

# **A Robust Algorithm to Compare Chemical Structures**

by

©Jiayi Zhou

A Thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of

**Master of Science**

**Computer Science Department  
Memorial University of Newfoundland  
Faculty Science**

Memorial University of Newfoundland

**January, 2015**

St. John's

Newfoundland

# Abstract

The recognition of chemical similarities between molecules plays an important role in chemical science, especially in the subjects of chemistry, biology and pharmaceuticals. Traditional methods of structure recognition are time consuming, usually involving a lot of experimentation and computational effort. The goal of the research is to create an algorithm to compare two chemical structures automatically with basic information. The algorithm only requires atom Cartesian coordinates, atom types and connectivity information of the structures as input. It uses a novel method to pair the atoms in the two structures such that the best superimposition is achieved. A similarity score is computed based on this best superimposition.

The algorithm can also be used to search a large set of molecules for a structure similar to a query molecule. An application is developed to display the two structures to be compared and provide a 3D image of their best superimposition based on the auto-pairing of the atoms. Run-time analysis of the algorithm reveals that the traditional time complexity does not describe the run-time of the algorithm well. Linear regression indicates that the run-time is strongly influenced by the number of triplets (consisting of 3 atoms joined by 2 bonds) matched between the two structures. Testing of the algorithm on an in-house data-set of 737 structures as well as a larger NCI-sourced database demonstrates its utility.

# Acknowledgements

First and foremost I would like to thank my supervisors Dr. Sharene Bungay and Dr. Edward Brown for taking me under their wings and giving me their unwavering support, patient help, sagacious advice and financial support. I really appreciate Dr. Proton Rahman to offer me the part time job at PTRG research group when the funding of my research was stopped. I extend my gratitude to all the colleagues at PTRG research group for their caring and cooperation with my part time job, especially Dr. Lourdes Pena-Castillo, I am so grateful for her supervision of my part time job. I must give a nod to Memorial University, its School of Graduate Studies and its Faculty of Science for their tireless commitment to the graduate students of this institution. Last but not least, no acknowledgement would be complete without some praise for Memorial's Department of Computer Science and every faculty, student and staff member in it for handling my pedantic concerns and network crises with the utmost professionalism; and for accepting me to come and be a part of them as a graduate student in the first place. To everyone I have had the pleasure of meeting and getting to know over the past three years as a result of this undertaking, finger crossed for everyone.

# Table of Contents

Abstract	ii
Acknowledgments	iii
Table of Contents	vi
List of Tables	vii
List of Figures	ix
<b>1 Introduction</b>	<b>1</b>
<b>2 Literature Review</b>	<b>5</b>
2.1 Non-Superimposing methods . . . . .	5
2.2 Superimposing methods . . . . .	10
<b>3 Objectives and Approach</b>	<b>14</b>
3.1 Objectives . . . . .	14
3.2 Approach . . . . .	15
3.2.1 Definition of similarity . . . . .	18
3.2.2 Comparison algorithm . . . . .	20
3.2.2.1 Finding the correct mapping of the atoms . . . . .	21

3.2.2.2	Finding the best superimposition . . . . .	35
3.2.3	Summary . . . . .	40
<b>4</b>	<b>Qualitative performance of the algorithm</b>	<b>41</b>
4.1	Objectives . . . . .	41
4.2	Application . . . . .	42
4.3	Data sets . . . . .	43
4.3.1	Test results . . . . .	44
4.3.1.1	Finding similar molecules . . . . .	44
4.3.1.2	Changes in orientation and labeling order . . . . .	47
4.3.1.3	Searching the sub-NCI database . . . . .	47
4.3.1.4	Test conclusion . . . . .	49
4.3.2	Some Features of the algorithm . . . . .	50
<b>5</b>	<b>Run-time analysis</b>	<b>56</b>
5.1	Time complexity analysis of the algorithm . . . . .	56
5.1.1	Time complexity . . . . .	56
5.1.2	Time complexity calculation . . . . .	57
5.2	Obtaining the run-time data . . . . .	66
5.3	Run-time analysis based on the empirical run-time . . . . .	68
5.4	Some examples of the empirical run-time of the algorithm . . . . .	72
5.5	Conclusion . . . . .	76
<b>6</b>	<b>Conclusion and future work</b>	<b>77</b>
<b>A</b>	<b>Detailed results of Table 4.1</b>	<b>80</b>
<b>B</b>	<b>Detailed results of using molecules in Table 4.1 to search the 737r-</b>	
	<b>data-set</b>	<b>83</b>

C Detailed results of Table 4.2	86
D An example of the input structure file used by the algorithm	90
E Detailed test results of 100 randomly selected pairs in Figure 5.2	92
Bibliography	96

# List of Tables

4.1	Top six similar molecules for five sample query structures searching the 737-data-set. . . . .	46
4.2	Top five similar molecules for five sample query structures searching the sub-NCI data set. . . . .	49
5.1	The linear regression results generated by R. . . . .	71
5.2	25 randomly selected structures to search the sub-NCI database. . . .	75

# List of Figures

1.1	3D geometric structure of 5-methyl-2-pyridinamine ( $C_6H_8N_2$ ). . . . .	2
1.2	Structure of hydroxybenzene(left) and ortho-hydroxybenzoic(right). .	3
3.1	Flow chart illustrating the algorithm to compute the similarity between two molecules. . . . .	16
3.2	A pair of two-dimensional “molecules” of five “atoms” at two different orientations. The green circles represent the location of the atoms in molecule A and the red circles represent the location of the atoms in molecule B. (redrawn from [12]) . . . . .	28
3.3	Two molecules to be compared, 2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ ) (left) and 2-hydroxy-5-methylbenzo-1,4-quinone ( $C_7H_6O_3$ ) (right). For illustration, the atoms within each structure are arbitrarily numbered.	32
4.1	The distribution of size (number of atoms) of the test molecules in the sub-NCI database. . . . .	44
4.2	Two similar molecules in the NCI database, the left one is 2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ ) and the right one is 2-hydroxy-5-methylbenzo-1,4-quinone ( $C_7H_6O_3$ ). . . . .	50



4.3	The labeling order in the structural files of 2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ ) (left) and 2-hydroxy-5-methylbenzo-1,4-quinone ( $C_7H_6O_3$ ) (right) in the NCI database. . . . .	51
4.4	Comparison result in Jmol based on the structural files of 2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ ) and 2-hydroxy-5-methylbenzo-1,4-quinone ( $C_7H_6O_3$ ) (in transparent yellow) in the NCI database. . . . .	52
4.5	Comparison result of the algorithm based on the structural files of 2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ ) and 2-hydroxy-5-methylbenzo-1,4-quinone ( $C_7H_6O_3$ ) (in transparent yellow) in the NCI database. . . .	53
4.6	Molecule $C_7H_{10}$ and its isomer. . . . .	54
4.7	Comparison result of $C_7H_{10}$ and one of its isomers (in yellow). . . . .	55
5.1	Number of atoms vs. the number of triplets found in the structure for all the structures in the sub-NCI database. . . . .	61
5.2	The run-time of 100 randomly selected pairs of comparing structures in the sub-NCI database. . . . .	67
5.3	Number of atoms in the first molecule vs. the run-time . . . . .	69
5.4	Number of atoms in the second molecule vs. the run-time . . . . .	69
5.5	The number of mapped triplets vs. the run-time . . . . .	70
5.6	The residual plot of the linear regression generated by R. . . . .	71
5.7	The run-time distribution of 3900 randomly selected pairs of structures in the sub-NCI database. . . . .	73
5.8	Chemical structure $C_{70}H_{138}N_8O_4S_4$ . . . . .	74

