and a subset of the NCI Human Tumor Cell Line Screen dataset. Both datasets have been published in the PubChem BioAssay database

(Wheeler et al., 2007), and can be retrieved by BioAssay IDs 179 and 85, respectively. The NCI AIDS Antiviral Assay tested 40,000 compounds for evidence of anti-HIV activity. The NCI Human Tumor Cell Line Screen checked interesting compounds for anti-cancer activity by measuring growth inhibition of the MDA-MB-435 human breast tumor cell line. Both datasets record confirmatory activity information as well as concentration-response relationships, but only the confirmatory activity information was used in our experiments.

The NCI AIDS Antiviral Screen dataset contains structure and activity information for 44,150 compounds, among which 1,812 are classified as having antiviral activity. After removing compounds with missing structure information, the dataset contains 42,689 compounds, 1,504 of which are classified as active. The subset of the NCI Human Tumor Cell Line Screen dataset covers 27,706 compounds, including 1,822 compounds that show anti-cancer activity. After removal of compounds without structure information, 26,366 compounds remain in the dataset, 1,647 of which are active compounds.

3.5.2 Evaluation Methods

We first evaluated the performance of the two similarity measures by applying them in similarity searches. As similarity metrics we used for the MCS-approach the size of the MCS between the query and each compound in the database, and for the AP method the Tanimoto coefficient. The AP method was chosen for benchmarking, because Chen and Reynolds (2002) showed that it outperforms other 2D structural descriptor-based similarity measures, such as the approach based on MACCS keys proposed by Molecular Design Limited (MDL).

The performance of the two similarity search methods were compared by calculating their average *positive predictive values* (PPV) (Altman and Bland, 1994) on the search results concerning the two chosen compound datasets. A series of simulated similarity searches were performed using a random set of active compounds as queries. Each simulated similarity search ranked all compounds from the most similar one to the least similar one relative to the query. Considering the set of the k top-ranked compounds, the ratio of active compounds in this set was defined as the PPV. PPV expresses the capability of a similarity search method to enrich active compounds at the top of the search results. The higher the PPV, with the same k value, the better a similarity search performs.

3.5.3 Experimental Results

MCS Calculation Speed

To evaluate the speed performance of our MCS algorithm, we performed the following test. We calculated the MCS between 1000 randomly selected pairs of compounds from the NCI anti-cancer data which had on average 24 non-hydrogen atoms and 27 bonds. The tests were performed in single thread mode on a workstation with a 2.0GHz Intel Xeon processor and 2GB of RAM. The average time required to finish a MCS calculation was 27.4 milliseconds, with a standard deviation of 35.6 milliseconds. We also tested flexible matching on the same dataset by allowing one atom mismatch and disjoint MCSs. As expected, the flexible matching required more computation time. For example, when allowing one mismatch, the average time for one MCS calculation was about 2.2 seconds with a standard deviation of

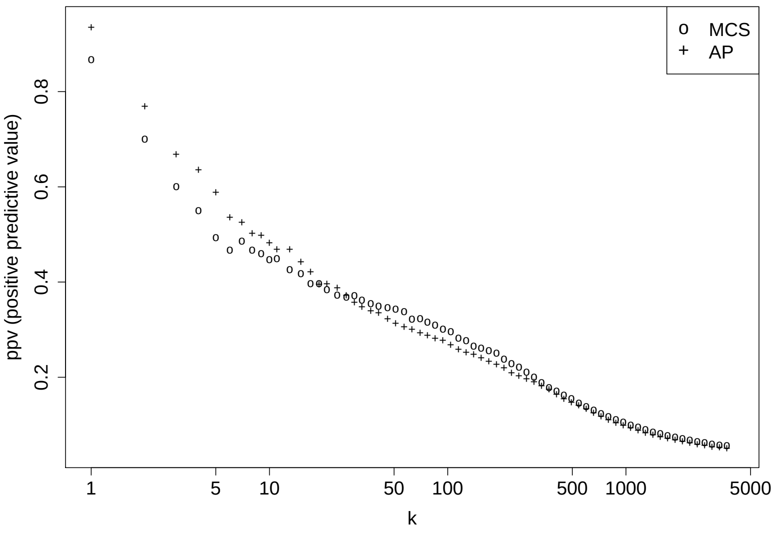
1.6 seconds. Unfortunately, we were not able to perform direct speed and accuracy comparisons with other methods because their implementations are currently not freely available in the public domain. In addition, the provided speed estimates are not comparable to those recorded in previous publications, because of major differences in their implementation, such as heuristic filtering and preprocessing steps before the MCS computation (*e.g.* Raymond et al., 2002a; Yan et al., 2005).

Similarity search

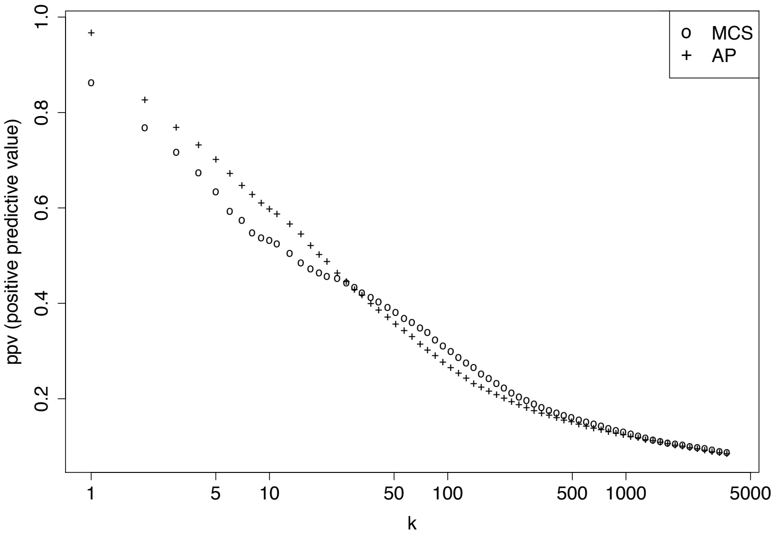
Using the method described in the previous section, we have plotted the average PPV across the simulated similarity searches with various k values representing the number of predicted positives. The plot (see Figure 3.5) shows that PPVs of both search methods decrease when k increases. This means that if one moves down the ranked search result list, the cumulative ratio of active compounds decreases. Interestingly, the MCS method performs better for larger k values (>29) and the AP method for smaller k values. This result is expected, because the AP method performs well in detecting compounds with high global similarities which are more likely to share the same bioactivity than compounds with local similarities. On the other hand, the MCS method is able to rank global and more sophisticated local similarities equally well. Therefore, the MCS method identifies a larger set of bioactive candidates, but it does not necessarily rank compounds with global similarities at the top of the list. The following example illustrates this performance benefit of the MCS approach. For instance, if the user is interested in more than 29 of the top compounds, the MCS searches exhibit a better performance than the AP searches. If the top 100 compounds are of interest, which is a reasonable choice for similarity searches, the MCS method has a PPV of 0.30, and the AP method has a PPV of 0.27.

3.6 Summary

This chapter describes a new backtracking algorithm for finding the MCS between a pair of graphs and evaluates its performance in similarity search applications. This algorithm can be applied to chemical structure graphs directly (instead of association graphs), and hence supports several matching constraints as well as relaxations that can be useful when applied to comparison of chemical compounds. Several strategies are introduced to increase the efficiency and flexibility of the algorithm. This algorithm can be effectively used with a progressive optimization strategy, which is very beneficial for ranking a large number of chemical structures by their similarities to query structures. Our experimental results show that despite less efficiency, the MCS-based similarity measure is more effective in searching chemical databases than the well-known atom pair- and PubChem fingerprint-based methods. It is also capable to detect novel compound structures that are missed by other methods, and is valuable to be used in combination of other methods.



(a)



(b)

Figure 3.5: Performance Comparisons of AP- and MCS-based search methods. The average PPVs from all simulated similarity searches are plotted against the k values. Part (a) provides the results for the NCI antiviral dataset and part (b) for the NCI anti-cancer dataset.

Chapter 4

Similarity Measures and Machine Learning Models

This chapter continues the study of MCS as an alternative chemical similarity measure by applying it in building statistical prediction models for biological activities. Similarity search is a crude way to exploit chemical similarity because it uses knowledge about active compounds and only one active compound at a time. When information about multiple active compounds and inactive ones are available, building a statistical model for predicting biological activity offers more value. This chapter demonstrates a generic scheme to easily apply virtually any chemical similarity in machine learning models. This scheme enables new applications of MCS-based and many other similarity measures that are otherwise difficult to be employed in these models. It also allows easy integration of multiple similarity measures in a single model to potentially increase the predictive power.

4.1 Machine Learning in Cheminformatics

Recent advancements in combinatorial chemistry and compound synthesis have greatly expanded our access to chemical space. However, our capability to screen an ever-increasing set of compound libraries has only been slightly increased in the past years. Moreover, even by screening the set of all compound libraries currently available to human beings, we are still far short of the over 10^{60} potential compounds in the chemical space (Dobson, 2004a). The need for an alternative to the traditional method of producing and testing every single molecular combination possible has sparked a lot of interest in using machine learning to assist refined search as an integral part of drug screening process.

Many model building methods have been applied for predicting biological activities based on structural similarities. Schneider (2000) reveals that artificial neural networks (ANN) have been used to "turn a blind search for novel drug-like molecules into an informed search". Burbidge et al. (2001) demonstrate the potential of support vector machines (SVM) for structure-activity relationship analysis and compares it to several other machine learning techniques used in cheminformatics, including several ANN methods and decision trees. Schwaighofer et al. (2007) present an accurate statistical model of aqueous solubility based on Gaussian Process nonlinear regression model. Cannon et al. (2007) investigate the classification performance of circular fingerprints in combination with the Naive Bayes Classifier (MP2D), Inductive Logic Programming (ILP) and Support Vector Inductive Logic Programming (SVILP).

All these methods require the compound representation to be in a feature space of limited dimensions, or in other words, the input to be vector data. They also work on structural descriptors directly and often employ their own similarity coefficient explicitly or implicitly. For instance, in Cannon et al's investigation of Naive Bayes Classifier method for classification of bioactive and inactive compounds, each compound is represented by a feature vector. Each element of this feature vector is used in calculating a conditional probability of the compound being active. This does not utilize any similarity coefficient. Similarly, in Burbidge's study, Gaussian RBF is used as the kernel function for the SVM:

$$\mathbf{K}(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{||\mathbf{x} - \mathbf{z}|\,|^2}{2\delta^2}\right).$$

Here the use of Euclidean distance assumes both $\mathbf{x}$ and $\mathbf{z}$ representing chemical compounds to be vectors. Euclidean distance is used in place of any similarity coefficient that may be more appropriate to the used with the structural representation.

Because MCS-based similarity *is* a similarity coefficient, it is difficult to enforce its use in the machine learning models more sophisticated than nearest neighbor classifiers. As a similarity coefficient, MCS-based similarity quantifies relatedness between compound structures by working directly on the 2D graph representation of structures without any intermediate structural descriptors.

The family of kernel-based methods is one exceptional case in which MCS-based similarity may be applicable. This is because kernel-based methods such as SVM allow custom kernel functions, and one may be able to create a kernel function based on MCS-based similarity. Neuhaus and Bunke (2006) introduced a kernel function based on graph edit distance. The authors pointed out that the validity of the kernel method cannot be established for edit distance, but the proposed kernel functions are nevertheless suitable for classification purpose and perform well. Since MCS and graph edit distance is related (Bunke, 1997), one can construct a similar SVM classifier using MCS-based similarity. However, in addition to the lack of theoretic validity of the kernel functions, computational complexity of MCS and therefore the derived kernel function is a great concern in such a classifier.

Another notable exception is nearest neighbor classifier. Nearest neighbor classifier works by classifying an object based on the most common labels of its labeled neighbors in the training dataset. Such approach requires only a suitable distance measure. However, employing MCS-based similarity in nearest neighbor classifier is computationally very expensive, because it would require an against-all comparison for each object to be classified.

4.2 Vector Representation based on MCS Similarity

To avoid performing a large amount of expensive MCS calculations, and to achieve the maximum level of flexibility in harnessing the power of modern machine learning techniques, we use the concept of *basis compounds* to generate vector representations of chemical structures using the MCS approach. Basis compounds are a designed set of n diverse compounds, $C_1, C_2, \ldots, C_n$, such that each compound D_i from the compound database can be represented by an n-dimensional vector

$$(|MCS(D_i, C_1)|, |MCS(D_i, C_2)|, \ldots, |MCS(D_i, C_n)|).$$

In other words, each compound is represented by a vector, where each element is the size of an MCS between the compound and a basis compound. The set of basis compounds can be carefully designed or randomly chosen from a large set of diverse compounds. It is important to note that vectorization of a chemical structure using basis compounds requires only a numerical similarity measure and is not specific to MCS. Therefore, the MCS-based similarity measure can be replaced by any other similarity measure scheme, such as the AP-based similarity measure. When using basis compounds, each chemical structure can be represented as a multi-dimensional vector. Since the dimensionality is controllable and each vector contains real numerical values, these vectors can be used in statistical modeling and machine learning techniques. In this paper, we use SVMs as an example modeling technique to demonstrate the utility of the above vectorization method in predicting bioactive compounds.

4.3 Support Vector Machines

Support vector machines (SVM) was chosen as the machine learning technique to build models for predicting biological active compounds. SVM is a linear classifier that simultaneously minimizes the empirical classification error and maximizes the geometric margin. When used with kernel methods, SVM can yield powerful non-linear classifiers. Formally, given a set of input and response

$$(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)$$

the SVM method will search for a vector $\mathbf{w}$ and a margin b, such that for every pair of input and response $(\mathbf{x_i}, y_i)$,

$$y_i(\mathbf{w}\mathbf{x_i} - b) \geq 1, \tag{4.1}$$

and $|\mathbf{w}|$ is minimized. If 4.1 does not have a solution, which means there is no hyperplan that can separate the positive points from the negative points, a soft margin ξ is introduced, and the problem is then

$$min(||\mathbf{w}||^2 + C\sum_i \xi_i) \;\; s.t. \;\; y_i(\mathbf{w}\mathbf{x_i} - b) \geq 1 - \xi_i, 1 \leq i \leq n \tag{4.2}$$

SVM is a linear classifier. However, it can be converted into a non-linear classifier by using a kernel method. For this, the kernel function can transform a vector in feature space into a higher dimensional space, where SVM is applied. Although the classifier in the higher dimensional space is linear, it is non-linear in the original feature space. The approach has been described in all details by Boser et al. (1992).

For our application of predicting biologically active compounds, the vector representation of a chemical structure is the input, and the biological activity of the structure is the response. A training testset with known structures and known activities is used to train the SVM, and then the SVM can predict the biologically active candidate structures in novel compound libraries given their structure information.

4.4 Sequential Screening

Sequential screening, also known as hybrid screening, is a chemical screening strategy that combines *in vitro* screening and *in silico* modeling and prediction. In sequential screening, a relatively small set of compounds is assayed and the results are statistically analyzed to produce a mathematical model. The model is used to predict activity and select additional compounds for screening. The new screening results are added to the results for the initial set and a new model is determined. This process is iterated until a sufficient amount of candidate compounds has been identified.

The above vectorization and SVM method can be easily incorporated into a sequential screening strategy. For this, the initial screening can provide both the basis compounds and the training data, while the prediction is used to guide the second screening. With the results of the second screening, the basis compounds can be redesigned, and the model can be retrained.

4.5 Implementation

We use the MCS implementation described in Chapter 3. All statistical analyses, modeling and prediction steps were performed in R 2.5. The SVM implementation used by this project is from the `e1071` package (Dimitriadou et al., 2005), which is based on the C++-implementation by Chang and Lin (2001). The function `svm` is used for building the models with the kernel function set to `radial`, and the `predict` function is used for predictions.