# Chapter 1

## Introduction

Chemistry is a science that studies the composition, structure, properties and variations of substances on an atomic level. An atom is the basic building block of chemical structures. The arrangement of a substance's atoms reveals its properties. In earlier ages, people could study the properties of a substance only by chemical experiments. All that was known was when mixing a substance with a particular substance, some reactions may happen (color change, precipitations, etc.). Thus, the properties of a substance were studied by the reactions observed in different chemical experiments. With the development of quantum chemistry, the study of chemistry reached the electronic level. A substance is composed of molecules and a molecule is composed of atoms. The arrangement of atoms in a molecule can be determined, which provides a more intuitive way to study the properties of a substance. By using spectroscopic methods, a 3D geometric structure of a molecule can be obtained. A molecule can be represented by spheres and cylinders in three dimensional space. Different atoms are represented by different size spheres. The bonds between the atoms are represented by different length cylinders. Figure 1.1 shows an image of the 3D geometric structure of the molecule 5-methyl-2-pyridinamine ( $C_6H_8N_2$ ). The white spheres represent hy-

drogen atoms, the grey spheres represent carbon atoms and the blue spheres represent nitrogen atoms. The cylinders connecting the spheres represent the bonds between the atoms. It can be seen that the 3D representation of a molecule provides a more clear and intuitive image of a molecule than its molecular formula or name.

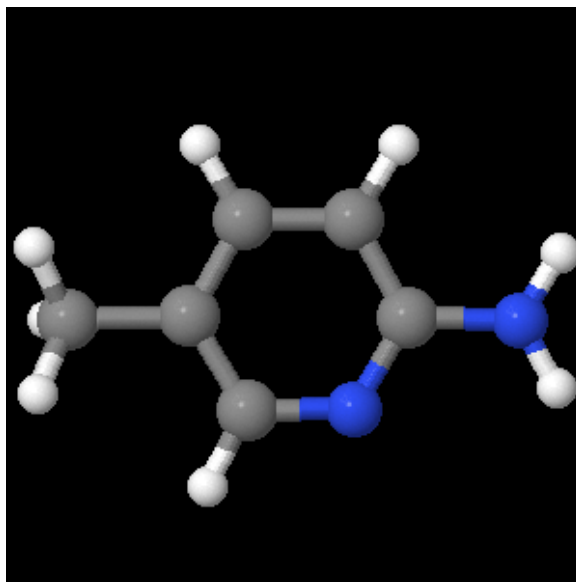
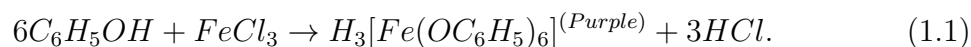


Figure 1.1: 3D geometric structure of 5-methyl-2-pyridinamine ( $C_6H_8N_2$ ).

Examining the structure of a molecule provides a more effective way to study a substance. It avoids the use of time-consuming experiments. Chemists usually try to study an unknown substance's properties by chemical experiments. Assuming that chemists would like to study the properties of ortho-hydroxybenzoic ( $C_6H_4(OH)(COOH)$ ), they could mix it with other chemical substances and observe the chemical reactions that occur. For example, if ferric trichloride ( $FeCl_3$ ) liquid is added to ortho-hydroxybenzoic liquid, the mixed liquid will become purple, as in reaction (1.1):



As a result, chemists know that ortho-hydroxybenzoic contains a substructure hy-

droxybenzene ( $C_6H_5OH$ ), since the hydroxybenzene ion will form purple complexes with  $Fe^{3+}$ , as in reaction (1.1). One can then assume that ortho-hydroxybenzoic may have the same properties as hydroxybenzene and further experiments can be performed to prove it. In practice, chemists may have tried hundreds of liquids other than ferric trichloride because they did not know the composition of ortho-hydroxybenzoic. Even observing purple complexes upon adding ferric trichloride liquid to ortho-hydroxybenzoic liquid, does not confirm that ortho-hydroxybenzoic contains hydroxybenzene since ions other than hydroxybenzene may also form purple complexes with  $Fe^{3+}$ . Thus, further experiments are required. It takes quite a long time to perform all the experiments required.

However, by examining the structure of ortho-hydroxybenzoic, it can be seen directly that it contains hydroxybenzene as a substructure. The structure of hydroxybenzene and ortho-hydroxybenzoic is shown in Figure 1.2.

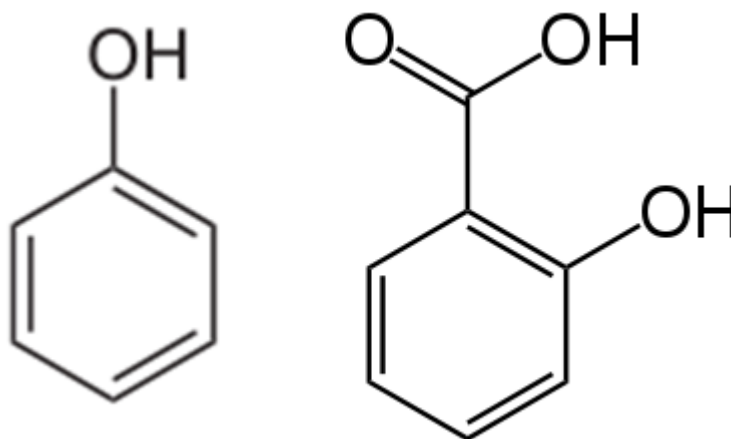


Figure 1.2: Structure of hydroxybenzene(left) and ortho-hydroxybenzoic(right).

As can be seen from the above example, studying the structure from an atomic perspective greatly eases chemical research. However, in real world practice, the substance to be studied is much more complex. It may be impossible for chemists to find similar substructures manually. With the development of computers, chemists

may automate the process of finding similar structures or substructures. Thus, a reliable method of comparing structural differences is needed. Recognizing the similarity between two chemical structures with hundreds or even thousands of atoms is a challenge, and an interesting topic in computational chemistry.

In order to use computers to compare molecules, a quantitative value to describe their similarity is needed. It is a challenge to define a strict and accurate value that can generally describe the similarity between molecules. Consequently, a large number of approaches have been invented over the past several years [1–3,5,6,8–12]. However, these approaches all have their pros and cons in describing the similarity between the molecules (details will be discussed in Chapter 2). It remains a struggle to find a universal measurement to define similarity.

With the increasing use of 3D molecular structure databases in modern chemical research, finding similar molecules from a database containing thousands of molecules becomes more and more crucial. Thus, an algorithm must not only compare the similarity between two molecules, but also search an entire database to find molecules that are similar to a particular query structure. A structure that a user uses as a reference structure to find similar structures is called a “query” structure in this thesis, while the molecule to which it is being compared is called the “target” structure.

The goal of this research is to define a proper measurement to describe the similarity between molecules and create a robust algorithm that is able to compare two molecules automatically, requiring no additional information besides structural files containing only atomic Cartesian coordinates, atom types and connectivity information. This algorithm can also be used to search 3D molecular structure databases for molecules similar to a particular query molecule.

# Chapter 2

## Literature Review

Methods for recognition of chemical similarities between molecules are essentially of two types: superimposing methods and non-superimposing methods. Superimposing methods rely on the best superimposition of the molecules for comparison, while non-superimposing methods use internal quantities of each molecule that are independent of a particular orientation to measure the difference.