The aforementioned `svm` function is used as a regression machine, in which the response values are numeric values representing the confidence of bioactivity of compounds. The response values are determined in the following way. If a training compound is active, then its corresponding response y is set to 1, and if it is inactive, its corresponding y is set to 0. The predicted responses for the testing data points will be numeric values, each of which is a confidence scores describing how likely the corresponding compound is active.

After the prediction step, each compound will have its confidence score. All the compounds can then be sorted by decreasing order of their scores. The top results can then be used for the following screening iterations. To illustrate this in the context of the sequential screening process: a screening facility can use the predicted bioactive candidates in experimental screens and the verified hits can be used to improve the subsequent predictions.

4.6 Evaluation

4.6.1 Datasets

The same dataset used in Chapter 3 is used in this evaluation.

4.6.2 Evaluation Methods

The MCS- and AP-based similarity measures were also compared with regard to their performance in building SVM models for predicting bioactive candidates in compound databases. The proposed basis compound concept was used in converting chemical structures into vectors, which were then used in the training and testing of SVM-based prediction models. Between 20 to 140 randomly selected compounds from the whole dataset were used as basis compounds in the tests.

Three models were built and tested independently. The three models varied in their representations of chemical compounds. The first model uses the MCS similarity measure to build a vector for each compound (see section 4.2). The second one uses the AP similarity measure for vector building. The third one - the *hybrid* model - concatenates the vectors from both previous models. The three models were tested in a series of experimental settings which varied in the size of the training set, the number of basis compounds, and the coefficients used in defining similarity between compounds. The output of each model was a ranking of the compounds, in which the compounds were sorted by how likely they were bioactive based on the prediction. From these ranked results, an ROC curve was generated by plotting the true positive rates against the false positive rates (Provost and Fawcett, 2001). The performance of the prediction models for different experimental settings were then evaluated by measuring the corresponding areas under the ROC curves (AUC). Each experimental setting was tested in multiple runs, and the average AUC was used for comparisons, in which a larger AUC value indicates a better performance.

4.6.3 Experimental Results

To compare the predictive power of the MCS method against the AP method and the hybrid approach, each method was coupled with SVMs to build a model for predicting bioactive compounds. The input of the SVMs was the vector representation of compounds generated by each method. The output was either 0 for inactive compounds, or 1 for active compounds. SVM-based models were used to predict the bioactive compounds. The AUC values were recorded and used for evaluating the qualities of the predictions.

The SVM implementation provided in the R package `e1071` (Dimitriadou et al., 2005) was used for the tests. This package is based on the widely-used C++-based LibSVM from Chang and Lin (2001). The default regression parameters defined in the package were used in the tests. In one of our tests we also used the *ROC Curve for Binary SVM* tool distributed with LibSVM, to achieve a fair comparison against data reported elsewhere. In this test, cross validation was used to select the parameters (C and γ) for the radial kernel function-based classifiers.

Varying the Size of the Training Set

One of the key factors impacting the prediction performance is the size of the training set. For real applications, it is desirable to achieve an acceptable performances with small training set sizes, because the prediction model is often used to select compound subsets from a large compound library to be used in the experimental screening process. The prediction models are only practically useful if the training set, obtained from an experimental screening process, is much smaller than the whole screening library. This is especially important for sequential screening, in which only a small initial set of compounds are experimentally tested and used to train the initial models. For both datasets, we performed experiments with training sets consisting of 1%, 5%, 10%, and 25% of the whole dataset. 20 basis compounds were selected randomly from the whole dataset for each test. The average AUC values and standard deviations are listed in Table 4.1.

Table 4.1 shows that the performance of all three models improves with the size of the training dataset. The hybrid model achieves in all cases the best performance, while the MCS model performs almost consistently better than AP model. Moreover, the hybrid model outperforms the AP model with much smaller training sets. For instance, in both datasets, the hybrid model performs with a training set of 10% of the dataset better than the AP model with a training set that is 2.5 times as large.

Models	*Training set size*			
	1%	*5%*	*10%*	*25%*
MCS-based	57.9(3.0)	64.0(2.4)	67.0(1.3)	70.0(0.9)
AP-based	58.2(3.1)	63.7(1.8)	65.8(1.8)	68.9(1.5)
Hybrid	61.3(3.4)	66.7(1.9)	69.2(1.3)	71.6(1.2)

(a)

Models	*Training set size*			
	1%	*5%*	*10%*	*25%*
MCS-based	60.3(2.8)	65.4(1.8)	68.0(1.7)	70.9(1.3)
AP-based	59.3(3.3)	65.2(1.8)	67.8(1.7)	70.9 (1.8)
Hybrid	62.7(3.2)	69.2(1.8)	71.8(1.4)	74.8(1.2)

(b)

Table 4.1: Average AUC values using different prediction models and different training set sizes. Table (a) lists the result for the NCI antiviral dataset, and Table (b) lists the result for the NCI anti-cancer dataset. The *MCS-based* model uses the absolute MCS sizes to represent a chemical structure as a vector. The *AP-based* model uses the AP-based similarity, and the *hybrid* model concatenates the vectors from both previous models. Standard deviations are given in parentheses.

Overlap of Active Compounds Identified by the Models

The superior performance of the hybrid approach can be explained by the performance differences of the individual models in identifying similarities with different structural features (*e.g.* global or local similarities). For instance, in one of the tests with the NCI antiviral dataset, the active compound NSC 79521 was ranked by the AP model at position 1, while the MCS model ranked it at position 10,568. On the other hand, the active compound NSC 683278 was ranked at position 23 by the MCS model, and at position 2,673 by the AP-based model. The hybrid model ranked both of these actives at the very top of the list, at positions 33 and 4, respectively.

To evaluate these performance differences more carefully, we analyzed the overlaps between the active compounds identified by each pair of the three models. For a pair of models

A and B, if the set of active compounds identified by A is $S(A)$, and the set of active compounds identified by B is $S(B)$, we define ***A's coverage of B***, or $C_A(B)$ as

$$\frac{|S(A) \cap S(B)|}{|S(B)|} \tag{4.3}$$

The larger the coverage is, the better model A can cover the chemical space covered by B. If both $C_A(B)$ and $C_B(A)$ are small, the two models cover different subspaces of the chemical space and they complement each other. We studied the cross coverage between each pair of models by plotting the corresponding coverage values with different number of predicted active compounds, n. The plot shown in Figure 4.1 shows that with a typical choice of n, one's coverage of the other is relatively small between the AP model and the MCS model. On the other hand, the hybrid model's coverages of the other two models are relatively high. For example, if 5% of the compounds are predicted to be active, 62% of the active compounds identified by the AP model are also identified by the MCS model, and 59% vice versa. On the other hand, the hybrid model identifies 85% of the active compounds of the AP model and 73% of the MCS model. These data show that the AP model and the MCS model cover different subspaces of the chemical space and they complement each other. The hybrid model effectively utilizes both the MCS and the AP similarity information, and covers a wider chemical space than any one of the other two models alone.

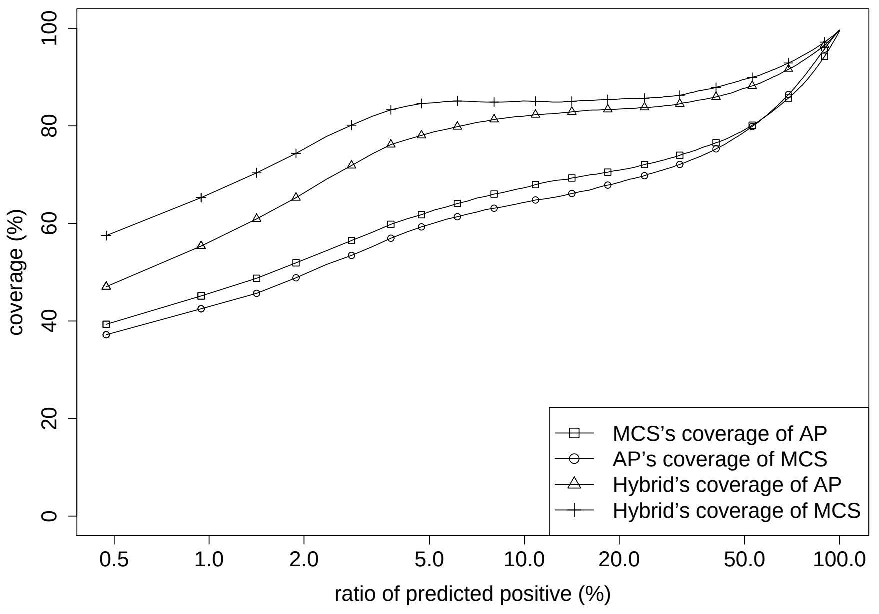


Figure 4.1: Cross coverages between pairs of prediction models. The x axis is the number of predicted positives over the total number of compounds in the dataset. The y axis is one model's coverage of another model (see Equation 4.3). For example, when 10% of the compounds are predicted to be active, 82% of the compounds identified by the AP model are also identified by the hybrid model. This means that under this condition the hybrid model's coverage of the AP model is 82%. This corresponds to point $(10, 82)$ in the curve.

Varying the MCS Coefficients

In addition to the absolute size of the MCS between structures, a normalized MCS size has been considered in previous studies to measure similarity between a pair of structures (Bunke and Shearer, 1998). We performed tests using different MCS-based similarity measures, or *MCS coefficients*. The following coefficients are included in these comparisons:

$$\frac{MCS(G_1, G_2)}{\max(|G_1|, |G_2|)}, \tag{4.4}$$

$$\frac{MCS(G_1, G_2)}{\min(|G_1|, |G_2|)}, \tag{4.5}$$

$$\frac{MCS(G_1, G_2)}{|G_1|} \tag{4.6}$$

Here, G_1 is the chemical structure to be vectorized, and G_2 is one of the basis compounds. Each coefficient is also used to derive a hybrid model, which is included in the comparisons. All the models have been tested using training sets of different sizes and multiple sets of 20 randomly selected basis compounds on both compound sets. Results are listed in Table 4.2.

Data in Table 4.2 suggests that the coefficient from Equation 4.5 exhibits in general the best performance, although the advantage is marginal. When using only MCS information, *MCS c2* generates the best results in the NCI antiviral dataset, and close to the best results in the NCI anti-cancer dataset. When both the MCS and the AP information are used, the

Models	*Training set size*			
	400	*2000*	*5000*	*10000*
MCS	58.5(3.0)	64.3(2.4)	67.2(1.3)	69.8(0.9)
MCS c1	58.8(3.1)	65.2(1.7)	68.2(1.4)	70.0(1.9)
MCS c2	59.7(3.2)	67.0(1.5)	69.2(1.0)	71.0(0.9)
MCS c3	59.2(2.7)	65.8(1.7)	68.5(1.1)	70.5(1.9)
hybrid	61.3(3.4)	67.0(1.9)	69.7(1.3)	71.5(1.2)
hybrid c1	60.1(3.3)	66.6(1.6)	69.4(1.3)	71.8(1.7)
hybrid c2	60.8(3.4)	67.6(1.7)	70.4(1.2)	72.3(0.9)
hybrid c3	60.2(3.2)	67.0(1.7)	69.9(1.2)	72.3(1.2)

(a)

Models	*Training set size*			
	300	*1000*	*2000*	*5000*
MCS	60.0(2.8)	64.5(1.8)	66.8(1.7)	69.9(1.3)
MCS c1	59.5(3.2)	64.7(1.8)	67.7(1.5)	71.1(1.3)
MCS c2	59.4(3.1)	64.6(1.8)	67.5(1.5)	71.0(1.2)
MCS c3	58.2(3.0)	64.2(1.7)	67.4(1.4)	71.4(0.9)
hybrid	62.7(3.2)	67.6(1.8)	70.4(1.4)	73.8(1.2)
hybrid c1	61.5(3.2)	67.2(1.6)	70.2(1.5)	73.8(1.2)
hybrid c2	61.7(3.5)	67.4(1.7)	70.7(1.3)	74.4(1.1)
hybrid c3	60.4(3.4)	66.8(1.7)	70.2(1.5)	74.2(1.3)

(b)

Table 4.2: Average AUC values using prediction models based on different MCS coefficients and different training set sizes. Standard deviations are given in parentheses. Table (a) lists the result for the NCI antiviral dataset, and Table (b) lists the result for the NCI anti-cancer dataset. The *MCS* model uses the absolute MCS sizes. The models *MCS c1*, *MCS c2* and *MCS c3* use the MCS coefficients listed in Equations 4.4, 4.5 and 4.6, respectively. The *hybrid* model uses the absolute MCS sizes and the AP information. The models *hybrid c1*, *hybrid c2* and *hybrid c3* use the MCS coefficients listed in Equations 4.4, 4.5 and 4.6, respectively, and the AP information.

Models	*Number of basis compounds*						
	20	*40*	*60*	*80*	*100*	*120*	*140*
AP	70.7	72.4	73.3	73.9	74.0	72.9	72.9
MCS c2	73.0	74.6	74.6	75.8	75.5	75.4	75.2
hybrid c2	74.4	75.2	75.4	76.2	76.1	75.4	75.6

(a)

Models	*Number of basis compounds*						
	20	*40*	*60*	*80*	*100*	*120*	*140*
AP	69.5	72.4	72.6	73.2	73.9	74.7	73.7
MCS c2	71.0	74.2	75.2	75.5	75.9	76.6	76.4
hybrid c2	74.4	75.9	75.9	76.1	76.5	77.2	76.9

(b)

Table 4.3: Average AUC values using the prediction models with different numbers of basis compounds. Table (a) lists the result for the NCI antiviral dataset, and Table (b) lists the result for the NCI anti-cancer dataset. For the NCI antiviral dataset, 25000 randomly selected compounds were used as the training set. For the NCI anti-cancer dataset, 5000 randomly selected compounds were used as the training set.

hybrid c2 model (Equation 4.5) shows the best performance in both datasets.

Varying the Number of Basis Compounds

Another factor that might affect the performance of prediction models is the choice of the basis compounds. In principle, the set of basis compounds should be as diverse as possible and should cover the chemical space as thoroughly as possible. To evaluate the sensitivity of our models regarding the choice of basis compounds, we performed a series of tests on both compound datasets. We varied the number of basis compounds from 20 to 140 in increments of 20 compounds. The obtained AUC values are listed in Table 4.3.

The results in Table 4.3 indicate that for all the listed models, the AUC values increase with the number of basis compounds. This trend peaks at around 80 to 120 basis compounds. After this, no significant improvement can be achieved. It is important to point out, that the

Models	*hybrid c2*	*physicochemical-based*	*descriptor-based*	*SUBDUE*	*SubdueCL*	*FSG*
AUC	82.3	47.3	72.1	58.5	65.2	79.4

Table 4.4: AUC values for different prediction models applied to the NCI antiviral dataset. *Hybrid c2* is our proposed hybrid model using the coefficient from Equation 4.5 and 80 randomly selected compounds as basis compounds. Radial kernel function-based classifier were used, with C set to 64 and γ set to 0.0625. The *physicochemical-based* method is described in Deshpande et al. (2005). The *descriptor-based* method combines 166 MACCS keys from the MDL and Daylight fingerprints. *SUBDUE* (Holder et al., 1994) and *SubdueCL* (Gonzalez et al., 2001) are methods based on heuristic substructure discovery. *FSG* is the method proposed by Deshpande et al. (2005) using topological subgraphs but not geometrical subgraphs.

peak performance of each method in Table 4.3 should be used for comparisons between the three methods, and not their performance within the same number of basis compounds.

Comparisons with Other Prediction Approaches

We compared the performance of our prediction models to results reported by Deshpande et al. (2005). In this study, the authors listed the prediction performance measured by the AUC values of several representative approaches. The authors applied 5-way cross validation to all models using several datasets, among which was the NCI antiviral dataset. To compare our proposed approach with these models, we used the *ROC Curve for Binary SVM* tool to calculate the AUC value of our proposed hybrid model by applying 5-way cross validation on the NCI antiviral dataset. The result for our hybrid model is listed in Table 4.4 along with the data reported by Deshpande et al. (2005).