### 2.1 Non-Superimposing methods

Non-superimposing methods focus on the comparison of pre-computed descriptors (sometimes computing these descriptors is very expensive) that can represent the essential features of the molecule. This allows for identification of similarities between molecules at a low computation cost based on these descriptors (once they are pre-computed), and is an obvious advantage over superimposing methods that are quite time-consuming.

The basic idea of non-superimposing methods is to obtain some specific values calculated from atom types, distances between atoms, or other information in a molecule, to form one or more histograms of these values. Interatomic distances or surface data

are often used to form these specific values. The shape similarity is then based on the similarity in the statistical properties such as moments or frequencies of these values between two molecules. In [1], the devised method considers a molecular structure that may be abstracted by its all-pairs-shortest-paths matrix and the distance matrix. The shortest distance between any two atoms in a molecule, through the molecule’s bonds, is stored in the all-pairs-shortest-paths matrix and the geometric distance between any two atoms in a molecule is stored in the distance matrix. This method calculates the Euclidean metric between the histograms of these two matrices to obtain a similarity score for the two molecules. Two years later, this method was updated by using atom-based descriptors and surface-based descriptors [2]. Atom-based descriptors are atom triangle bit strings and atom triangle histograms. This method considers all combinations of atom triplets in the molecule. It digitizes atom-atom distances using a 1.0 Angstrom range and sorts the sides of each triplet containing distances smaller than 30 Angstroms by length (long, intermediate, short). The resulting values are used to turn on the variable, which is stored in a logical array named SHAPE (long, intermediate, short). After all triplets have been calculated, the SHAPE array is packed into a bit string, which acts as one of the atom-based descriptors. The atom triangle histogram is constructed from triplets’ perimeters and triplets’ area ratios. A triplet area ratio is calculated as:

$$Ratio = \frac{A}{A_{max}} \exp(-(((P/3 - S_1)^2)^{1/2} + ((P/3 - S_2)^2)^{1/2} + ((P/3 - S_3)^2)^{1/2} / 2P/3)), \quad (2.1)$$

where,

$$A = (P/2(P/2 - S_1)(P/2 - S_2)(P/2 - S_3))^{1/2},$$

$$A_{max} = (P^4/432)^{1/2},$$

P is the perimeter of a triplet and  $S_1$ ,  $S_2$ ,  $S_3$  are the lengths of its three edges. All the triplets' perimeters are binned in 1 Angstrom ranges up to 50 Angstroms in length, and all longer perimeters are ignored. The area ratio is multiplied by 10 and digitized, producing 10 ratio bins. The resulting integer array is SHAPE (ratio, perimeter). Each time triplet data are calculated, the associated array variable is increased by one. In this way, the atom triangle histogram is built up.

Surface based descriptors are centroid triangle perimeter surface histograms, centroid triangle/normal angle surface histograms, multi-surface descriptor bit strings and clustered triangle surface histograms [2]. These descriptors require surface data of a molecule generated by placing uniform points on the molecular or active-site surface. The method then constructs centroid triangles using surface point centroids and all combinations of surface point pairs, and stores the perimeter data in 50 bins of 1 Angstrom to form the centroid triangle perimeter surface histograms. The centroid triangle/normal angle surface histograms are constructed using the centroid triangles with the same method as the atom triangle histogram, but with the perimeters binned in 1.5 Angstrom ranges up to 60 Angstroms. The area ratio is then multiplied by 7 and digitized to produce seven area ratio bins. In addition, the angles between normals are binned in three angle bins ( $60^\circ$ ,  $120^\circ$ ,  $180^\circ$ ) to form a histogram. The multi-surface descriptor bit string is created by packing a five-dimensional array containing information of centroid triangle perimeter, area ratio, normal angle, normal torsion and curvature as a bit string. The five-dimensional array contains 40 perimeter slots of 1.5 Angstroms, five area ratio slots, three  $60^\circ$  normal angle slots, six  $60^\circ$  normal torsion slots and six curvature slots. The clustered triangle surface histograms are built up using the same procedure as the atom triangle histogram but using the triangles determined between clustered points. All surface points associated with a particular atom are clustered. All points within 1.5 Angstrom of each other are assigned to the



same cluster and all clusters with a surface area less than  $0.5 \text{ Angstrom}^2$  are removed. The method then calculates the similarity between the molecules by measuring the difference between the histograms with  $\sum_{i=1}^n B_i^2 / \sum_{i=1}^m \max(T_{A,i}, T_{B,i})^2$ . Here,  $m$  and  $n$  are the total number of bins of a histogram, or the length of a bit string, the  $B$  is the number of descriptors overlapping between equivalent histogram bins, or the number of matching bits for equivalent packed integers, and  $T_A$  and  $T_B$  are the values for the individual descriptors for the same histogram bins/packed integers.

Randy J. Zauhar et al. [3] presents a method using a quite different approach to describe molecules by shape signature. This method compares the difference between a histogram of the information derived from the simulation of a ray-trace reflecting within the molecular volume to measure the similarity between two molecules. It uses the smooth molecular surface triangulator algorithm [4] to generate the solvent-accessible molecular surface. The method then initiates a ray from a random point on the molecular surface in a random direction towards the inside of a molecule. The ray is propagated by the rules of optical reflection and the line segments that connect two successive reflection points are saved. The distribution of the length of these segments is then used as the shape signature of the molecule. The similarity score is calculated as the sum of all the differences in all the bins between the histograms of two molecules.

Among all of these non-superimposing methods, the ultrafast shape recognition (USR) method is perhaps the most popular one [5]. USR uses the first three moments of the distribution of different preconstructed distance sets in a molecule as shape descriptors. First, USR chooses four significant locations in a molecule to construct the distance sets: the molecular centroid (CTD), the closest atom to CTD (CST), the farthest atom from CTD (FCT) and the farthest atom from FCT (FTF). USR then calculates the distances between every atom of the molecule and these four locations,

so that four distances sets,  $\{d_k^{CTD}\}_{k=1}^N$ ,  $\{d_k^{CST}\}_{k=1}^N$ ,  $\{d_k^{FCT}\}_{k=1}^N$  and  $\{d_k^{FTF}\}_{k=1}^N$  ( $N$  is the number of atoms) are obtained. Next, USR calculates the first three moments of the distribution of these four data sets. The moments of the distribution of random variables  $X_1, X_2, \dots, X_n$  can be calculated with the following equations:

$$\mu_1 = \frac{1}{n} \sum_{i=1}^n X_i,$$

$$\mu_r = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})^r \quad (r = 2, 3 \dots n), \quad (2.2)$$

where  $\mu_r$  is the  $r^{th}$  moment and  $\bar{X}$  is the mean of the data set. By using these two equations, the first three moments of the data sets  $\{d_k^{CTD}\}_{k=1}^N$ ,  $\{d_k^{CST}\}_{k=1}^N$ ,  $\{d_k^{FCT}\}_{k=1}^N$  and  $\{d_k^{FTF}\}_{k=1}^N$  can be calculated. Thus, twelve shape descriptors can be obtained for a molecule. Every molecule can be denoted as a vector of these shape descriptors:  $\mathbf{M} = \{\mu_1^{CTD}, \mu_2^{CTD}, \mu_3^{CTD}, \mu_1^{CST}, \mu_2^{CST}, \mu_3^{CST}, \mu_1^{FCT}, \mu_2^{FCT}, \mu_3^{FCT}, \mu_1^{FTF}, \mu_2^{FTF}, \mu_3^{FTF}\}$ . Once each molecule in a database is described with these shape descriptors, a normalized score function is used to quantify the degree of similarity between molecules. USR uses the inverse of the translated and scaled Manhattan distance between both vectors of shape descriptors, where a value of 1 corresponds to maximum similarity and 0 to minimum similarity. The similarity score of USR is defined in Equation (2.3) where  $m_l$  is the  $l^{th}$  element in the 12 shape descriptors of a query molecule, and  $m'_l$  is the  $l^{th}$  element in the 12 shape descriptors of a target molecule,

$$S = \frac{1}{1 + \frac{1}{12} \sum_{l=1}^{12} \|m_l - m'_l\|} \in (0, 1]. \quad (2.3)$$