According to these comparisons, our method outperforms all the methods listed in Table 4.4. In addition to its improved performance, our method uses a feature space with a much lower dimensionality. Therefore, it is computationally less complex to train the SVM model and make predictions. For example, to achieve the performance exhibited in the table, the FSG model used 18,542 features to represent a compound. The authors had to apply feature selection to reduce this number to 2460. However, this feature selection step also reduced the prediction quality and cut the AUC value to 78.5. In comparison, our method used only 160 features to achieve the better performance without requiring any feature selection step. Furthermore, it is easy to incorporate other similarity measures into our hybrid model to further advance its performance. For example, one can easily incorporate the similarity measure used in FSG into our hybrid model.

4.7 Summary

This chapter describes the development of a simple generic scheme to apply virtually any chemical similarity measure to machine learning models and applies it in building SVM models based on MCS similarities to efficiently predict biologically active compounds. Using this scheme, it also established the increased predictive of power of combining multiple similarity measures in a single hybrid model. Quantitative evaluations have shown the effectiveness of this model building technique and also the superior quality of MCS similarity when used in predictive models alone and in combination of other similarity measures.

More specifically, the concept of basis compounds is proposed to easily integrate the MCS-based similarity measure and other similarity measures in modern machine learning techniques to form effective bioactivity prediction models. This concept allows for a vector representation of chemical structures, similar to traditional fingerprint approaches, while avoiding many drawbacks of the traditional fingerprint approaches. Finally, the derived vectors are incorporated into SVMs to build prediction models of biological activities of compounds.

Our experimental results show that the MCS-based similarity measure is more effective in searching chemical databases than the well-known AP-based method. They also show that the SVM models based on vectors derived from basis compounds are effective for identifying bioactive compounds. Moreover, the MCS-based similarity measure complements the AP-based measure effectively. Our proposed hybrid model, which combines the MCS and the AP similarity measures, provides the most effective predictions in comparison with the existing approaches.

Chapter 5

High-Fidelity Embedding of Chemical Structures

The basis compound concept introduced in the previous chapter is a simple yet powerful technique to generate vector representations of chemical compounds based on arbitrary similarity measures. Despite the lossy nature of the conversion, this numeric representation captures important similarity information and at the same time brings extraordinary flexibility and possibilities. To enable and explore more interesting applications, this chapter aims at demonstrating another technique to produce vector representation in Euclidean space. This technique is less lossy, hence the name of *high-fidelity embedding*, and allows high-quality reproduction of a chemical similarity value using the Euclidean distance between the corresponding vector pair. In additional to many applications that directly benefit from the accelerated similarity calculation, adaptation in many more areas can be foreseen.

5.1 Previous Studies on Embedding

Our goal of generating vector representation for chemical structure in Euclidean space which can be used to approximate compound dissimilarities by the inter-vector distances is in line with studies on *embedding*. Embedding objects in Euclidean space offers many benefits, such as the possibility of accelerating nearest neighbor search. In addition, they are useful for *all-pair* query approaches used in data visualization, clustering and data mining (Faloutsos and Lin, 1995).

The problem of geometric embedding has previously been studied and applied to nearest neighbor search in metric space. Three of the most widely used methods in this area are multi-dimensional scaling (MDS; Kruskal and Wish, 1978), Stochastic Proximity Embedding (SPE; Agrafiotis and Xu, 2002; Agrafiotis, 2003; Smellie et al., 2006) and FastMap (Faloutsos and Lin, 1995). MDS is used to discover structures in data sets by representing the relationships among its objects as spacial distances in a low-dimensional display plane.

Given a set of N objects and their pairwise dissimilarities, MDS induces a D-dimensional vector for each object such that dissimilarities among objects are preserved by the Euclidean distances among the induced vectors. More specifically, the following stress function is minimized:

$$stress = \sqrt{\frac{\sum_{i,j}(\hat{d_{i,j}} - d_{i,j})^2}{\sum_{i,j} d_{i,j}^2}}, \tag{5.1}$$

where $d_{i,j}$ is the dissimilarity measure between objects i and j, and $\hat{d_{i,j}}$ is the Euclidean distance between the two induced vectors for objects i and j.

It is computationally expensive because it depends on the availability of all pairwise dissimilarities among the objects in a data set. As a result, it becomes quickly infeasible for compound databases with more than a few thousand entries.

Many variances of MDS have been proposed to solve the problem for large data sets. Chang and Lee (1973) selected from the whole data set a smaller number of representative objects, called *pivots*, and applied classic MDS to this subset. The remaining objects were then embedded to the same Euclidean space based on their distances to the pivots.

SPE is an alternative to MDS. As a self-organizing algorithm, SPE starts with an initial assignment of data points to coordinates and carries out iterative pairwise refinement steps by adjusting randomly selected pairs of coordinates to better approximate the corresponding dissimilarity values. SPE is very efficient and is reported to scale linearly with the size of the data set.

FastMap is another fast alternative of MDS with linear time complexity. The key idea of the method is to consider the objects as unknown vectors in D-dimensional space. The algorithm recursively projects these unknown vectors onto perpendicular lines, and uses known dissimilarity values among the objects to calculate the required projections and vector coordinates. It gains its computational efficiency by directly calculating the induced vectors rather than iterative improvement steps used by most MDS implementations. In addition, it requires only $2n + 1$ dissimilarity computations for n objects, while MDS requires $\frac{n^2}{2}$. However, the proper choice of the set of *pivot objects* used to compute perpendicular lines can be complicated by the presence of large numbers of *outlier compounds* that are not similar to any other compounds in a compound library and will be selected as pivot objects by definition to result in explosive number of dimensions.

Several studies have used these embedding methods in cheminformatics, but all of these studies target a low-dimensional embedding space in small to medium size compound datasets. For instance, Agrafiotis and Lobanov (1999) reported an algorithm based on k-dimensional (or k-d) trees for diversity analyses, and mentioned the possibility of using MDS to generate low-dimensional vector representations to feed into the k-d trees. The method was tested using 10,000 artificial points of at most eight dimensions. Shi et al. (2000) investigated the use of several embedding and dimensionality reduction techniques to visualize 70,000 anti-cancer drugs in 2D space. Many methods used in these studies, such as principle component analysis (PCA) (Meglen, 1992) even serves the specific purpose of reducing the dimensionality of vector data. By using a low-dimensional representation, these studies traded precision in data representation for manageability and ease of processing in applications.

5.2 Modified MDS for Embedding

5.2.1 Overview

The goal of this study is to design an embedding method to produce high-dimensional vector representation of chemical structures that is accurate enough to preserve inter-molecule similarity and scalable enough to comfortably process large compound dataset with millions of entries. The initial steps of the embedding procedure are similar to the method from Chang and Lee (1973), but it differs significantly in its final optimization steps. The method starts by dividing all compounds into two sets: a small *reference compound set* and a much larger *target compound set*. The reference compound set is a user-definable parameter that can be generated by maximum diversity or random selection methods of compounds in a given library. Traditional MDS is applied to obtain the coordinates of the induced *reference vectors* for the reference compounds. Subsequently, an induced *target vector* is obtained for each target compound by computing the vector coordinates that can best preserve its dissimilarity to all reference compounds. More specifically, for the ith target compound o_i, the following stress function is minimized:

$$stress = \sqrt{\sum_{j=1}^{|\mathbf{R}|} (d(o_i, r_j) - \hat{d}(x_i, \hat{r_j}))^2}. \tag{5.2}$$

In this equation, $d(o_i, r_j)$ is the dissimilarity value between target compound o_i and reference compound r_j. The variable $\hat{r_j}$ is the coordinate of the jth induced reference vector obtained by applying MDS to the reference compounds. The unknown coordinate of the ith target vector is x_i, and $\hat{d}(x_i, \hat{r_j})$ gives the Euclidean distance between x_i and $\hat{r_j}$. By minimizing the stress function with a global optimization algorithm, the coordinate x_i can be computed so that it best preserves the dissimilarities from target compound o_i to all reference compounds.

The L-BFGS-B algorithm from Zhu et al. (1997) is used to obtain the induced vectors for each target object by minimizing the above stress function. The algorithm is a general method for non-linear optimization with bounded constraints. It can minimize any non-linear function of n variables, each of which is bounded by lower and upper bounds. We utilize this method to obtain the induced vectors of compounds by bounding each element of the vector in the $[-1, 1]$ region after the coordinates of the reference vectors are centered. The correctness of this bound is because Tanimoto similarity coefficient which can only take values between 0 and 1.

5.2.2 User-defined Parameters

Two very important parameters in our modified version of the MDS algorithm are the number of dimensions D and the reference compound set. Large D values will not increase the minimum value of the stress function, and thus will never negatively impact the embedding quality. To maximize the embedding quality, our method can be used with conservatively large D values of over 100, but at the expense of longer computation times for both the embedding and the downstream similarity searches. Often D values above 100 may not be necessary for many types of molecular descriptors (*e.g.* physicochemical), due to their frequently high redundancy and correlation among each other (Agrafiotis and Xu, 2002). It might also be possible to take advantage of low intrinsic dimensionality of some similarity measures, using methods such as ISOMAP (Tenenbaum et al., 2000) and Locally Linear Embedding (LLE; Roweis and Saul, 2000). However, our tests using ISOMAP with atom pair descriptor also showed that D values above 100 may still be necessary for some molecular descriptors to achieve satisfactory accuracy.

The reference compound set is the second important parameter for our approach. The size of the set, R, directly controls the complexity of the stress function. Therefore, the larger the reference compound set, the longer the embedding time. Furthermore, both the size and the composition of the reference set have an influence on the embedding quality. This is because the induced reference vectors serve as similarity landmarks to position each target compound in the embedding space according to its dissimilarity profile to all reference compounds. To minimize ambiguous placements, the reference set should ideally be chosen as diversely as possible and cover the entire chemical space of the target compounds. However, our benchmark tests for structure similarity searching and clustering, have shown that randomly selecting reference compounds is sufficient to obtain robust results (see Section 6.4.3). This is partly because R is usually much larger than the number of D. Due to this redundancy, choosing one or several suboptimal reference compounds will not have a great impact on the embedding quality.

Moreover, it is important to identify the smallest values for R and D that can achieve satisfactory accuracy. Increasing either value will result in longer processing time. This does not only apply to the embedding of the compound database, but also to every query structure used in similarity searches, because both components have to go through the embedding process. This slightly offsets the time saving gained by the other parts of the algorithm. Agrafiotis et al. (2001) presented a neural network-based method that could reduce the embedding time of our method significantly. However, in our tests we were not able to achieve embedding qualities with this method that were sufficient for similarity search and clustering applications. This is shown in Table 5.1.

#Pivots	γ	Recall	
		top 10	top 100
15	1	13.0%	2.5%
	2	13.5%	3.6%
	5	14.4%	6.2%
	10	15.6%	9.7%
	30	19.4%	20.0%
700	1	12.0%	1.0%
	2	12.1%	1.3%
	5	12.2%	2.2%
	10	12.3%	3.6%
	30	13.0%	6.1%

Table 5.1: Embedding quality of a non-linear mapping network method benchmarked by recall rates of similarity searches. An embedding algorithm known as *non-linear mapping networks* (Agrafiotis et al., 2001) was implemented using the `nnet` package in R 2.10. The input to the network is a set of similarity keys, which are obtained by comparing a compound to a set of pivot compounds. The embedding experiments were performed with the NCI data set using different numbers of pivots and 3000 randomly selected compounds to train the network. Each compound was embedded into a 100-D vectors. With these embedding results, 1000 random similarity searches were carried out using different relaxation ratios γ. The table lists the average recall rates that were achieved in these tests. Both recall rate and relaxation ratio are defined in Section 6.4.3.

5.2.3 Comparison with Previous Methods

Compared to the quadratic time complexity of traditional MDS methods, our algorithm offers large time savings. Given similar distributions of pairwise dissimilarities and a fixed number of reference compounds, the running time of the above embedding algorithm is roughly linear to the number of compounds. Because each target compound is processed independently, our method can be easily parallelized on computers with multiple CPU cores or compute clusters. This is a very desirable feature when processing very large compound databases.

The related SPE method is in its current form less effective for processing very large data sets because of the challenges involved in implementing a parallelized version of this algorithm. Our tests with a publicly available implementation of the SPE algorithm show that its unparallelized compute times on very large data sets with millions of compounds are too long to obtain embeddings of high enough qualities to perform accurate nearest neighbor searches with acceptable recall rates (see Table 5.2).

Though reference compound set in our method is similar in concept to pivot objects employed by FastMap, the selection of reference compounds is far less stringent and far more resistant to the impact of outlier compounds. By randomly selecting reference compounds instead of rationally calculate them, the outlier compounds will not create unnecessary extra dimensions. (They are instead grouped together.) Due to the redundancy in the reference compound set, making suboptimal choice in selecting reference compounds will not have a great impact on the embedding quality.

#Dimensions	#Steps	Average Embedding Time	Recall	
		(in second per compound)	top 10	top 100
2	1,000,000	0.012	10.3%	2.6%
	5,000,000	0.054	10.2%	2.9%
	10,000,000	0.099	10.4%	3.0%
	32,000,000	0.286	10.2%	3.3%
	50,000,000	0.432	10.3%	4.4%
120	1,000,000	0.012	13.3%	21.4%
	5,000,000	0.063	55.5%	80.0%
	10,000,000	0.116	77.8%	91.2%
	20,000,000	0.179	88.7%	95.4%
	32,000,000	0.340	94.2%	97.6%
	50,000,000	0.535	97.0%	98.8%
Modified MDS (120D)		0.259	97.9%	96.4%
Modified MDS (120D)		0.146	95.9%	95.0%

Table 5.2: Embedding efficiency and accuracy of SPE-based embedding method compared to EI's modified MDS method. The table gives the average per-compound embedding times and recall rates for the NCI data sets when using the SPE algorithm or EI's modified MDS method. Embedding with SPE was carried out using different number of output dimensions and different number of steps (first and second column). Recall rates, as defined in Section 6.4.3, are given for searches of the 10 and 100 most similar neighbors of 1000 randomly selected query compounds. Relaxation ratio is set to 30 in all tests. All experimental parameters for the SPE method were chosen according the recommendations from Agrafiotis and Xu (2002). The corresponding results from our modified MDS algorithm are given in the last two rows.

5.3 Implementation

We implemented the modified MDS algorithm as C++ programs. Internally, it uses the L-BFGS-B library (Zhu et al., 1997) for the global optimization step in the embedding procedure. In our tests, we used `libmcs` and `libap` as implementations of MCS- and atom pair-based similarities, both described in Chapter 7 for computation of chemical similarity. To also include in our tests one of the most widely used fingerprint methods, we employed the fingerprint descriptors used by PubChem and utilized a fingerprint implementation provided by the NIH Chemical Genomics Center (2009). An embedding and evaluation workflow program has been created using the Python programming language. This allows us to perform a large number of automatic tests of our modified MDS algorithm in a number of settings.

5.4 Evaluation

5.4.1 Data sets and Testing Platform

To test the performance of the proposed method, benchmark comparisons for embedding were performed using the publicly available compound structures from NCI and PubChem. The NCI data set consisted of 260,071 compounds. After removing entries that did not generate any usable atom pair descriptors, 260,027 compound structures were used from this collection. From PubChem we used the structures from the PubChem Compound data sets. From this collection we selected two sets: one subset consisting of 2.3 million compounds with at least 5 non-hydrogen bonds (PubChem Subset) as well as the entire PubChem Compound library as of May 2009 with over all 19 million compounds (PubChem Compound).