The descriptors that some non-superimposing methods use do not contain any chemical information. These methods are less effective than superimposing methods. Andrew et al. suggests that it is better to use the methods described in [1, 2]

as pre-screening methods to search a chemical structure database rather than as a stand-alone application. Hence, some non-superimposing methods cannot ensure success for all cases. Sometimes it is best if a user can look at the results or combine non-superimposing methods with a superimposing method. In addition, although the comparison calculations may seem efficient, the descriptors used in these methods must be pre-computed for every molecule, as their computation is quite time-consuming. For example, Pedro et al. [5] reported that a database of just 113, 331 molecules takes about 1600 hours to calculate the shape signature. Also, the storage of these descriptors needs additional space. The USR is reported in [5] to be both effective and efficient among these non-superimposing methods. The descriptors it uses are easy to calculate and this algorithm does find structures similar to the target structure. This method essentially calculates the similarity between molecules using those statistical measurements of interest to the chemical researcher. In [5] they find the most similar molecules within a multi-billion molecule database in a matter of minutes using a single processor, which is more than three orders of magnitude faster than the previous fastest method.

## 2.2 Superimposing methods

Superimposing methods attempt to find the best overlap between molecules, including the correct assignment of atoms, and measuring the similarity using root-mean-square-distance or angles. The general idea of all superimposing methods is the same, that is, to find the best superimposition by rotation and translation of one of the structures [6, 8–11].

Nissink et al. [6] uses a crystallographic Fourier transform approach to superimpose molecules. For two molecules superimposed in space, a similarity index can be

calculated based on electron density. A molecule can be represented by electron distribution in Fourier space based on X-ray crystallography. A molecule is placed at the center of a cubic unit cell of side  $a$  which is part of a hypothetical cubic lattice. The continuous electron distribution is described by a set of reciprocal lattice vectors by Laue indices,  $\mathbf{h}$ , and their corresponding structure factors,  $F_{\mathbf{h}}$ ,

$$F_{\mathbf{h}} = \int_{V_{cell}} \rho(\mathbf{r}) \exp(2\pi i \mathbf{h} \cdot \mathbf{r}) V d\mathbf{r}, \quad (2.4)$$

where  $\mathbf{r}$  is the positional vector of the unit cell,  $\rho$  is the electron density distribution and  $V$  is the volume of the unit cell.  $F_h$  also can be calculated as a complex vector of length  $|F_h|$  and phase  $\varphi_{\mathbf{h}}$ :

$$F_{\mathbf{h}} = |F_h| \exp(i\varphi_h) \quad (2.5)$$

The optimal superimposition is achieved when the overlap of electron density is maximized by the following function:

$$\sum_{\mathbf{h}} \|F_{\mathbf{h}}^A\| \|F_{\mathbf{h}}^B\| \cos(\varphi_{\mathbf{h}}^A - \varphi_{\mathbf{h}}^B) - 2\pi \mathbf{h} \cdot \mathbf{y}, \quad (2.6)$$

where  $\varphi_{\mathbf{h}}^A$  and  $\varphi_{\mathbf{h}}^B$  are phases of structure factors  $F_{\mathbf{h}}^A$  and  $F_{\mathbf{h}}^B$ , and  $\mathbf{y}$  is the translation vector to be optimized given the orientation of structure  $A$  and  $B$  [6]. The structure factor can be calculated by Equation (2.4) and the phase can be calculated by Equation (2.5). There are also many other molecular similarity methods based on electron density, which are outside of the scope of this thesis. A detailed review can be found in Popelier [7]. Gironés et al. [8] introduced a superimposing method by overlapping the largest substructures between two molecules. The most popular and intuitive method is trying to find the rotation that minimizes the distance be-

tween corresponding atoms [9–11]. All of the methods that use this approach find the overlap that minimizes the root mean square deviation (RMSD) as in Equation (2.7), where  $N_A$  and  $N_B$  denote the number of atoms in molecules  $A$  and  $B$  respectively,  $(x_i^A, y_i^A, z_i^A)$  and  $(x_i^B, y_i^B, z_i^B)$  are the Cartesian coordinates of the  $i^{th}$  atom in molecules  $A$  and  $B$ ,

$$RMSD = \sum_{i=1}^{\min(N_A, N_B)} \sqrt{\frac{(x_i^B - x_i^A)^2 + (y_i^B - y_i^A)^2 + (z_i^B - z_i^A)^2}{\min(N_A, N_B)}}. \quad (2.7)$$

Henceforth in this thesis, two molecules to be compared will be denoted by superscript  $A$  and  $B$ , with the above notation for the Cartesian coordinates. The subscript refers to a particular atom within a molecule, and in the above case, the atoms between the two molecules have been paired. The difference between existing RMSD methods is in the approach used to find the required rotation. Liu and Van Rapenbusch [10] presented a method using the Eulerian matrix, Lesk [11] published a method based on polar angles, and the most preferable method uses quaternion parameterizations [9]. Since quaternions include information about the rotation axis and the rotation angle, they avoid extra parameters to represent a rotation, as required by other methods. The principle of using quaternions will be introduced in detail in the next chapter.

Each of above superimposing methods has its shortcomings. The method in [6] needs additional electron density information of a molecule that is not always available. The method in [8] is trying to find the most similar molecule by overlapping the largest identical substructure of two molecules. This method works fine for similar molecules with identical atoms but may fail at some cases when the overall geometric structure is very similar or the same but the atoms are not. Methods introduced in [9–11] suffer from the labeling problem of atoms since these methods require the correct mapping between the atoms of the two molecules being compared. Recently,

Vásquez-Pérez et al. [12] designed a superimposing algorithm that solves the atom-labeling problem and finds the best superimposition in a self-consistent way. However, this method uses a probabilistic approach, which applies random rotations to the query structure, until a minimum RMSD is found 50 times. Owing to the randomness of the approach, it is possible that the 50-times-occurred RMSD is not the true minimum RMSD. Thus, its reliability is arguable. Furthermore, since the 50 repetitions is an arbitrary criteria, the run-time performance of the approach is also arbitrary. In the next chapter, an effective and intuitive algorithm that computes the similarity between molecules based on finding the correct assignment and overlap between atoms will be presented.

# Chapter 3

## Objectives and Approach

### 3.1 Objectives

The objective for this thesis is to design a robust algorithm based on the superimposition method that can compare two chemical molecules and produce a similarity score. The algorithm requires the types of the atoms (as defined in Staveley [13]), their Cartesian coordinates, and their connectivity information as input. The atom type of an atom is based on its atomic symbol, the atomic numbers of its connected (nearest-neighbour) atoms, and the valency (that is, the number of atoms that it can bond to). It is based on the proprietary descriptor as described in Staveley’s thesis [13]. The connectivity can be calculated from the Cartesian coordinates of a molecule’s atoms by many chemical software packages such as Jmol [14]. As it is a basic calculation in computation chemistry, it is assumed to be given (input) for the algorithm presented here.