The tests were performed on a Linux computer equipped with Xeon CPUs clocked at 2.4 GHz and 16 GB of RAM. For parallelized computations, a computer cluster was used with the exact same hardware configuration per compute node. Compute times are given as total CPU hours throughout the text.

5.4.2 Time Efficiency of Embedding

First, we evaluated the running time of our modified MDS algorithm for different parameters. For this, the compounds from all three data sets were embedded into a high-dimensional Euclidean space.

Time efficiency with respect to the size of the data set

Our results show that the time required for embedding increases almost linearly with the number of compounds in the library (see Table 5.3 in Supplementary Materials). The average time to embed one compound varies only slightly across the three data sets and is below 0.3 seconds using a practical parameter set. Because the embedding algorithm processes each compound independently after applying MDS to the reference compound set, it is trivial to further reduce its computation time by using a compute cluster. In our experiments, it took around 10 minutes to process the 260,027 compounds of the NCI data set using 87 CPUs. Similarly, the 19 million compounds of the PubChem Compound library could be processed

Data set	NCI	PubChem Subset	PubChem Compound
Size	260,071	2,288,680	19,629,027
Embedding time			
Total (hours)	18.72	179.24	1543.83
Per-compound (seconds)	0.259	0.282	0.284

Table 5.3: Processing time for embedding. The table gives the time required for embedding each compound library as well as the average time required per compound. All experiments were performed on the same hardware using the same parameters (R=300 and D=120).

in less than one day using 80 CPUs.

Time efficiency with respect to the embedding parameters

As discussed above, the number of dimensions D and and the number of reference compounds R are the two main factors that will affect the embedding time. To estimate their impact, we randomly selected from the NCI library seven reference compound sets with sizes ranging from 240 to 800 compounds. These reference sets were used in independent test runs of our embedding algorithm where D was set to a fixed value to study the influence of R on the total CPU time. Similarly, to model the impact of D, the value of R was fixed to three times the value of D and the total CPU time was collected using D values ranging from 120 to 260. These ranges were chosen to ensure high quality embedding, but also to keep the computation within manageable time limits. Our test results, shown in Figure 5.1 indicate that the total CPU time of our method grows linearly with R, and exponentially with D. Based on this time behavior, the values for R and D should be chosen as small as possible, but large enough to maintain an embedding quality that is sufficient for the downstream

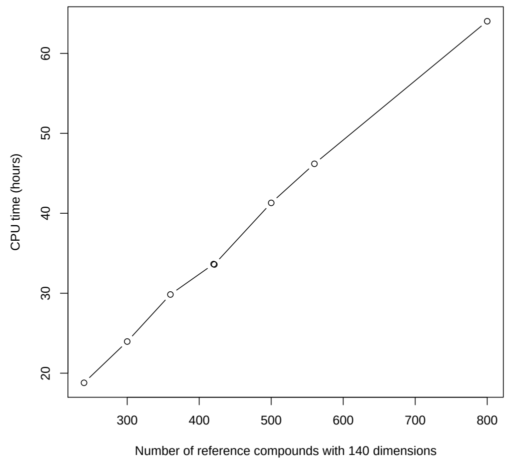
Parameters	Average Embedding Time	Recall	
	(in second per compound)	top 10	top 100
FACTR=10^{10}, PGTOL=0.001	0.526	98.0%	97.0%
FACTR=10^{12}, PGTOL=0.010	0.259	97.9%	96.4%
FACTR=10^{13}, PGTOL=0.040	0.146	95.9%	95.0%
FACTR=10^{16}, PGTOL=0.100	0.035	86.9%	85.7%

Table 5.4: Processing times for the embedding step using different optimization parameters. The table gives the average per-compound embedding times for the NCI data set using the indicated optimization parameters for the L-BGFS-B algorithm. All experiments were performed on the same hardware using the same parameters (retrieval of top 100 compounds by performing 3000-nearest-neighbor search in the embedding space. R=300 and D=120).

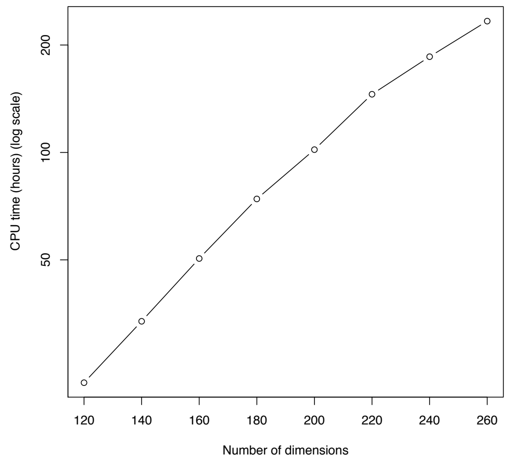
similarity search and clustering steps (see below).

Time efficiency and global optimization parameters

Solving the global optimization problem accounts for most of the embedding time. Therefore, the parameter choice of this step has a great impact on the time efficiency of the embedding step of our method. Using less stringent termination conditions for the L-BFGS-B algorithm, will typically reduce the embedding time, but at a cost of embedding quality. For example, the average time to embed one compound could be easily cut in half with a small loss in accuracy (see Table 5.4). Although the preprocessing of new data sets is time consuming, our method is relatively flexible with respect to adding new entries to an already preprocessed library. Because all entries are embedded independently, one can easily add new ones without repeating this process for the entire data set. This meets the work flow requirements of many large compound databases, where minor updates occur frequently, but major revisions are rare.



(a)



(b)

Figure 5.1: Embedding time and parameters. The graphs show the impact of the sizes of reference compound set R (a) and the number of dimensions D (b) on the total CPU time required for embedding the compounds from the NCI data set in a high-dimensional Euclidean space. In graph (b) R was set to three times the value of D.

5.4.3 Quality of Embedding

As pointed out by previous studies, embedding metric representations of objects in Euclidean space may reduce the accuracy of nearest neighbor searches (Fu et al., 2000). However, when the parameters for our embedding method are chosen properly, the method is accurate enough to be used in the pre-screening step of similarity search (as shown in Section 6.4.3). As a performance test of the embedding step, we randomly selected 10 million compound pairs from the NCI data set, computed their atom pair-based Tanimoto similarity coefficients and compared them with the corresponding distance values obtained from the induced vectors. The two data sets were highly correlated, as indicated by a Pearson correlation coefficient of 0.79. Additionally, the agreement among the data sets was evaluated by grouping the Tanimoto coefficients into ten similarity intervals from 0-1 using increments of 0.1. Subsequently, the distributions of the vector distances for each interval were plotted in form of box plots. In this representation a low degree of overlap among the boxes from adjacent intervals indicates a strong agreement between the two methods. The box plots obtained from our tests, shown in Figure 5.2, suggest only a moderate overlap among adjacent intervals, and only minor to no overlap among more distant intervals. This indicates a strong agreement among the two methods for the majority of the compounds with some inconsistencies at the interval boundaries, but for a much smaller number of cases. For example, for high Tanimoto similarities ranging from 0.8 to 1, the similarities of the corresponding vectors pairs $(1 - distance)$ fall all into a very narrow range of 0.75-1.0. The two methods also agree very well in the low similarity range. This is indicated by the fact that 98.17% of all compound pairs with a Tanimoto coefficient <0.4 are placed into a very similar range (<0.38) in vector space. Based on these results, our embedding method appears to preserve well separated ranges of Tanimoto coefficients in a robust manner.

5.5 Summary

This chapter demonstrates modified MDS, an efficient and effective algorithm that scales linearly to size of the dataset and can comfortably process large compound datasets with millions of entries. Built to avoid the quadratic time complexity of traditional MDS, modified MDS also outperforms many MDS alternatives in terms of scalability through straightforward parallelism and resistance to the impact of large numbers of outlier compounds in databases. It is shown to generate high-quality embedding that preserves groups of similar compounds in the embedding as neighboring points in the high-dimensional space. Such results are valuable in applications such as building an efficient pre-screening step for similarity search.

In addition, this chapter provides a detailed study on the user-defined parameters of the modified MDS method. After establishing the relationship of these parameters to the computational time, it identifies a set of parameters suitable to be used with atom pair-, PubChem fingerprint- and MCS-based similarity. A few observations suggest potential rules in choosing these parameters in other applications and databases.

In evaluating the quality of embedding, nearest neighbor search in the embedding space

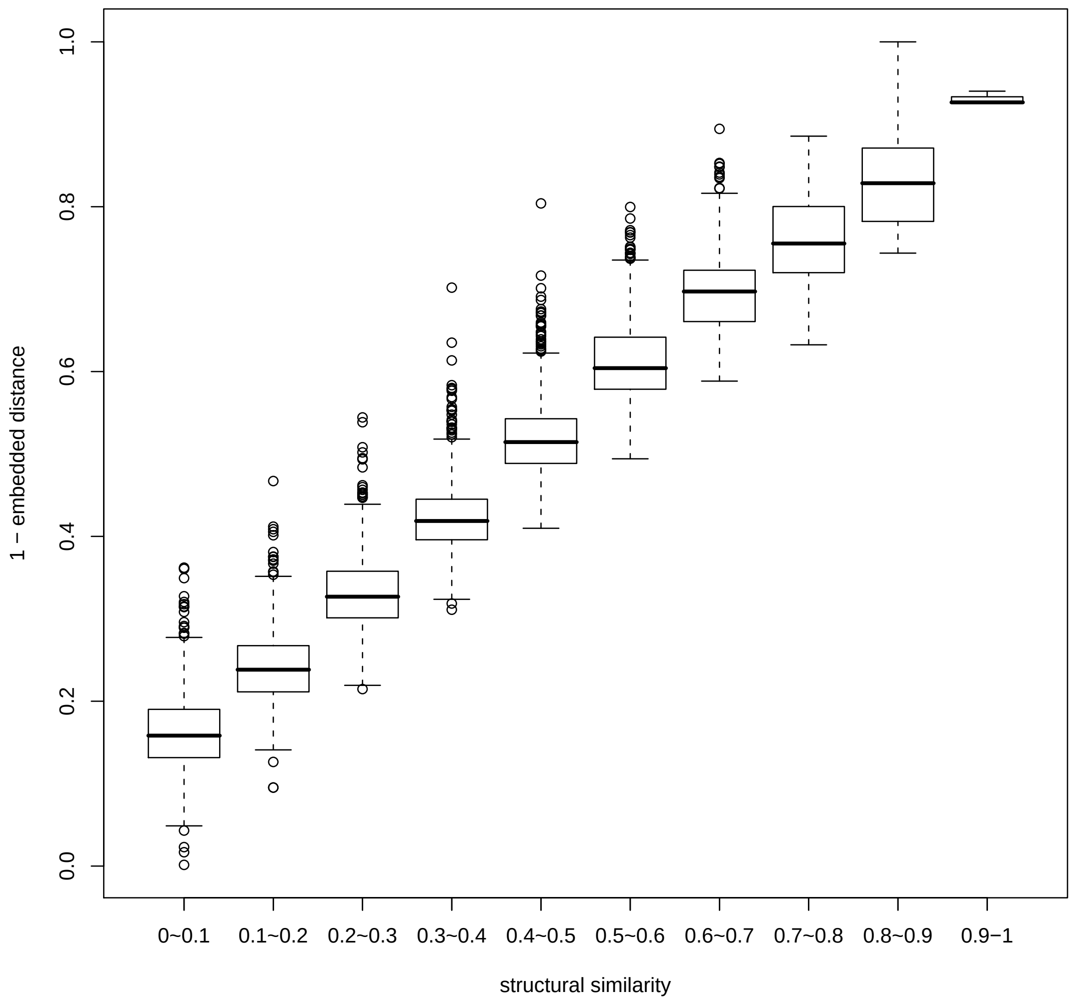


Figure 5.2: Embedding accuracy. The box plots compare the compound-to-compound distances obtained from the embedding method with the corresponding Tanimoto similarities for ten intervals ranging from 0-1.

is performed in lieu of chemical similarity search in the chemical space. The high recall rates and precisions achieved in real chemical databases suggest that the generated vector representations faithfully preserve the similarities between related structures. This and the computational efficiency collectively demonstrate that our high-fidelity embedding provides a valuable tool to generate high-quality vector representation of compounds that can benefit many applications.

Chapter 6

Accelerating Structure Similarity Search and Clustering

Similarity searching and clustering of chemical compounds by structural similarities are among the most important approaches in cheminformatics. Most algorithms available for these tasks are limited by their speed and scalability, and cannot handle today's large compound databases with million of entries. The practical value of MCS-based similarity introduced in Chapter 3, for example, is plagued by the intractable computation complexity of applying it in similarity search and clustering applications. This chapter describes an novel approach to accomplish fast similarity search and clustering by integrating high-fidelity embedding with locality sensitive hashing technique. Accomplishing significant efficiency enhancement in these applications is a success in itself and also highlights the usefulness of high-fidelity embedding technique introduced in Chapter 5.

6.1 Bottleneck in Similarity Search and Cluster Analysis

A variety of structure similarity search methods are available (reviewed by Willett et al., 1998). Unfortunately, they are often not fast enough for systematic analyses of very large compound collections with millions of compounds. This is because most of these methods sequentially compare a query structure against all entries in the database and then rank the results by a chosen scoring system, and thus the cost to perform similarity searches grows linearly with the size of the compound database. Therefore, more efficient and sophisticated search methods need to be developed to utilize the available chemical space efficiently.

Clustering of compound sets is essential on practically all stages of the discovery process of bioactive compounds (reviewed by Downs and Barnard, 2002). Commonly, structure similarity-based clustering utilizes the pairwise similarity measures generated by the above compound search methods to partition the data into discrete groups of similar compounds. An example is Jarvis-Patrick clustering, which is among the most widely used clustering methods in cheminformatics (Willett, 1987). Alternatively, they can be used to build hierarchical trees that represent the similarity relationships among all items in a compound data set. One of the main challenges in this area is the difficulty to cluster the millions of compounds structures that are currently available in the public domain. This is because many cluster analysis approaches multiply the complexity of a chosen similarity search method by the number of compounds in the data set. They often require the calculation of all-against-all similarities for the compounds under investigation and the computational cost grows quadratically with the size of the data set. Therefore, novel clustering methods need to be developed for exploring this vast chemical space efficiently.

A few methods have been proposed to accelerate nearest neighbor searching in compound databases. Agrafiotis and Lobanov (1999) reported an algorithm based on k-dimensional (or k-d) trees for diversity analyses. This problem is relevant to clustering (Bentley, 1975) and the method can possibly be adopted to nearest neighbor searching. It works by representing each chemical compound as a multi-dimensional vector and organizing the compound data set in a k-d tree structure to speed up diversity analyses. However, the vector representation used is very limited and the k-d tree cannot handle high-dimensional vector data. Later the authors proposed a new data structure called μ-tree to perform guided nearest neighbor search of compounds in general metric space (Xu and Agrafiotis, 2003). A μ-tree organizes the database in recursively partitioned Voronoi regions and represents these partitions as a tree. The method greatly reduces the number of similarity comparisons required for nearest neighbor search by applying an efficient branch and bound strategy. As a distance-based indexing method, the μ-tree approach does not require the compounds to be represented as multi-dimensional vectors. However, it requires the similarity measure used to be a metric. Swamidass and Baldi (2007) used upper bounds on similarity measures for structural fingerprints to reduce the number of molecules that need to be considered in nearest neighbor searches. Later, Baldi et al. (2008) employed tighter bounds to achieve further time savings. These latter two methods have been designed to be used only with fingerprint-based similarity measures. Recently, Dutta et al. (2006) utilized locality sensitive hashing (LSH) to hash chemical descriptors so that points close to each other in the descriptor space are also close to each other in the hashed space. They achieved similarity search that was at least 2 order os magnitude faster than traditional sequential scan-based search. However, their method is only applicable to structural descriptors with limited dimensions and Euclidean distance measure.