Usually to find very similar items, one wants to maximize a similarity score. However, in this thesis, the smaller the value of the similarity score, the more similar the two molecules. The algorithm can also be used to find molecules similar to a query

molecule by searching a database of chemical structures.

## 3.2 Approach

The basic purpose of the algorithm is to measure the similarity between two chemical structures by finding the best superimposition in three dimensional space. This superimposition is performed by rotating one of the molecules to match the other.

The following sections will introduce the algorithm in detail. First, a measurement is defined to describe the similarity between two molecules. Second, a method is presented to pair the atoms in two molecules. The reason for including this step, rather than finding the rotation directly, is discussed. This pairing method is the novel contribution of the thesis. Finally, an existing method that is used to find the rotation that achieves the best superimposition of two molecules is introduced [9].

The complete algorithm is illustrated by the flow chart in Figure 3.1.



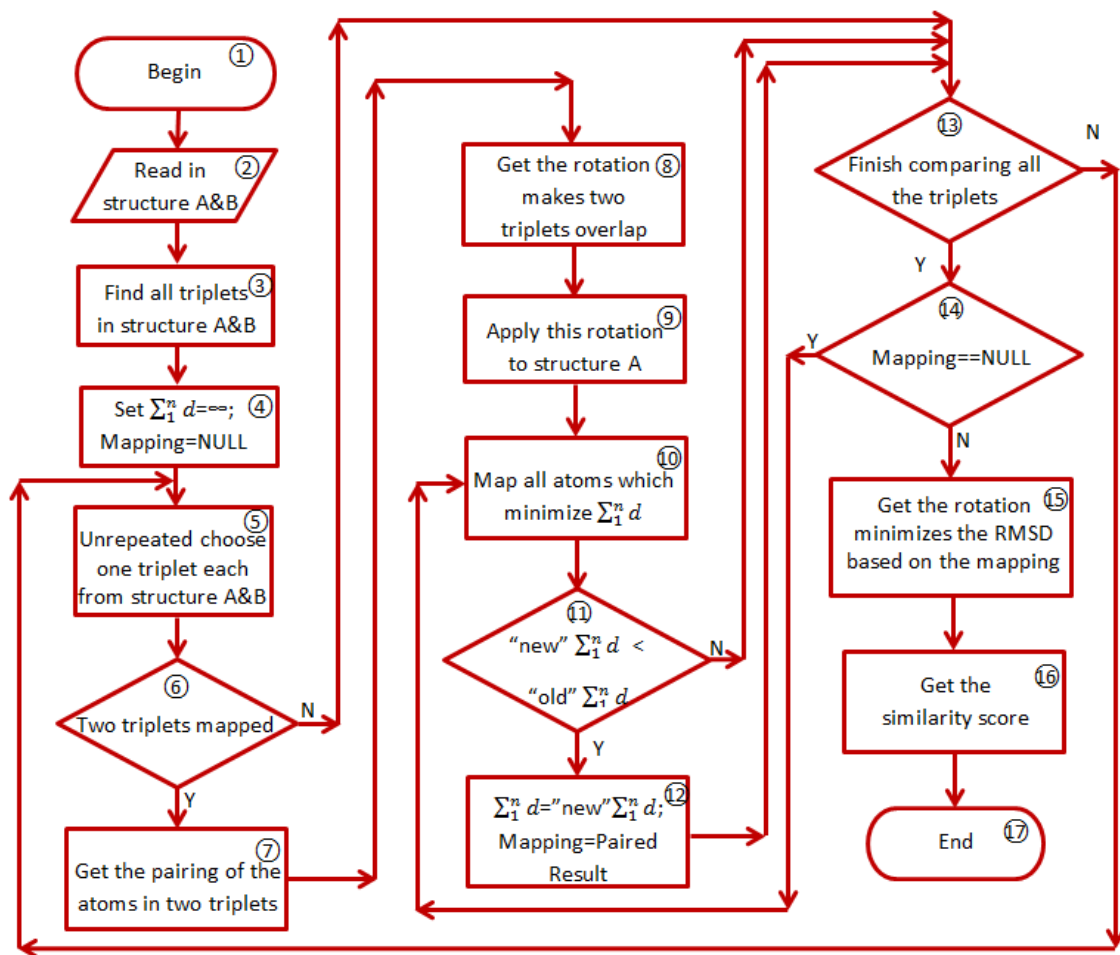


Figure 3.1: Flow chart illustrating the algorithm to compute the similarity between two molecules.

A summary of the algorithm is as follows (the step number corresponds to the circled number in each box of the flow chart in Figure 3.1), with the details presented in the following sections:

1. Algorithm starts.
2. Read in structural information of two molecules that includes Cartesian coordinates, type of each atom (as defined in Staveley [13]), and connectivity information.

3. Find all atom triplets in each molecule. For each atom triplet, three atoms have to be bonded in the molecule and cannot be co-linear in geometrical space (the definition of atom triplet is introduced in Section 3.2.2.1).
4. Initialize the variable used to save the sum of all the distances between the mapped atoms of two molecules, as well as the variable used to save the potential correct mapping of atoms between the two molecules.
5. Each time through the loop (Step 5 to Step 13), without replacement, choose one triplet from molecule A and one triplet from molecule B to compare.
6. Compare the triplet in molecule A to the triplet in molecule B. If the atoms in the two triplets are mapped, continue to next step. Otherwise go to step 13 (details about how to distinguish if the atoms in the two triplets are mapped are introduced in Section 3.2.2.1).
7. Compute the mapping of each atom for matched triplets (details about how to map the atoms for mapped triplets are introduced in Section 3.2.2.1).
8. Compute the rotation that makes the atoms of two triplets overlap (details about how to get the rotation are introduced in Section 3.2.2.2).
9. Apply this rotation to molecule A.
10. Run the Hungarian algorithm to get the best mapping of all the atoms in the two molecules for the current structural orientation (rationale for the Hungarian algorithm and how to apply it are introduced in Section 3.2.2.1).
11. If the sum of all the distances between the mapped atoms in the two molecules is smaller than the saved one, proceed to next step. Otherwise, go to step 13.

12. Replace the saved sum of all the distances between the mapped atoms, as well as the mapping information with the newly calculated ones.
13. If finished comparing all the triplets of two molecules, continue. Otherwise, go to step 5.
14. If the variable for saving the correct mapping is NULL (which means there is no mapping between all the triplets in the two molecules), then, go to step 10 to obtain the mapping based on the original orientation of the two molecules. Otherwise, go to next step.
15. Get the rotation that minimizes the RMSD between the two molecules based on the best mapping of atoms achieved from the above steps (details about how to get the rotation are introduced in Section 3.2.2.2.)
16. Calculate the similarity score (details of defining similarity score are introduced in Section 3.2.1.)
17. Algorithm ends.

### 3.2.1 Definition of similarity

As introduced in the last chapter, from a geometric point of view, if two chemical molecules are the same, then they can be fully overlapped in three dimensional space. The distance between the corresponding atoms is then zero. If they are similar molecules, some of the corresponding atoms may be overlapped while others may be close to each other when they are at their best superimposition. Thus, the chosen superimposing method uses the root mean square deviation (RMSD in Equation (2.7)) as a similarity measure. The smaller the RMSD, the more similar the molecules

are. However, some circumstances may exist where the RMSD is not enough to describe the similarity adequately. For example, the overall geometrical structure of two molecules may be very similar or the same while the atom types are different. For this case, the RMSD is the same (zero) even though there are different atoms between these two molecules. Obviously, it is unreasonable that two molecules with more identical atoms have the same similarity score as two molecules with less identical atoms, even if the geometrical structure is the same. If a chemist wants to search a database for a molecule similar to a query molecule using this similarity score, he may find many molecules with the same (minimal) RMSD. Among these molecules with the same minimal RMSD, some will have the same geometrical structure but different atom types from the query molecule. In this case, a chemist has to check the atom types manually in order to find the most similar molecule.

Although the RMSD has its limitations, it is still useful to describe the structural differences on a geometric basis. In order to take atom types into consideration when measuring the similarity between the two molecules, the algorithm uses both the RMSD and the atom type to measure the similarity of two molecules. The algorithm calculates the RMSD and the number of identical atoms between the two molecules. It then uses the ratio of the number of identical atoms to the total number of atoms (the “similarity ratio”) as a weight for the similarity score RMSD. In the case where the two molecules contain a different number of atoms, the total number of atoms is the number of atoms in the larger molecule. Therefore, the similarity score ( $S$ ) between any two molecules (denoted as  $A$  and  $B$ ) is defined as the RMSD between the paired atoms divided by the similarity ratio:

$$S = \frac{\sum_{i=1}^{\min(N_A, N_B)} \sqrt{\frac{(x_i^B - x_i^A)^2 + (y_i^B - y_i^A)^2 + (z_i^B - z_i^A)^2}{\min(N_A, N_B)}}}{\frac{N_{A=B}}{\max(N_A, N_B)}}, \quad (3.1)$$

where  $N_{A=B}$  is the number of identical atoms in  $A$  and  $B$ . For molecules with the same geometrical structure and identical atoms, the similarity score is equal to the RMSD. Note that when RMSD is equal to zero (identical structure), the similarity ratio  $([0, 1])$  will not have any effect. As a result, structures with identical geometry but different atoms would still have a similarity score of zero. In practice, due to round-off errors in computers, even for two identical molecules, the calculated RMSD value may be close to zero but will never equal zero.

### 3.2.2 Comparison algorithm

This section will introduce and explain the comparison algorithm. The algorithm requires only the structural files for the molecules to be compared, which contain the Cartesian coordinates, atom types (as defined in Staveley [13]) and the connectivity information of the atoms in the molecule. The algorithm will then find the best superimposition of the two molecules and calculate a similarity score. In this thesis, the best superimposition is considered to be the superimposition with minimum RMSD of the two compared structures. In contrast to the available superimposition algorithms [9–11], the Cartesian coordinates of the atoms do not need to be in the same order in the two structural files. The algorithm will automatically find the mapping of the corresponding atoms in the two molecules for the user.

As introduced in Chapter 2, all the available algorithms compute the similarity by rotating one molecule until the best superimposition of the two molecules is achieved [9–11]. All the equations to calculate the rotation require values for corresponding atoms. These algorithms all assume that the order of the atoms in the structural files of the two molecules to be compared is the correct mapping. However, for general chemical structures, there is no conventional ordering for the atoms in the structural file. Even for the same molecule, the sequence of labeling the atoms

in the structural file may vary because the file was created by different applications or different users. Mapping the atoms of two molecules according to the default order of atoms in the structural files may produce the wrong result. Most available algorithms suffer from this labeling problem [9–11]. Applications like Jmol try to solve this problem by allowing the user to provide the correct mapping information manually [14]. For molecules with few atoms, this may not be such an onerous task. However, it is unrealistic for the user to provide the correct mapping information for complex molecules with tens or hundreds of atoms (or more). Furthermore, if the users want to use this algorithm to find a similar molecule from a database containing hundreds of thousands or millions of molecules, it is impossible for them to provide the mapping information of all the atoms of all of these molecules. As a result, superimposition methods without correct mapping cannot be used to search a database to find the most similar molecule to a query molecule, even though they can provide more accurate results.

Thus, the algorithm has two parts. The first part will find the correct mapping of the atoms between the two molecules to be compared. The second part will use known methods to find the best superimposition of the two molecules based on the identified mapping, and then provide a similarity score.

### 3.2.2.1 Finding the correct mapping of the atoms

As mentioned in Section 3.2.1, this algorithm is looking for the minimum similarity score (Equation (3.1)). For any two molecules, the values of  $N_A, N_B, N_{A=B}$  (number of atoms) are constant. Thus, this algorithm is looking for the minimum value of the following equation, which is the sum of all the distances between the corresponding atoms,

$$d = \sum_{i=1}^{\min(N_A, N_B)} \sqrt{(x_i^B - x_i^A)^2 + (y_i^B - y_i^A)^2 + (z_i^B - z_i^A)^2}. \quad (3.2)$$

However, as mentioned in the previous section, the labeling order of the atoms in molecule A may not be the same as in molecule B. The correct mapping between the atoms in the two molecules is unknown. The most intuitive way to find the minimum  $d$  is to try all the different mappings of atoms between two molecules and to see which mapping minimizes the sum of all the distances between the corresponding atoms. However, this method is too computationally expensive. If there are  $N$  atoms in each molecule, then  $N!$  mappings need to be tried. An algorithm developed by Kuhn and Munkres [15] solves this kind of general assignment problem. The algorithm is called the Hungarian algorithm. It is a linear programming algorithm designed for solving the problem of personnel-assignment. This problem is to find the best assignment of a set of  $n$  persons to a set of  $n$  jobs, where the possible assignments are ranked by the total scores or ratings of the workers in the jobs to which they are assigned [15, 16]. The Hungarian algorithm reduces the time complexity of the problem from  $n!$  to  $n^3$ . Here, this algorithm is applied to solve the mapping problem of the atoms in two molecules to be compared.

In our case, we can consider the set of atoms in one molecule as a set of persons and the set of atoms in the other molecule as a set of jobs. The result of the Hungarian algorithm provides the assignment between corresponding atoms that we are looking for.

The Hungarian algorithm requires a matrix containing ranking information between the persons and the jobs. Usually the ranking values represent the efficiency of person  $i$  performing job  $j$ . Here, we rank all the atoms according to the atom type (as defined in Staveley [13]) and the distances between every two atom pairs. The information can be presented effectively by a matrix  $F$ , in which horizontal rows (persons) correspond to atoms of one molecule and vertical columns (jobs) correspond to atoms of the other molecule. An element  $F_{ij}$  of the matrix is the ranking between the  $i^{th}$  atom in

molecule A and the  $j^{th}$  atom in molecule B. We use the distance between these two atoms, plus a penalty if the atoms are different, as their rank. If we use the same notation as in the previous section, we define,

$$F_{ij} = \sqrt{(x_j^B - x_i^A)^2 + (y_j^B - y_i^A)^2 + (z_j^B - z_i^A)^2} + b, \quad (3.3)$$

where  $b$  is the penalty [17]. To avoid the wrong assignment between two structurally equivalent atoms having a different atom type, we set  $b = 0$  if atom  $i$  and  $j$  have the same atom type and  $b = 10$  otherwise [17]. Other positive quantities for  $b$  could also be used as long as it is relatively larger than 0. If the number of atoms of the two structures is different, virtual atoms are added to the structure with fewer atoms in order to make the number of atoms of the structures the same. The value of  $F_{ij}$  between virtual atoms to the real atoms in the other structure is considered to be infinite. In practice, any value that is relatively larger than the distances between atoms can be used. In the implementation of this algorithm, 100 is used for this value. With this matrix, we can use the Hungarian algorithm to find the mapping between the atoms of the two molecules. Since we are looking for a mapping that minimizes the sum of the distances between all the corresponding atoms, and the rankings correspond to their distances, we are looking for the mapping that minimizes the sum of the rank, as is the case in the personnel-assign problem. The Hungarian algorithm is based on the theorem that if a number is added to or subtracted from all of the entries of any one row or column of a cost matrix, then an optimal assignment for the resulting cost matrix is also an optimal assignment for the original cost matrix [15]. The algorithm applies to the above generated  $N \times N$  ( $N$  is the larger value of  $N_A$  and  $N_B$ ) cost matrix to find an optimal assignment (the mapping) for the atoms is follows:



1. Subtract the smallest entry in each row from all the entries of the row.
2. Subtract the smallest entry in each column from all the entries of the column.
3. Draw lines through appropriate rows and columns so that all the zero entries of the cost matrix are covered and the minimum number of such lines is used.
4. If the minimum number of lines drawn in step 3 is  $N$ , an optimal assignment of zeros is possible and the algorithm is finished. If the minimum number of lines is less than  $N$ , proceed to next step.
5. Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to step 3.

In order to present a clearer illustration of this algorithm, consider the following example. Given a cost matrix

$$\begin{bmatrix} 90 & 75 & 75 & 80 \\ 35 & 85 & 55 & 65 \\ 125 & 95 & 90 & 105 \\ 45 & 110 & 95 & 115 \end{bmatrix},$$

where the (fictional) distances are taken to be integers for convenience. The smallest entries of each row are indicated in red:

$$\begin{bmatrix} 90 & \mathbf{75} & 75 & 80 \\ \mathbf{35} & 85 & 55 & 65 \\ 125 & 95 & \mathbf{90} & 105 \\ \mathbf{45} & 110 & 95 & 115 \end{bmatrix}.$$

**Step 1.** Subtract 75 from Row 1, 35 from Row 2, 90 from Row 3 and 45 from Row 4 to obtain:

$$\begin{bmatrix} 15 & \mathbf{0} & \mathbf{0} & \mathbf{5} \\ \mathbf{0} & 50 & 20 & 30 \\ 35 & 5 & 0 & 15 \\ 0 & 65 & 50 & 70 \end{bmatrix}.$$

**Step 2.** Subtract 0 from Column 1, 0 from Column 2, 0 from Column 3 and 5 from Column 4 to obtain:

$$\begin{bmatrix} 15 & 0 & 0 & 0 \\ 0 & 50 & 20 & 25 \\ 35 & 5 & 0 & 10 \\ 0 & 65 & 50 & 65 \end{bmatrix}.$$

**Step 3.** Cover all the zeros of the matrix with the minimum number of horizontal and/or vertical lines. In order to do so, for each cell,  $m_{ij}$ , that has a value zero, determine which direction has the most zeros, row  $i$  or column  $j$  (previously covered cells with zeros do not count). Draw a line through that row or column. If the number of zeros in the row is equal to the number of zeros in the column, then draw a line in either direction. Repeat the above steps until all the cells that have a value zero are covered.

$$\begin{bmatrix} \overline{15} & \overline{0} & \overline{0} & \overline{0} \\ 0 & 50 & 20 & 25 \\ \overline{35} & 5 & \overline{0} & 10 \\ \overline{0} & 65 & \overline{50} & \overline{65} \end{bmatrix}.$$

**Step 4.** Since the minimum number of lines is less than 4, proceed to Step 5.

**Step 5.** Note that 5 is the smallest entry not covered by any line. Subtract 5 from each uncovered row to obtain:

$$\begin{bmatrix} 15 & 0 & 0 & 0 \\ -5 & 45 & 15 & 20 \\ 30 & 0 & -5 & 5 \\ -5 & 60 & 45 & 60 \end{bmatrix}.$$

Now add 5 to each covered column to obtain:

$$\begin{bmatrix} 20 & 0 & 5 & 0 \\ 0 & 45 & 20 & 20 \\ 35 & 0 & 0 & 5 \\ 0 & 60 & 50 & 60 \end{bmatrix}.$$

and return to Step 3.

**Step 3.** Cover all the zeros of the matrix with the minimum number of horizontal or vertical lines.

$$\begin{bmatrix} \cancel{20} & 0 & 5 & 0 \\ 0 & 45 & 20 & 20 \\ \cancel{35} & 0 & 0 & 5 \\ 0 & 60 & 50 & 60 \end{bmatrix}.$$

**Step 4.** Since the minimum number of lines is less than 4, proceed to Step 5.

**Step 5.** Note that 20 is the smallest entry not covered by a line. Subtract 20 from each uncovered row to obtain:

$$\begin{bmatrix} 20 & 0 & 5 & 0 \\ -20 & 25 & 0 & 0 \\ 35 & 0 & 0 & 5 \\ -20 & 40 & 30 & 40 \end{bmatrix}.$$

Then add 20 to each covered column,

$$\begin{bmatrix} 40 & 0 & 5 & 0 \\ 0 & 25 & 0 & 0 \\ 55 & 0 & 0 & 5 \\ 0 & 40 & 30 & 40 \end{bmatrix}.$$

and return to Step 3.

**Step 3.** Cover all the zeros of the matrix with the minimum number of horizontal or vertical lines.

$$\begin{bmatrix} \cancel{40} & \cancel{0} & \cancel{5} & \cancel{0} \\ \cancel{0} & 25 & \cancel{0} & \cancel{0} \\ \cancel{55} & \cancel{0} & \cancel{0} & 5 \\ \cancel{0} & \cancel{40} & \cancel{30} & \cancel{40} \end{bmatrix}.$$

**Step 4.** Since the minimum number of lines is 4, an optimal assignment of zeros (one of the optimal assignments is highlighted in red) is possible and the algorithm is finished.

$$\begin{bmatrix} 40 & 0 & 5 & \mathbf{0} \\ 0 & 25 & \mathbf{0} & 0 \\ 5 & \mathbf{0} & 0 & 5 \\ \mathbf{0} & 40 & 30 & 40 \end{bmatrix}.$$

It can be seen that there exists multiple optimal mappings in this case. Because the Hungarian algorithm is designed for finding the optimal mapping of personnel-assignment, it is possible that multiple optimal mappings exist. However, when the two molecules are overlapped, there only exists one optimal mapping between the

molecules unless one or both the molecules are symmetric or contains symmetric substructures. The different optimal mappings are just the different mappings between the symmetric atoms. It does not matter which mapping is used to find the rotation because superimposition of the molecule after rotation remains the same. Thus, any optimal mapping can be used to find the rotation for the algorithm of this thesis. Therefore, the atoms are mapped according to the result of the Hungarian algorithm.

The result generated by the above algorithm is only the best mapping in one orientation of the two molecules. However, a molecule can be translated to any position or rotated by any angle in a structural file. An example from Vásquez-Pérez et al. [12] describes this problem nicely in two dimensional space. Although it is not a real world example and is not in three dimensional space, the principle is the same. Presenting it in two dimensional space gives a clearer view of this problem.