A variety of data structures and algorithms have been proposed to accelerate nearest neighbor search in multi-dimensional Euclidean space (reviewed by Bohm et al., 2001). These methods, often referred to as Multidimensional Access Methods (MAMs), have not been widely used in similarity searching and clustering of chemical compounds. The reason for this may be the popularity of non-Euclidean similarity coefficients in chemical structure similarity searching, which are not immediately compatible with MAMs. Moreover, Fu et al. (2000) reported that the embedding of the generic metric space into multi-dimensional space can introduce considerable inaccuracy in nearest neighbor search applications.

6.2 EI-Search

6.2.1 Assisted Nearest Neighbor Search

We designed an efficient similarity search algorithm that accomplishes dramatical acceleration by taking advantage of a spatial index based on Locality Sensitive Hashing (LSH) (Datar et al., 2004; Gionis et al., 1999; Shakhnarovich et al., 2005; Lv et al., 2007). Unlike Dutta et al. (2006)'s study, which also employs LSH, EI-Search capitalizes on our high-fidelity embedding method described in the previous chapter to achieve high degree of compatibility with similarity measures and scalability in working with large compound dataset with millions of entries.

The algorithm works by assisting structure similarity search with a proper companion nearest neighbor search. After the compounds are embedded into the D-dimensional Euclidean space using our high-fidelity embedding method, a structure similarity searches will be prefixed with an assistant nearest neighbor search in the embedding space. For this, the query compound is embedded into the same D-dimensional Euclidean space. Subsequently, the corresponding induced vector is used to perform a nearest neighbor search in the embedding space. The result of the assistant search will be used to retrieve a small candidate compound set from the original compound dataset. The actual structure similarity search will then take place in the candidate set instead of the whole compound dataset.

This search method offers great time savings and flexibility for similarity searches because of two major reasons. First, compared to other similarity measures used for comparing compound structures, Euclidean distances can be calculated very efficiently. Therefore, using an assistant search with small overhead to avoid sequential scan of the whole dataset can often dramatically reduce search time. Second, many MAMs can be employed in the nearest neighbor search to further improve its time efficiency.

6.2.2 Further Speedup through MAMs

Although many MAMs have been proposed to speed up the nearest neighbor search problem, most of them are affected by the dimensionality problem, often referred to as *the curse of dimensionality*. This is the phenomenon that for various geometric search problems, including nearest neighbor search, the best algorithms known have performance trends that degrade exponentially with an increase in dimensionality (Weber et al., 1998). Our tests also confirmed that the SR-tree method (Katayama, 1997) becomes slower than sequential scans of the data set as soon as D is set to values greater than 100. However, our tests also indicated that the dissimilarities between compound structures can only be preserved reliably with large D values of at least 100. Thus, it is important for our method to combine high-dimensional embedding and an indexing method that are both insensitive to the dimensionality problem.

It has become increasingly popular to use the approximate nearest neighbor search in high-dimensional space to avoid the dimensionality problem. One of these approximation approaches utilizes a spatial index using Locality Sensitivity Hashing (LSH) to perform fast nearest neighbor search in Euclidean space. LSH uses a family of hashing functions to map database objects into buckets. It is designed to join related items based on a given similarity measure with high probability. Accordingly, many hashing functions vary with respect to their similarity measures. For example, Gionis et al. (1999) introduced in the original LSH paper a bit sampling-based LSH approach that uses the Hamming distance measure. Datar et al. (2004) proposed an LSH scheme for p-stable distributions along with Euclidean distances. However, so far no hashing functions have been developed for the similarity measures that are commonly used for comparing compound structures. This limitation can be addressed by embedding compounds in Euclidean space and building for them a spatial index using induced vectors in embedding space.

Taking advantage of the above embedding and the LSH-based spatial indexing approaches, we designed an efficient approximate compound structure similarity search algorithm. This algorithm is named *EI-Search* after its two key components: *embedding* and *indexing*. The algorithm first pre-processes the compound data set by embedding it into a high-dimensional Euclidean space and generating a spatial index using LSH. When searching for k compounds that are most similar to a given query compound, a two-step approach is employed to reduce the error introduced in the embedding process and the approximate nearest neighbor search. First, the query compound is embedded in the same Euclidean space. The resulting induced vector is then used in an index-assisted nearest neighbor search of the embedding space to retrieve a candidate set consisting of $\gamma \cdot k$ vectors that are most similar to it. The *relaxation ratio* γ is a user-defined parameter that controls the trade-off between processing time and search accuracy. Larger values for γ result in larger candidate sets and possibly higher accuracy, but at the cost of longer search times. Second, a refinement step applies exact structure similarity searches to the candidate set obtained in the first step. This allows the selection of the final k compounds that are most similar to the query structure.

6.2.3 Advantages of EI-Search

Compared to commonly used structure similarity search methods, EI-Search has several advantages. The most important ones are its time efficiency and compatibility with a wide range of similarity measurements. This makes the method potentially useful for accelerating similarity searches of a variety of data objects that are of relevance to many life science and non-life science areas.

6.3 EI-Clustering

In addition to similarity searching, Euclidean space representations can be used for clustering large data sets very efficiently. For example, spatial join can be used to perform single linkage clustering by finding all vector pairs which are separated by a distance below a given threshold (Brinkhoff et al., 1993). The nearest neighbor information required for Jarvis-Patrick clustering can also be obtained very efficiently in Euclidean space by using an efficient algorithm for the all-nearest-neighbors problem (Vaidya, 1989).

In this paper, we introduce a new clustering method, *EI-Clustering*, that takes advantage of the accelerated search speed provided by EI-Search to cluster very large sets of chemical compounds under the Jarvis-Patrick clustering framework. Jarvis-Patrick clustering requires a nearest neighbor table, which consists of p nearest neighbors for each compound in the data set. This information is then used to join compounds into clusters that share at least m nearest neighbors. The values for p and m are user-defined parameters. In case of EI-Clustering, the EI-Search method generates the nearest neighbor information for each compound in the data set. The resulting nearest neighbor table is then used as direct input for Jarvis-Patrick clustering. When clustering very large data sets, EI-Clustering is particularly efficient due to its fast computation of the nearest neighbor table.

6.4 Evaluation

6.4.1 Implementation

We implemented the EI-Search and EI-Clustering algorithms as C++ programs. Internally, they use the modified MDS implementation introduced in Chapter 5. For the LSH-based spatial indexing, we utilized the *lshkit* library that implements the MPLSH algorithm, a variant of LSH that requires fewer hash tables and offers large space savings as well as shorter query time (Lv et al., 2007). Similar to evaluation of our high-fidelity embedding method, we used `libmcs` and `libap` as implementation of MCS- and atom pair-based similarities, and utilized the PubChem fingerprint implementation provided by the NIH Chemical Genomics Center (2009). Jarvis-Patrick clustering was performed with a custom program implemented in C++.

6.4.2 Data sets and Testing Platform

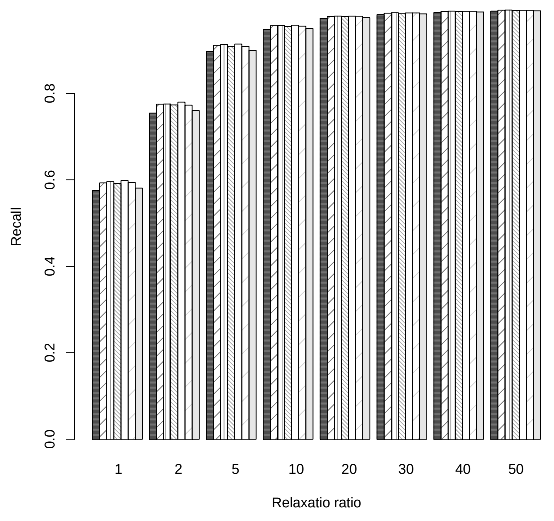
We used the same data sets and testing strategy as in Chapter 5.

6.4.3 Accuracy of EI-Search

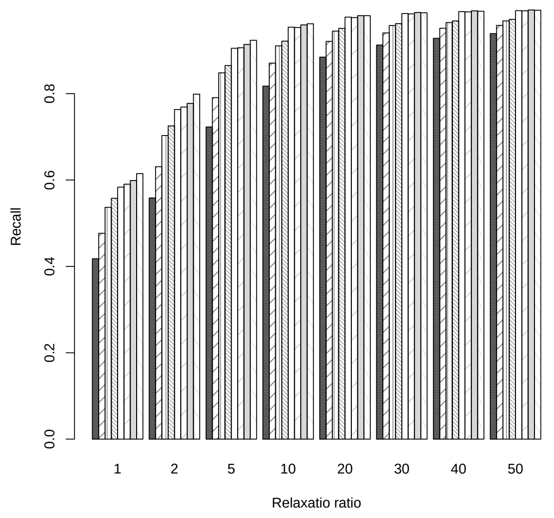
To further examine the accuracy of the EI-Search method for compound similarity searching, we implemented an EI-Search program and tested it with different parameters. The accuracy of EI-Search is measured by the *recall rate*. When retrieving the k most similar compounds to a query structure, the recall rate is defined as the percentage of compounds obtained with EI-Search that are also returned by sequential search methods. To evaluate the impact of the different parameters used by EI-Search, the program was run using different values of D, R, k and γ. In each run, the recall rates for 1000 random queries from the NCI data set were calculated.

First, we compared the recall rates when searching for the k most similar compounds with different R and γ values while the values of k and D were fixed. Figure 6.1a shows the results for constant k and D values of 100 and 140, respectively. The results indicate that increasing γ from 1 to 50 results in a significant improvement of the recall performance, while this effect starts to plateau off at values of greater than 20. With respect to the number of reference compounds R, the recall rate increases from values 240 to 420. Based on these results, a good empirical choice for R is between two and three times the value of D. Another observation is the fact that the effect of R diminishes for γ values above 20. In other words, large enough γ values can compensate a suboptimal choice of R. For example, when γ was set to 20, and k to 100 and D to 140, the best and worst recall rates were 97.87% and 97.35%, respectively. This corresponds to a difference of only 0.52%.

Similarly, we investigated the correlation between the recall rates and the values of D



(a)



(b)

Figure 6.1: Recall rates of the EI-Search method. Panel (a) provides the recall rates of EI-Search using different relaxation ratios γ and numbers of reference compounds R. Both recall rate and relaxation ratio is defined in Section 6.4.3. Panel (b) shows the same data for variable γ and D values. In panel (a) the chosen R values for each γ were: 240, 300, 360, 420, 500, 560, and 800, and 260 (from left to right). In panel (b), the D values were: 40, 60, 80, 100, 120, 140, 160 and 180.

and γ. According to the results from the previous tests, the R values were always set to three times the values of D. When searching for the k most similar compounds the recall rates were collected for different D and γ values while the value of k was fixed. Figure 6.1b in the Supplementary Materials shows the results for a k value of 100. These results indicate that the recall rates consistently improve with the number of dimensions. This effect is much stronger for smaller γ values. For example, when k was set to 100 and γ to 1, then the recall rate could be improved from 58.35% to 65.57% for D values of 120 and 260, respectively. For large γ values above 100 this effect is again much less pronounced. While larger γ values will result in an increase in processing time, their impact is less severe than increasing the value of D. Accordingly, we chose in the subsequent experiments the D values as small as possible and the γ values as large as necessary to maintain both high accuracy and time efficiency of the method.

6.4.4 Time Efficiency of EI-Search

While maintaining high recall rates, EI-Search was able to greatly reduce the time for performing structure similarity searches in large compound databases. To examine the time efficiency of EI-Search, 1000 random queries were performed on each of the three data sets using first the EI-Search program and then exhaustive sequential searches with the atom pair and fingerprint similarity search programs. For each comparison we used for EI-Search the same descriptor type as for the sequential search methods. The query compounds were randomly selected from the data set. To obtain realistic search results, the 100 most simi-

Table 6.1: Performance tests of EI-Search. Search times and recall rates are listed for searching three large compound sets with EI-Search and the sequential search methods. The same descriptor type was used for each comparison pair. The experiments were performed on the same hardware using the same embedding and relaxation parameters (R=300, D=120 and γ=30). The LSH parameters were supplied by *lshkit*.

Data set	NCI	PubChem Subset	PubChem Compound	
Descriptor type	Atom Pair	Atom Pair	Atom Pair	Fingerprint
Average search time (seconds)				
Sequential search	0.800	11.570	93.121	19.658
EI-Search	0.067	0.170	0.427	0.499
Recall of EI-Search (%)				
Mean	99.95	99.60	97.38	96.32
Standard deviation	0.44	1.82	5.61	11.54

lar compounds with a minimum similarity of 0.5 were retrieved for each query. Atom pair descriptors combined with Tanimoto coefficients were used as similarity measure. For the PubChem Compound library, we also included in the tests the Tanimoto similarities of PubChem's fingerprints. According to the above parameter optimization results, we used in all tests the following settings: R=300, D=120 and γ=30. The obtained search times and recall rates are listed in Table 6.1.

Several conclusions can be drawn from Table 6.1. First, in comparison to the highly accurate atom pair method, EI-Search achieves in the atom pair descriptor tests very high recall rates ranging from 97.38% to 99.95%. In comparison to the fingerprint method, the recall rate of the corresponding EI-Search is slightly lower with 96.32%. The reason for this reduction may be the fact that fingerprints provide less accurate similarity measures than atom pairs (Chen and Reynolds, 2002). This could result in less robust rankings of the nearest neighbor search results, and therefore a slightly lower recall rate is reasonable. Second, EI-

Search provides significant time savings for nearest neighbor searches. For example, the average time required to search the $>$19 million compounds of the PubChem Compound data set is reduced for the atom pair approach from over 93 to less than 0.5 seconds, and for the fingerprint method from over 19 to less than 0.5 seconds. This corresponds to accelerations by our EI-Search method of over 200 and 40 folds, respectively. Third, while the search time for the exhaustive sequential search methods increases linearly with the size of the databases, this increase is less than linear for the EI-Search method. For instance, searching the PubChem Subset data set with EI-Search takes on average only 2.5 times longer than searching the NCI data set, that has one ninth of the size of PubChem Subset. Finally, another outstanding feature of EI-Search is the fact that its search speed is much less impacted by the complexity of the similarity measure used for database searching than this is the case for the exhaustive methods. For instance, the switch from the fingerprint similarity measure to the more computationally expensive atom pair similarity measure results only in a minor increase of EI-search's query time, while it is a more than four fold increase for the exhaustive sequential search methods.

6.4.5 Accuracy and Time Efficiency of EI-Clustering

To test the performance of our EI-Clustering method for partitioning large compound sets, we clustered all three compound sets with the Jarvis-Patrick algorithm. The required nearest neighbor tables were generated with EI-Search and the exhaustive sequential search methods. EI-Search was run with the same embedding and searching parameters (R=300, D=120 and

Table 6.2: Performance tests for EI-Clustering. The table compares the time and accuracy performance of EI-Clustering with Jarvis-Patrick clustering when using exhaustive search methods for generating the required nearest neighbor information. The compute time is given in hours of total CPU time. The agreement among the clustering results is given in the last row in form of Jaccard partition coefficients. Clustering of the PubChem Compound data set was not possible with the exhaustive search methods due to their insufficient performance on this large data set.

Data set	NCI	PubChem Subset	PubChem Compound	
Similarity Measure	Atom Pair	Atom Pair	Atom Pair	Fingerprint
Total clustering time (hours)				
Jarvis-Patrick	72.9	7355.6	N/A	N/A
EI-Clustering	3.5	92.2	1517.2	2869.71
Jaccard Coefficient	0.9913	0.9887	N/A	N/A

γ=30) as in the previous section. The LSH parameters were slightly changed to achieve higher accuracy. The process of generating the nearest neighbor tables was parallelized on a computer cluster. For each clustering result the total search time was calculated for all utilized CPUs. To measure the agreement among the clustering results, we computed the Jaccard partition coefficient for each pair of clustering results (Table 6.2). Jaccard coefficients close to 0 indicate low similarities and values close to 1 high similarities among the evaluated cluster sets.

The results in Table 6.2 show that the EI-Clustering method can dramatically reduce the processing time of the Jarvis-Patrick clustering approach while maintaining a high level of agreement with the results obtained by Jarvis-Patrick clustering with exhaustive nearest neighbor search methods (Jaccard coefficients $>$0.98). The total CPU time to process over 2.3 million compounds from the PubChem Subset could be reduced from over 306 days to just 4 days. This is a major improvement, because it makes it feasible to cluster millions of compounds in a few days on a regular workstation or in a few hours when a small computer cluster is available. By running EI-Search on 80 CPUs on a computer cluster, we were able to cluster the entire PubChem Compound library in only one day. Because EI-Clustering spends most of the time running EI-Search which runs in time sub-linear in the size of the data set, the compute time of EI-Clustering is sub-quadratic to the size of the data set. Therefore, EI-Clustering scales much more efficiently to larger data sets than traditional methods. The superior speed of EI-Clustering comes at the cost of a larger memory footprint compared to the other methods. Most of the memory is consumed by its LSH index. For example, when clustering the 19 million PubChem Compound library, the LSH index requires around 13GB of memory. Considering the performance of today's research workstations, this memory requirement appears to more than manageable.

6.5 Discussions and Summary

In this chapter, we have presented EI-Search and EI-Clustering as efficient methods for accelerating structure similarity searches and clustering of very large compound data sets. The acceleration is achieved by applying *embedding and indexing* techniques to represent chemical compounds in a high-dimensional Euclidean space and to employ ultra-fast pre-screening of the compound data set using LSH-assisted nearest neighbor search in the embedding space. Our tests show that the method can dramatically reduce the search time of large databases, by a factor of 40-200 fold when searching the 100 closest compounds to a query. Recently published acceleration methods achieved only a 5.5-fold reduction in search time when using a Tanimoto threshold of 0.8 and up to 20-fold with a relatively restrictive threshold of 0.9 (Swamidass and Baldi, 2007; Baldi et al., 2008). Another limitation of these methods is their narrow utility spectrum that is currently restricted to fingerprint-based searches. In contrast to this, the EI-Search framework is designed to be useful for a wide spectrum of similarity measures. After embedding, EI-Search will run in most cases with comparable time efficiencies independent of the complexity of the similarity measure. This can be particularly useful for accelerating searches that use much more accurate, but computationally very expensive similarity measures, such as maximum common substructures or 3D approaches (Raymond et al., 2003; Willett, 2005; Cao et al., 2008).

By taking advantage of the fast similarity search speed of EI-Search, we developed EI-Clustering into an effective clustering method for very large data sets. The method accelerated the clustering of the three test data sets used in this study by 20 to 80 folds. Most importantly, the EI-Clustering made it feasible to cluster data sets of almost 20 million entries within acceptable time limits. Due to its subquadratic running time, the EI-Clustering method should scale well enough to cluster even larger data sets with tens or even hundreds of millions of objects.

Chapter 7

A Toolkit for Studying Similarity Measures

This dissertation has pursued a comprehensive quantitative study of MCS as an alternative chemical similarity measure and investigated its computational efficiency, effectiveness in retrieving bioactive molecules, integration with machine learning technologies, and time efficiency in similarity search and clustering applications. Along the course of this study, many tools and software packages have been created. We have designed them to be highly-reusable and user- and developer-friendly in hope it can facilitate further research in chemical similarity measures. This chapter will provide a brief introduction to some of the packages.

7.1 ChemmineR

7.1.1 Overview

To search and analyze the vast amounts of available compound and screening information, and to assemble diverse screening libraries, efficient compound analysis tools are a critical enabling resource. Unfortunately, most of the available software in this area is only commercially available, and open-source approaches are still the exception (*e.g.* OpenBabel, JoeLib, Guha et al., 2006). The longterm goal of the *ChemmineR* project is to narrow this resource gap by providing free access to a flexible and expandable open-source framework for the analysis of small molecule data from chemical genomics, agrochemical and drug discovery screens.

The development of compound analysis software for the statistical environment and programming language R has many obvious advantages (R Development Core Team, 2008). To name just a few: (1) R is one of the most widely used data mining environments used in bioinformatics. (2) The software and its associated packages are available for all common operating systems. (3) CPU and memory intensive calculations can be computed in high-performance languages, like C. (4) The data objects and base functions available in R are extremely efficient for typical compound and screening data mining routines. (5) Finally, an unmatched spectrum of data mining resources is available in R, such as extensive graphics utilities, powerful statistical functions, and a wide variety of algorithms for clustering and machine learning tasks.

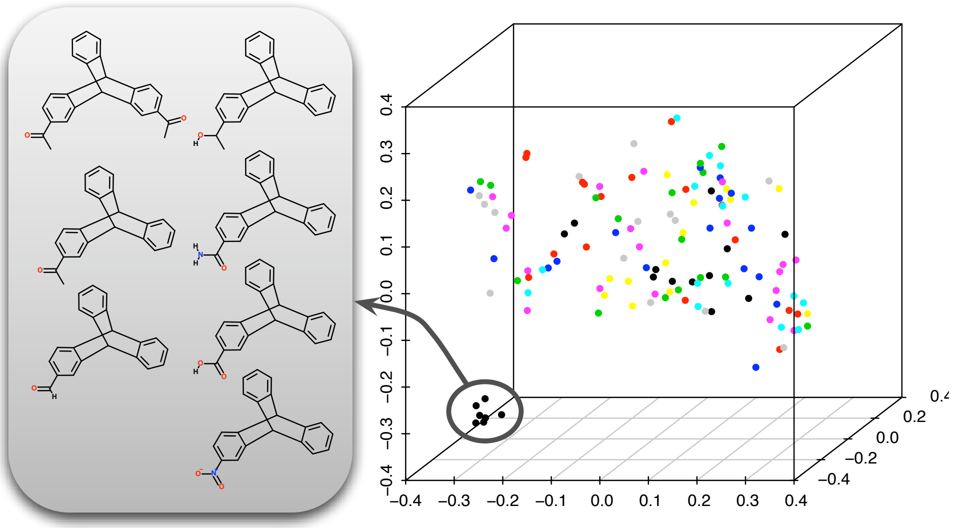


Figure 7.1: Sample 3D scatter plot of a clustering result and online visualization of the corresponding compound structures using *ChemmineR*.

The current release of the *ChemmineR* package contains functions for 2D structural similarity comparisons between compounds and similarity searching against compound databases. Both methods use the highly accurate atom pair approach for scoring structural similarities (Carhart et al., 1985; Chen and Reynolds, 2002). In addition, the package provides various functions for clustering entire compound libraries and visualizing clustering results and compound structures. All functions and data objects are well integrated into the existing infrastructure of the R environment. An overview of the *ChemmineR*-specific functions is provided in the online instructions and the PDF manual of this project.

7.1.2 Compound Import and Descriptor Calculation

Compound structures are imported into *ChemmineR* in the generic Structure Definition File (SDF) format. Single compounds are imported by providing an SDF with one compound structure, whereas entire compound databases are imported by providing all SDF formatted structures concatenated in one batch file. The atom pair descriptors are calculated during the SDF import and stored in a searchable descriptor database as a list object (Chen and Reynolds, 2002). Because the calculation of descriptors for thousands of compounds can be time consuming, functions are provided to efficiently store and reload existing descriptor databases as binary files. Custom compound databases can be generated with a SDF subsetting function.

7.1.3 Compound Similarity Searching

A search function is available to perform structure similarity searches against the generated atom pair descriptor databases. The default setting uses the Tanimoto coefficient as similarity measure (Holliday et al., 2003). The search function can return all entries in a compound database sorted by similarity score. Alternatively, the search results can be limited by a similarity threshold or a desired number of similar compounds. To view the compounds structures of search results, the function can automatically upload the returned compounds to the online ChemMine service where they are rendered into chemical structure images (see below).

The compound search function calls internally a generic function for calculating atom pair-based similarities between compound structures (Chen and Reynolds, 2002). This similarity function can be used to calculate pairwise compound similarities or to design custom subroutines for similarity scoring and searching.

7.1.4 Compound Clustering

Structure-based clustering is required for many analysis steps of compound libraries and HTS data sets. *ChemmineR* provides a novel binning clustering method that is optimized for these compound analysis tasks. The algorithm uses single-linkage clustering to join compounds into similarity groups, where every member in a cluster shares with at least one other member a similarity value above a user-specified threshold. The algorithm is optimized for speed and memory efficiency by avoiding the calculation of an all-against-all distance matrix. This is achieved by calculating on-the-fly only the distance values that are required in each clustering step. Because an optimum similarity threshold is often unknown, a series of binning clustering result can be calculated simultaneously for several user specified thresholds. Cluster results for several thresholds can be calculated almost with the same speed as for a single threshold by issuing multiple clustering processes simultaneously, but calculating the required distances only once.

If desired by the user, then the binning clustering function can generate an all-against-all distance matrix for clustering compound sets with many other classification algorithms available in R, such as hierarchical clustering or K-means. In addition, *ChemmineR* provides an interactive wrapper function for Multidimensional Scaling (MDS) clustering. The online instructions provide several examples on how to cluster compound sets in *ChemmineR* with external clustering utilities. These include examples for using the fully interactive visual data mining tool RGGobi (Lang et al., 2007).

7.1.5 Interacting with ChemMine

The *ChemmineR* package provides bidirectional communications with online tools and databases available on the ChemMine portal (Girke et al., 2005b). The service allows users to view and compare any combination of compound structure images in large batches via a standard internet browser along with extensive compound annotation information and custom data tables for basic QSAR analyses (Gedeck et al., 2006). This includes structure viewing of extensive similarity search results generated by *ChemmineR*. All online viewing utilities can be accessed directly from R simply by selecting the online viewing argument in various *ChemmineR* functions or issuing a dedicated data exchange function. Uploading compound data to the ChemMine interface gives the user access to many additional tools available on ChemMine's online compound analysis WorkBench. This includes the calculation of physicochemical property descriptors (Guha et al., 2006), inter-conversions between different structure formats (*e.g.* SMILES and SDF), searching of the millions of drug-like compounds available in ChemMine, and easy access to published bioactivity and target protein information.

7.1.6 Future Updates

The *ChemmineR* framework will be expanded in the future by adding many more useful compound and screening data analysis functions. These include functions for (1) calculating physicochemical properties of compounds directly in R, (2) local similarity searching based on most common substructures (MCS, Raymond et al., 2002a), various utilities for QSAR modeling (Gedeck et al., 2006), and (4) wrapper functions for interfacing directly with other open-source small molecule analysis projects, such as OpenBabel and JoeLib (Guha et al., 2006; O'Boyle et al., 2008). Extensive user tutorials and download options of different package versions will be available from the *ChemmineR* project site and from the BioConductor site (Gentleman et al., 2005).

7.1.7 Summary

ChemmineR is the first open-access compound mining toolkit for the popular statistical environment R. The package provides flexible functions for powerful structure similarity searching, compound clustering, screening library management and online batch viewing of chemical structures. Users with a basic understanding of the R environment can easily customize the provided functions and design sophisticated compound library analysis pipelines that utilize the extensive statistical and machine learning resources available in R.

7.2 `libap`

7.2.1 Overview

An important aspect of the ChemmineR project is its use of programming language R, which brings a number of advantages, such as the high degree of familiarity of the target audience and the vast availability of existing data mining resources in the programming environment and user community. However, performing computation intensive calculations and working with large compound dataset can result in impractically low efficiency due to the intrinsic limitation of the R programming language. Therefore, we create `libap`, a highly efficient implementation of the most important functions in ChemmineR using C++ programming language. Similar to ChemmineR, `libap` supports calculation of atom pair descriptor, similarity comparisons between compounds, and similarity search against compound databases. The package exposes the functionality as a standalone package, an R extension package, a Python module, and a PostgreSQL extension. The R extension package is particularly interesting because of its seamless integration with ChemmineR package. Support for other programming languages, called *biddings*, can be easily added due to the use of an automatic binding generation tool, SWIG (Beazley, 1996).

7.2.2 Compound Import and Descriptor Calculation

The package allows import of compound structure data in various formats, including the commonly used SDF and SMILES formats. This takes advantage of availability of Open-

Babel on the users' computer and automatically supports additional formats that OpenBabel accepts. In case OpenBabel does not exist on the user platform, the package activates its own parser that can robustly handle SDF format.

The imported structures are converted to atom pair descriptors and stored in a compound database. The atom pair descriptors are calculated using highly optimized C++ code and can process millions PubChem Compound chemical structures in an hour in our tests. By utilizing file system-backed database, there is no extra limit to the size of the database besides the constraint of the underlying file system itself. To assist different types of applications, two different implementations of file system-backed structure descriptor database are included, a simple list style database optimized for sequential scan and an associative array style one based on Berkeley DB (Olson et al., 1999) that is ideal for random access. While the former one is suitable for traditional similarity search routines based on exhaustively comparing the query to all database structures, the latter is useful in index-assisted searchers such as EI-Search introduced in Chapter 6.

7.2.3 Similarity Comparison and Search

Using the generated compound structural descriptor database, similarity comparison and search functions are available to measure inter-molecule similarity and perform structure similarity using atom pair descriptor-based structure similarity. In addition to one-to-one structure comparison, `libap` also allows all-pairwise comparisons between two compound datasets. This is useful in generating pairwise distance matrix and in comparing all database compounds to the set of reference compounds in the high-fidelity embedding algorithm (Chapter 5). Similarity search allows the retrieval of k most similar compounds to a query structure. If a candidate set is available, such as the one generated by an assistant nearest neighbor search in EI-Search, it can be supplied to avoid a sequential search and to greatly reduce the search time.

7.2.4 Binding for Python

Python has become a very popular programming language in the cheminformatics community (DeLano, 2005; O'Boyle et al., 2008). By providing access to Python programs, `libap` reaches to many researchers building their workflow in the Python ecosystem and also allows a high level of extensibility and flexibility thanks to the dynamic nature of the programming language.

`libap` exposes to the user the functions of construction and read-write access of structural descriptor databases, fine control of individual descriptor, and similarity comparison. Users can create a new database by parsing a compound structure file in supported formats, and open an existing database file for random access or traversal with an iterator. Functions are provided to modify the database, for example, by appending new descriptors.

Using these building blocks, many advanced functions can be created with a few lines of code. For example, taking advantage of the iterator of the database and the similarity comparison function, the package document demonstrate a custom similarity search routine in seven lines of code. In another example, one can build a sub-database of compounds with desired properties using the function that appends new descriptors to a database. The exposed functionality of the package is comprehensive enough to enable a number of possibilities in extending capabilities and building cheminformatics toolkit that satisfies customized needs.

7.2.5 Binding for R Programming Environment

The binding for R programming environment is highly similar to the python binding because both are generated with the automatic tool SWIG. However, the goal of the R binding is to enable seamless integration with ChemmineR and therefore zero extra learning for users of ChemmineR. Therefore, the R binding is created as a companion package for ChemmineR. When installed and detected by ChemmineR, it automatically boosts the performance of user program based on ChemmineR. There is no need to change any part of the user program to take advantage of the higher efficiency of `libap`.

7.2.6 Extension for PostgreSQL

As compound data become more versatile, many users have adopted relational database management system (RDBMS) such as PostgreSQL a centralized storage system for compound structure and annotation information. To avoid the extra maintenance effort of a separate descriptor database, we build a PostgreSQL extension to allow atom pair descriptor data to be stored directly in the PostgreSQL database alongside with the other compound information, and enable similarity search to be performed using standard SQL statements.

This extension adds two functions to be used in SQL statements, one for calculating atom pair descriptor using an SDF string, and the other for computing the similarity between a pair of descriptors. A typical user case is to use a trigger to invoke the former function to calculate the atom pair descriptor and fill a column with the result when a new compound is inserted. Similarity search can make use of the latter function and data in the column for atom pair descriptor.

7.3 `libmcs`

7.3.1 Introduction

We implemented algorithm described in Chapter 3 in ANSI C, and packaged the reusable code in the form of a software library called `libmcs`. The advantages of using ANSI C as the implementation language include high portability and efficiency and also the possibility to expose the functionality to higher-level scripting languages. Similar to `libap`, `libmcs` also includes API for several scripting languages including Python and R.

7.3.2 Usage

The library is designed to be small, easy to use and yet extensible. Using `libmcs`, users pass in two molecules represented using a supported format, such as SDF format, and a call to the single MCS function returns the maximum common substructure match. Internally, a molecule is represented using a simple labeled graph. Users could add support to more file formats and add matching flexibility by manipulating the mapping from the molecule file to the representation graph. For example, by labeling the charged atoms different from the uncharged atoms in the representation graph, the user can disallow matching between charged to uncharged atoms in the MCS match. In combination with other tools such as OpenBabel or Pybel (O'Boyle et al., 2008), it can yield further flexibility such as configurable matching behavior between aromatic and non-aromatic bonds.

The library also has time-out support to provide a reasonable workaround to search MCS between large molecules that would take too long to find the precise MCS. A parameter can be set to control the maximum time the MCS search can take. When that limit is hit, the search routine will stop the search process and return the best solution it has at that moment. In this case, though the program often does not return the actual MCS, in many applications such as similarity comparisons, the returned result is a good enough estimation.

`libmcs` has been heavily tested and used in several internal projects. For example, it is employed in generating data shown in Chapter 3 and building an online web-based MCS solver. It has also been downloaded and used by several third party research organizations.

7.4 High-Fidelity Embedding Toolkit

7.4.1 Introduction

We provide an implementation of the high-fidelity embedding techniques introduced in Chapter 5 as a customizable toolkit. The core embedding functionality is implemented as a C++ program. However, to provide maximum flexibility and extensibility, the toolkit consists a set of programs written in different languages. Each of these components focus on one self-contained sub-process of embedding, and they together are orchestrated by a user-facing script written in Python to convert compound representations into embedding results.

In its original form, we offer the capability to embed a compound set into a high-dimensional Euclidean space using atom pair- or PubChem-based Tanimoto similarity. Using the included LSH implementation, users then can perform highly efficient index-assisted similarity search or generating nearest neighbor table for downstream cluster analysis. The data presented in Chapter 5 and 6 are generated using this toolkit.

7.4.2 Customization

To exploit the full potential of our high-fidelity embedding technique, we design the toolkit as a generic utility that is neutral to the choice of similarity measure. By plugging a custom similarity measure implementation, users can power the same embedding technique with a custom similarity measure and even apply the tool in processing non-compound data. The process involves customization of a single program component that is used to generate dissimilarity table. Users are given the freedom to use any programming language or environment to implement this custom program.

7.5 EI Web Application

7.5.1 Introduction

Chapter 6 illustrated EI-Search and EI-Cluster as methods to greatly improve the efficiency of similarity search and cluster analysis of large compound database. To put the power of this method into real applications, we combined it with one of the largest public compound datasets to create the EI web application. EI Web application provides ultra fast similarity search for over 19 million compounds in PubChem Compound dataset. Most queries that retrieve 200 most similarity compounds are answered within a second. Being an approximate method, it could send user queries to PubChem search service on demand on the user's behalf and present the precise search result along with the result from EI-Search. EI Web application also publishes pre-computed cluster analysis for all these compounds generated by EI-Clustering.

7.5.2 Implementation

EI Web application uses the High-Fidelity Embedding Toolkit in Section 7.4 to generate high-quality embedding for all 19 million compounds in PubChem Compound collection using PubChem fingerprint Tanimoto similarity measure. It then applies EI-Cluster to pre-compute cluster analysis and uses EI-Search to answer similarity search queries. The Web UI is built with Django web framework (Adrian Holovaty, Simon Willison, et al, 2003). To evaluate the search results generated by EI-Search on demand, Power User Gateway (PUG) service of PubChem is employed to perform similarity search.

Due to the property of our embedding algorithm to independently process individual compound and the high efficiency of EI-Clustering, EI Web application can synchronize with PubChem's update, and therefore provide always up-to-date search and clustering results of PubChem Compound collection. This reinforce the value of EI Web application as an efficient alternative gateway to PubChem Compound database.

7.6 ChemMine Platform

The ChemMine platform is a web application that intends to build a foundation for sharing and collaboration of cheminformatics tools. It aims at bridging the gap between methodology researchers and end users by providing a publishing and integration platform of reusable functional modules. One of the major hurdles of wider and faster adoption of latest research in cheminformatic tools and methods is the weak connection between publishers and the consumer. There is no easy way for method researchers to publish new tools and methods in a form that end users can easily integrate into their existing workflow either for validation or evaluation purpose or for permanent deployment. ChemMine platform, similar to commercial products like Pipeline Pilot, is a repository of tools and functional modules published and maintained by method developers. Users can build custom workflows by chaining modules, and experiment with new tools by replacing a module in an existing workflow. As an open platform, it is an ideal solution for a method developer to provide *live* demonstration of a new method, using data provided by the method developer or, much more interestingly, using users' own existing data and workflows. Comparative study can be achieved simply by replacing the new method with an alternative old method and observing changes in the outcome. As a web application, it provides the maximum flexibility and inherits many other benefits of web applications, such as platform agnostics, zero install, and automatic update.

A component built for ChemMine platform, or a *ChemMine Tool*, is a simple Web application. From the perspective of the ChemMine platform, a component has only the sole responsibility to convert input in some format to output in some format. For example, a component to perform cluster analysis of compound data takes compound data as input and output a hierarchical tree to represent the clustering result. The external interface of the component is also simple. It only needs to answer one pre-defined query to allow ChemMine to remotely start an instance of the component. Upon the creation of the instance, it downloads input from the ChemMine platform, perform the actual data processing, and upload the output to ChemMine platform.

The simple design allows us to provide a framework for developers to build custom component in various programming languages. For example, a tool is provided to simplify the task of wrapping an existing command line program that confines to the program model of transferring input to output. This tool enables the developer to continue creating and releasing applications in traditional forms and at the same time quickly publish it on ChemMine platform with little effort. Allowing developers to host their own component yields high level of flexibility and freedom in development and also allows us to keep ChemMine platform simple, robust and flexible.

The design of ChemMine platform is influenced by the UNIX philosophy. Component, as UNIX programs, are small, simple and dedicated function modules that do one thing and do it well, and the platform provides a mechanism to allow these component to work together. ChemMine platform allows user to build linear workflow by chaining components. This allows the developer to focus on core method development and frees them from other concerns such as file format conversion or visualization. This also gives a user the flexibility in building complicated data processing workflow using his preferred components, and simplifies the task of comparative study of components of the same task. For example, using the same workflow that performs bioactivity prediction, a user can easily experiment a new similarity measure by replacing the existing similarity measure component with a new one and compare the prediction results.

Chapter 8

Conclusions

This dissertation addressed several problems in *in silico* ligand-based virtual screening. Specifically, it makes the following contributions.

- This dissertation investigates Maximum Common Substructure (MCS) as a promising alternative metric for measuring chemical similarity, used either as a standalone method or in combination of other similarity measures, in both similarity search and bioactivity prediction applications.

- This dissertation demonstrates EI-Search and EI-Clustering as an effective way to solve the scalability issue of compound similarity measures in similarity-based applications involving large chemical databases.

- This dissertation results in a toolkit to perform effectiveness and efficiency study of chemical similarity measures.

8.1 Maximum Common Substructure

This dissertation describes the development of a novel MCS algorithm and the concept of basis compounds for measuring the similarity between chemical structures. Both methods are applied to similarity searching of chemical databases and are incorporated into SVM models to efficiently predict biologically active compounds.

More specifically, a new backtracking algorithm for finding the MCS between a pair of graphs is designed and its performance is tested. This algorithm can be applied to chemical structure graphs directly (instead of association graphs), and hence supports several matching constraints as well as relaxations that can be useful when applied to comparison of chemical compounds. Several strategies are introduced to increase the efficiency and flexibility of the algorithm. This algorithm can be effectively used with a progressive optimization strategy, which is very beneficial for ranking a large number of chemical structures by their similarities to query structures.

In order to combine the MCS-based similarity measure and other similarity measures with modern machine learning techniques to form effective bioactivity prediction models, the concept of basis compounds is proposed. This concept allows for a vector representation of chemical structures, similar to traditional fingerprint approaches, while avoiding many drawbacks of the traditional fingerprint approaches. Finally, the derived vectors are incorporated into SVMs to build prediction models of biological activities of compounds.

Our experimental results show that the MCS-based similarity measure is more effective in searching chemical databases than the well-known AP-based method. They also show that the SVM models based on vectors derived from basis compounds are effective for identifying bioactive compounds. Moreover, the MCS-based similarity measure complements the AP-based measure effectively. Our proposed hybrid model, which combines the MCS and the AP similarity measures, provides the most effective predictions in comparison with the existing approaches.

8.2 EI-Search and EI-Clustering

In this dissertation, we have presented EI-Search and EI-Clustering as efficient methods for accelerating structure similarity searches and clustering of very large compound data sets. The acceleration is achieved by applying *embedding and indexing* techniques to represent chemical compounds in a high-dimensional Euclidean space and to employ ultra-fast pre-screening of the compound data set using LSH-assisted nearest neighbor search in the embedding space. Our tests show that the method can dramatically reduce the search time of large databases, by a factor of 40-200 fold when searching the 100 closest compounds to a query. Recently published acceleration methods achieved only a 5.5-fold reduction in search time when using a Tanimoto threshold of 0.8 and up to 20-fold with a relatively restrictive threshold of 0.9 (Swamidass and Baldi, 2007; Baldi et al., 2008). Another limitation of these methods is their narrow utility spectrum that is currently restricted to fingerprint-based searches. In contrast to this, the EI-Search framework is designed to be useful for a wide spectrum of similarity measures. After embedding, EI-Search will run in most cases with comparable time efficiencies independent of the complexity of the similarity measure. This can be particularly useful for accelerating searches that use much more accurate, but computationally very expensive similarity measures, such as maximum common substructures or 3D approaches (Raymond et al., 2003; Willett, 2005; Cao et al., 2008).

By taking advantage of the fast similarity search speed of EI-Search, we developed EI-Clustering into an effective clustering method for very large data sets. The method accelerated the clustering of the three test data sets used in this study by 20 to 80 folds. Most importantly, the EI-Clustering made it feasible to cluster data sets of almost 20 million entries within acceptable time limits. Due to its subquadratic running time, the EI-Clustering method should scale well enough to cluster even larger data sets with tens or even hundreds of millions of objects.

8.3 Chemical Similarity Toolkit

This dissertation resulted in several highly-reusable and user- and developer-friendly software packages and applications. ChemmineR provides support for 2D chemical structure similarity comparison, search, and clustering capability to the popular R programming environment. `libap` exposes the similar functionality with more efficient implementation to C++, Python and R developer. `libmcs` wraps our MCS solver in a reusable software package accessible using C++, Python and R programming languages. High-Fidelity Embedding

Toolkit provides efficient and easily customizable tools to embed compound or other data into high-dimensional Euclidean space with high quality preservation of inter-object dissimilarities. Built on EI-Search and EI-Cluster, the EI web application serves as a PubChem gateway with a similarity search functionality with subsecond response time and pre-computed clustering result synchronized with PubChem Compound collection. Finally, the ChemMine platform provides an ideal foundation for sharing and collaboration of cheminformatics tools, and facilitates quick publishing, integration and application of new cheminformatics methods.

Bibliography

Abt, M., Lim, Y., Sacks, J., Xie, M., Young, S., 2001. Sequential Approach for Identifying Lead Compounds in Large Chemical Databases. Statist. Sci 16 (2), 154–168.

Adrian Holovaty, Simon Willison, et al, 2003. Django web framework.
URL `http://www.djangoproject.com`

Agrafiotis, D., 2003. Stochastic proximity embedding. Journal of computational chemistry 24 (10), 1215–1221.

Agrafiotis, D., Lobanov, V., 1999. An efficient implementation of distance-based diversity measures based on kd trees. J Chem Inf Comp Sci 39, 51–58.

Agrafiotis, D., Rassokhin, D., Lobanov, V., 2001. Multidimensional scaling and visualization of large molecular similarity tables. Journal of Computational Chemistry 22 (5), 488–500.

Agrafiotis, D. K., Xu, H., 2002. A self-organizing principle for learning nonlinear manifolds. Proc Natl Acad Sci U S A 99 (25), 15869–15872.

Altman, D., Bland, J., 1994. Diagnostic tests 2: predictive values. BMJ. British medical journal(International ed.) 309 (6947).

Austin, C. P., Brady, L. S., Insel, T. R., Collins, F. S., Nov 2004. NIH molecular libraries initiative. Science 306 (5699), 1138–1139.

Baldi, P., Hirschberg, D., Nasr, R., 2008. Speeding up chemical database searches using a proximity filter based on the logical exclusive OR. J Chem Inf Model 48 (7), 1367–1378.

Barrow, H., Burstall, R., 1976. Subgraph Isomorphism, Matching Relational Structures and Maximal Cliques. Information Processing Letters 4 (4), 83–84.

Beazley, D., 1996. SWIG: An easy to use tool for integrating scripting languages with C and C++. In: Proceedings of the 4th conference on USENIX Tcl/Tk Workshop, 1996-Volume 4. USENIX Association, p. 15.

Bentley, J., 1975. Multidimensional binary search trees used for associative searching. Comm. ACM 18 (9), 509–517.

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.

Bohm, C., Berchtold, S., Keim, D., 2001. Searching in high-dimensional spaces: index structures for improving the performance of multimedia databases. ACM Comput. Surv. 33 (3), 322–373.

Boser, B., Guyon, I., Vapnik, V., 1992. A training algorithm for optimal margin classifiers. Proceedings of the fifth annual workshop on Computational learning theory, 144–152.

Brinkhoff, T., Kriegel, H., Seeger, B., 1993. Efficient processing of spatial joins using R-trees. In: Proc. ACM SIGMOD Conf. on Management of Data. ACM New York, NY, USA, pp. 237–246.

Brown, R., Martin, Y., 1996. Use of Structure- Activity Data To Compare Structure-Based Clustering Methods and Descriptors for Use in Compound Selection. J. Chem. Inf. Comput. Sci 36 (3), 572–584.

Bunke, H., 1997. On a relation between graph edit distance and maximum common subgraph. Pattern Recognition Letters 18 (8), 689–694.

Bunke, H., 2000. Graph matching: Theoretical foundations, algorithms, and applications. Proc. Vision Interface, 82–88.

Bunke, H., Shearer, K., 1998. A graph distance metric based on the maximal common subgraph. Pattern Recognition Letters 19 (3-4), 255–259.

Burbidge, R., Trotter, M., Buxton, B., Holden, S., 2001. Drug design by machine learning: support vector machines for pharmaceutical data analysis. Computers and Chemistry 26 (1), 5–14.

Cannon, E., Amini, A., Bender, A., Sternberg, M., Muggleton, S., Glen, R., Mitchell, J., 2007. Support vector inductive logic programming outperforms the naive Bayes classifier and inductive logic programming for the classification of bioactive chemical compounds. Journal of Computer-Aided Molecular Design 21 (5), 269–280.

Cao, Y., Jiang, T., Girke, T., 2008. A maximum common substructure-based algorithm for searching and predicting drug-like compounds. Bioinformatics 24 (13), i366.

Carhart, R., Smith, D., Venkataraghavan, R., 1985. Atom pairs as molecular features in structure-activity studies: definition and applications. Journal of Chemical Information and Computer Sciences 25 (2), 64–73.

Chang, C., Lee, R., 1973. A heuristic relaxation method for nonlinear mapping in cluster analysis. IEEE Transactions on Systems, Man, and Cybernetics 3, 197200.

Chang, C., Lin, C., 2001. LIBSVM: a library for support vector machines. Software available at http://www. csie. ntu. edu. tw/cjlin/libsvm 80, 604–611.

Chen, J. H., Linstead, E., Swamidass, S. J., Wang, D., Baldi, P., Sep 2007. ChemDB update–full-text search and virtual chemical space. Bioinformatics 23 (17), 2348–2351.

Chen, X., Reynolds, C., 2002. Performance of Similarity Measures in 2D Fragment-Based Similarity Searching: Comparison of Structural Descriptors and Similarity Coefficients. Journal of Chemical Information and Computer Sciences 42 (6), 1407–1414.

Cone, M., Venkataraghavan, R., McLafferty, F., 1977. Molecular Structure Comparison Program for the Identification of Maximal Common Substructures. J. Am. Chem. SOC 99, 7668–7671.

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 (3), 265–298.

Cordella, L., Foggia, P., Sansone, C., Tortorella, F., Vento, M., 1998. Graph Matching: a Fast Algorithm and its Evaluation. In: Proc. 14th Int. Conf. Pattern Recognition. pp. 1582–1584.

Cordella, L., Foggia, P., Sansone, C., Vento, M., 2001. An improved algorithm for matching large graphs. Proc. of the 3rd IAPR TC-15 Workshop on Graphbased Representations in Pattern Recognition, 149–159.

Datar, M., Immorlica, N., Indyk, P., Mirrokni, V., 2004. Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the twentieth annual symposium on Computational geometry. ACM New York, NY, USA, pp. 253–262.

Dean, P., 1995. Molecular Similarity in Drug Design. Blackie Academic & Professional.

DeLano, W., 2005. The case for open-source software in drug discovery. Drug discovery today 10 (3), 213–217.

Deshpande, M., Kuramochi, M., Karypis, G., SCIENCE, M. U. M. D. O. C., 2005. Frequent Sub-Structure-Based Approaches for Classifying Chemical Compounds. IEEE Transactions on Knowledge and Data Engineering 17 (8), 1036–1050.

Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., Weingessel, A., 2005. e1071: Misc Functions of the Department of Statistics (e1071), TU Wien.

Dobson, C., 2004a. Chemical space and biology. Nature 432 (7019), 824–828.

Dobson, C. M., Dec 2004b. Chemical space and biology. Nature 432 (7019), 824–828.

Downs, G., Barnard, J., 2002. Clustering methods and their uses in computational chemistry. Rev. Comput. Chem. 18, 1–40.

Dumay, A., van der Geest, R., Gerbrands, J., Jansen, E., Reiber, J., 1992. Consistent inexact graph matching applied to labelling coronarysegments in arteriograms. Pattern Recognition, 1992. Vol. III. Conference C: Image, Speech and Signal Analysis, Proceedings., 11th IAPR International Conference on, 439–442.

Dutta, D., Guha, R., Jurs, P., Chen, T., 2006. Scalable partitioning and exploration of chemical spaces using geometric hashing. J. Chem. Inf. Model 46 (1), 321–333.

Engels, M., Venkatarangan, P., 2001. Smart screening: approaches to efficient HTS. Curr Opin Drug Discov Devel 4 (3), 275–83.

Faloutsos, C., Lin, K., 1995. FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In: Proc. ACM SIGMOD Conf. on Management of Data. ACM New York, NY, USA, pp. 163–174.

Flower, D., 1998. On the properties of bit string-based measures of chemical similarity. Journal of Chemical Information and Computer Sciences 38 (3), 379–386.

Fu, A., Chan, P., Cheung, Y., Moon, Y., 2000. Dynamic vp-tree indexing for n-nearest neighbor search given pair-wise distances. VLDB J. 9 (2), 154–173.

Garey, M., Johnson, D., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. WH Freeman & Co. New York, NY, USA.

Gedeck, P., Rohde, B., Bartels, C., Sep-Oct 2006. QSAR–how good is it in practice? Comparison of descriptor sets on an unbiased cross section of corporate data sets . J Chem Inf Model 46 (5), 1924–1936.

Gentleman, R., Carey, V., Bates, D., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., et al., 2004. Bioconductor: open software development for computational biology and bioinformatics. Genome biology 5 (10), R80.

Gentleman, R., Carey, V., Dudoit, S., Irizarry, R., Huber, W., 2005. Bioinformatics and Computational Biology Solutions Using R and Bioconductor. Springer, New York.

Gionis, A., Indyk, P., Motwani, R., 1999. Similarity search in high dimensions via hashing. In: Proc. VLDB. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, pp. 518–529.

Girke, T., Cheng, L. C., Raikhel, N., Jun 2005a. ChemMine. A compound mining database for chemical genomics. Plant Physiol 138 (2), 573–577.

Girke, T., Cheng, L. C., Raikhel, N., 2005b. ChemMine. A compound mining database for chemical genomics. Plant Physiol 138 (2), 573–577.

Gonzalez, J., Holder, L., Cook, D., 2001. Application of Graph-Based Concept Learning to the Predictive Toxicology Domain. Proceedings of the Predictive Toxicology Challenge Workshop.

Guha, R., Howard, M. T., Hutchison, G. R., Murray-Rust, P., Rzepa, H., Steinbeck, C., Wegner, J., Willighagen, E. L., 2006. The Blue Obelisk-interoperability in chemical informatics. J Chem Inf Model 46 (3), 991–998.

Hagadone, T., 1992. Molecular substructure similarity searching: efficient retrieval in two-dimensional structure databases. Journal of Chemical Information and Computer Sciences 32 (5), 515–521.

Haggarty, S. J., Jun 2005. The principle of complementarity: chemical versus biological space. Curr Opin Chem Biol 9 (3), 296–303.

Haranczyk, M., Holliday, J., 2008. Comparison of similarity coefficients for clustering and compound selection. Journal of chemical information and modeling 48 (3), 498–508.

Holder, L., Cook, D., Djoko, S., 1994. Substructure discovery in the subdue system. Proc. AAAI 94, 169–180.

Holliday, J. D., Salim, N., Whittle, M., Willett, P., May-Jun 2003. Analysis and display of the size dependence of chemical similarity coefficients. J Chem Inf Comput Sci 43 (3), 819–828.

Ihlenfeldt, W. D., Voigt, J. H., Bienfait, B., Oellien, F., Nicklaus, M. C., Jan-Feb 2002.

Enhanced CACTVS browser of the open NCI database. J Chem Inf Comput Sci 42 (1), 46–57.

Irwin, J. J., Shoichet, B. K., Jan-Feb 2005. ZINC–a free database of commercially available compounds for virtual screening. J Chem Inf Model 45 (1), 177–182.

James, C., Weininger, D., Delany, J., 1992. Daylight theory manual. Mission Viejo, CA: Daylight Chemical Information Systems Inc.

Johnson, M., Maggiora, G., 1990a. Concepts and applications of molecular similarity. Wiley New York.

Johnson, M., Maggiora, G., 1990b. Concepts and Applications of Molecular Similarity. Wiley.

Katayama, N., 1997. The SR-tree: an index structure for high-dimensional nearest neighbor queries. In: Proc. ACM SIGMOD Conf. on Management of Data. ACM New York, NY, USA, pp. 369–380.

Kearsley, S., Sallamack, S., Fluder, E., Andose, J., Mosley, R., Sheridan, R., 1996. Chemical Similarity Using Physiochemical Property Descriptors. J. Chem. Inf. Comput. Sci 36 (1), 118–127.

Koch, I., 2001. Enumerating all connected maximal common subgraphs in two graphs. Theoretical Computer Science 250 (1), 1–30.

Kruskal, J., Wish, M., 1978. Multidimensional Scaling. Sage Publications.

Lang, D. T., Swayne, D., Wickham, H., Lawrence, M., 2007. rggobi: Interface between R and GGobi. R package version 2.1.7.

Levi, G., 1973. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. Calcolo 9 (4), 341–352.

Luo, B., Hancock, E., 2001. Structural graph matching using the EM algorithm and singular value decomposition. IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (10), 1120–1136.

Lv, Q., Josephson, W., Wang, Z., Charikar, M., Li, K., 2007. Multi-probe LSH: efficient indexing for high-dimensional similarity search. In: Proc. VLDB. VLDB Endowment, pp. 950–961.

McGregor, J., 1982. Backtrack Search Algorithms and the Maximal Common Subgraph Problem. SOFTWARE-PRACT. AND EXPER. 12 (1), 23–34.

Meglen, R., 1992. Examining large databases: a chemometric approach using principal component analysis. Marine Chemistry 39 (1-3), 217–237.

National Center for Biotechnology Information, 2009. Pubchem substructure fingerprint. URL `ftp://ftp.ncbi.nlm.nih.gov/pubchem/specifications/pubc% hem_fingerprints.txt`

Neuhaus, M., Bunke, H., 2006. Edit distance-based kernel functions for structural pattern classification. Pattern Recognition 39 (10), 1852–1863.

NIH Chemical Genomics Center, 2009. PubChem Fingerprint for JChem. URL `http://ncgc.nih.gov/resources/software.html`

Nikolova, N., Jaworska, J., 2004. Approaches to measure chemical similarity-a review. QSAR & Combinatorial Science 22 (9-10), 1006–1026.

O'Boyle, N. M., Morley, C., Hutchison, G. R., Mar 2008. Pybel: a Python wrapper for the OpenBabel cheminformatics toolkit. Chem Cent J 2 (1), 5–5.

Olson, M., Bostic, K., Seltzer, M., 1999. Berkeley db. In: Proceedings of the FREENIX Track: 1999 USENIX Annual Technical Conference. pp. 183–192.

Oprea, T. I., Jun 2002. Chemical space navigation in lead discovery. Curr Opin Chem Biol 6 (3), 384–389.

Oprea, T. I., Tropsha, A., Faulon, J. L., Rintoul, M. D., Aug 2007. Systems chemical biology. Nat Chem Biol 3 (8), 447–450.

Provost, F., Fawcett, T., 2001. Robust Classification for Imprecise Environments. Machine Learning 42 (3), 203–231.

R Development Core Team, 2008. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, ISBN 3-900051-07-0.

Raymond, J., Gardiner, E., Willett, P., 2002a. Heuristics for similarity searching of chemical graphs using a maximum common edge subgraph algorithm. J Chem Inf Comput Sci 42 (2), 305–16.

Raymond, J., Gardiner, E., Willett, P., 2002b. RASCAL: Calculation of Graph Similarity using Maximum Common Edge Subgraphs. The Computer Journal 45 (6), 631.

Raymond, J. W., Blankley, C. J., Willett, P., Mar 2003. Comparison of chemical clustering methods using graph- and fingerprint-based similarity measures. J Mol Graph Model 21 (5), 421–433.

Roweis, S., Saul, L., 2000. Nonlinear dimensionality reduction by locally linear embedding. Science 290 (5500), 2323.

Savchuk, N. P., Balakin, K. V., Tkachenko, S. E., Aug 2004. Exploring the chemogenomic knowledge space with annotated chemical libraries. Curr Opin Chem Biol 8 (4), 412–417.

Schneider, G., 2000. Neural networks are useful tools for drug design. Neural Networks 13 (1), 15–16.

Schwaighofer, A., Schroeter, T., Mika, S., Laub, J., Ter Laak, A., Sulzle, D., Ganzer, U., Heinrich, N., Muller, K., 2007. Accurate solubility prediction with error bars for electrolytes: A machine learning approach. J. Chem. Inf. Model 47 (2), 407–424.

Seiler, K. P., George, G. A., Happ, M. P., Bodycombe, N. E., Carrinski, H. A., Norton, S., Brudz, S., Sullivan, J. P., Muhlich, J., Serrano, M., Ferraiolo, P., Tolliday, N. J., Schreiber, S. L., Clemons, P. A., Jan 2008. ChemBank: a small-molecule screening and cheminformatics resource database. Nucleic Acids Res 36 (Database issue), 351–359.

Shakhnarovich, G., Darrell, T., Indyk, P., 2005. Nearest-neighbor Methods in Learning and Vision: Theory and Practice. MIT Press.

Sheridan, R., Miller, M., Underwood, D., Kearsley, S., 1996. Chemical Similarity Using Geometric Atom Pair Descriptors. J. Chem. Inf. Comput. Sci 36 (1), 128–136.

Sheridan, R. P., Kearsley, S. K., 2002. Why do we need so many chemical similarity search methods? Drug Discov Today 7 (17), 903–911.

Shi, L., Fan, Y., Lee, J., Waltham, M., Andrews, D., Scherf, U., Paull, K., Weinstein, J., 2000. Mining and Visualizing Large Anticancer Drug Discovery Databases. J. Chem. Inf. Comput. Sci 40 (2), 367–379.

Smellie, A., Wilson, C., Ng, S., 2006. Visualization and interpretation of high content screening data. J. Chem. Inf. Model 46 (1), 201–207.

Strausberg, R. L., Schreiber, S. L., Apr 2003. From knowing to controlling: a path from genomics to drugs using small molecule probes. Science 300 (5617), 294–295.

Swamidass, S., Baldi, P., 2007. Bounds and algorithms for fast exact searches of chemical fingerprints in linear and sub-linear time. J Chem Inf Model 47 (2), 302.

Tenenbaum, J., Silva, V., Langford, J., 2000. A global geometric framework for nonlinear dimensionality reduction. Science 290 (5500), 2319.

Tsai, W., Fu, K., 1979. Error-correcting isomorphisms of attributed relational graphs for pattern analysis. IEEE Transactions on Systems, Man, and Cybernetics 9, 757–768.

Vaidya, P., 1989. An O (n logn) algorithm for the all-nearest-neighbors problem. Discrete and Computational Geometry 4 (1), 101–115.

Wang, Y., Fan, K., Horng, J., 1997. Genetic-based search for error-correcting graph isomorphism. Systems, Man and Cybernetics, Part B, IEEE Transactions on 27 (4), 588–597.

Weber, R., Schek, H., Blott, S., 1998. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: Proc. VLDB. IEEE, pp. 194–205.

Wheeler, D., Barrett, T., Benson, D., Bryant, S., Canese, K., Chetvernin, V., Church, D., DiCuccio, M., Edgar, R., Federhen, S., et al., 2007. Database resources of the National Center for Biotechnology Information. Nucleic Acids Research 35 (Database issue), D5.

Willett, J., 1987. Similarity and Clustering in Chemical Information Systems. John Wiley & Sons, Inc., New York, NY, USA.

Willett, P., Jun 2005. Searching techniques for databases of two- and three-dimensional chemical structures. J Med Chem 48 (13), 4183–4199.

Willett, P., Barnard, J., Downs, G., 1998. Chemical Similarity Searching. Journal of Chemical Information and Computer Sciences 38 (6), 983–996.

Wilson, R., Hancock, E., 1997. Structural matching by discrete relaxation. IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (6), 634–648.

Xu, H., Agrafiotis, D., 2003. Nearest neighbor search in general metric spaces using a tree data structure with a simple heuristic. J Chem Inf Comp Sci 43 (6), 1933–1941.

Yan, X., Yu, P. S., Han, J., 2005. Substructure similarity search in graph databases. In: SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. ACM Press, New York, NY, USA, pp. 766–777.

Young, S., 1999. Analysis of a Large Structure/Biological Activity Data Set Using Recursive Partitioning. J. Chem. Inf. Comput. Sci 39, 1017–1026.

Zhu, C., Byrd, R., Lu, P., Nocedal, J., 1997. L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. ACM Trans. Math. Software 23 (4), 550–560.