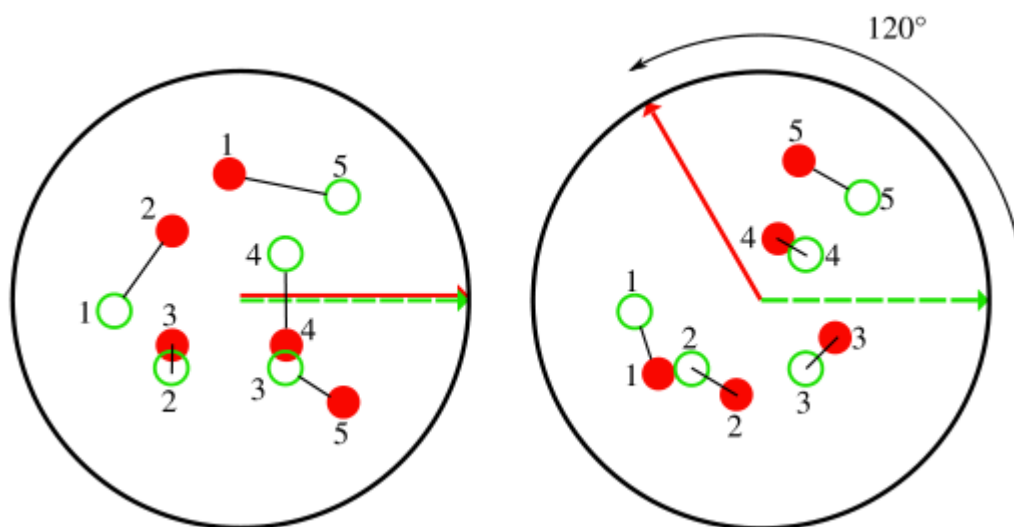


Figure 3.2: A pair of two-dimensional “molecules” of five “atoms” at two different orientations. The green circles represent the location of the atoms in molecule A and the red circles represent the location of the atoms in molecule B. (redrawn from [12])

Consider the two molecules depicted in Figure 3.2. The green circles represent the location of the atoms in molecule A, while the red circles represent the location of

the atoms in molecule B. With the original orientation of the molecules (left), the pairing (based on minimum distance) is not ideal (1-2, 2-3, 3-5, 4-4, 5-1). However, by rotating molecule B (red) by  $120^\circ$  (right), a much better pairing is obtained (1-1, 2-2, 3-3, 4-4, 5-5). Although these two mappings are generated by the Hungarian algorithm, the different orientations of the molecules produce different results. If we assume the numbers correspond to atom types, the second mapping is better since it is the same mapping as when the two structures are fully overlapped. For the second mapping, the best superimposition (fully overlapped) can be achieved by rotating the red “atoms” counterclockwise by  $90^\circ$  with respect to the green “atoms”. Additionally, we can see that rotating the red “atoms” in the first orientation counterclockwise by any angle in the range of  $90^\circ \sim 120^\circ$  allows the Hungarian algorithm to produce the correct mapping. So more than one orientation may produce the correct mapping. There exists an orientation neighbourhood within which the mapping produced by the Hungarian algorithm is correct. In [12], a probability driven approach is used to ensure the orientation produces the correct mapping. In this approach, one molecule is rotated by an arbitrary angle and the above Hungarian algorithm is applied to get the mapping. Based on this mapping, one molecule is then rotated to obtain the minimum value of Equation (3.2). These two steps are repeated until the same minimum  $d$  value in Equation (3.2) is found 50 times. The position that produces this minimum  $d$  value is considered to be the best superimposition. In the paper, a good result is reported by applying the method to two series of carbon fullerenes ( $C_{60}$ ,  $C_{74}$ ,  $C_{180}$ ,  $C_{240}$ ) that were randomly substituted by boron atoms [12]. However, with the probability driven approach, it is arguable whether success with other molecules can be guaranteed. It is possible that a selection of random rotations may not produce the best mapping.

The basic idea of the method of [12] is to look for a correct mapping such that the sum of the distances between matched atoms is minimized over different orientations.

Thus, the algorithm presented in this thesis needs to ensure that the orientation of the two molecules are within the neighbourhood of producing the best mapping. Since we are interested in finding similar molecules, for every two molecules to be compared, we are looking for a substructure of each molecule that is similar. At the best superimposition of the two molecules, there must be some substructure(s) of these two molecules overlapping or almost overlapping. When these substructures of the two molecules overlap, although that position may not be the best superimposition of the two molecules, it is sufficient to find an orientation that will produce the best mapping. Thus, by applying the rotation that makes the substructures overlap, the algorithm can adjust the orientation of one molecule in order to make the two molecules in the orientation produce the correct mapping. The algorithm needs at least three atoms for a substructure, because in three dimensional space, at least three points are needed to form a plane to define the rotation. The orientation of the structures can be adjusted by making these two planes overlap. Although, not every rotation that makes these substructures overlap will place the two molecules in the orientation to produce the correct mapping, there must exist at least one substructure rotation that will place the two molecules in the orientation for producing the correct mapping.

In order to find the best mapping of atoms between two molecules, for all the atoms in the first molecule, the algorithm will find all the triplets consisting of three atoms. The three atoms in a triplet cannot be co-linear (they should be able to form a unique plane) and they have to be bonded in the molecule (reducing the number of triplets to be compared). A valid triplet of atoms could be defined in graph terms if the molecule is thought as a graph: nodes are the atoms, bonds are the edges. A valid triplet is a subgraph with exactly three nodes where there are at least two paths connecting two different node-pairs in the subgraph and such a path does not visit a node which is not in the subgraph. Triplets are mapped (atoms in these two triplets are considered to

be corresponding atoms) if they are identical substructures. Identical substructures are a valid atom triplet in the target molecule and a valid atom triplet in the query molecule such that the distances between every two atoms are the same and the atom type (as defined in Staveley [13]) is same for the corresponding atoms. Let us call the set of triplets of molecule  $A$  set  $T_A$ . Similarly, the triplets of molecule  $B$  are set  $T_B$ . Every triplet in  $T_A$  is compared to every triplet in  $T_B$ . For each two triplets to be compared, if the three inter-atom distances are the same as the inter-atom distances in the other triplet and the atoms are also the same type (as defined in Staveley [13]), the atoms in these two triplets are considered corresponding atoms. In Figure 3.3, the atoms that form the green and yellow triangles are two examples of atom triplets found by the algorithm. However, the atoms that form the pink triangle are not an atom triplet, because only atoms 7 and 12 are bonded. Atom triplet (3,4,6) in molecule 2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ ) (Figure 3.3 left), and atom triplet (5,7,8) in molecule 2-hydroxy-5-methylbenzo-1,4-quinone ( $C_7H_6O_3$ ) (right), are corresponding atom triplets (3 corresponds to 8, 4 corresponds to 7, 6 corresponds to 5). The distance between atoms  $A_3$  and  $A_4$  is same as the distance between atoms  $B_7$  and  $B_8$ , the distance between atoms  $A_4$  and  $A_6$  is same as the distance between atoms  $B_5$  and  $B_7$ , the distance between atoms  $A_3$  and  $A_6$  is same as the distance between  $B_5$  and  $B_8$ , the atom type of  $A_3$  is same as  $B_8$ , the atom type of  $A_4$  is same as  $B_7$ , and the atom type  $A_6$  is same as  $B_5$ .



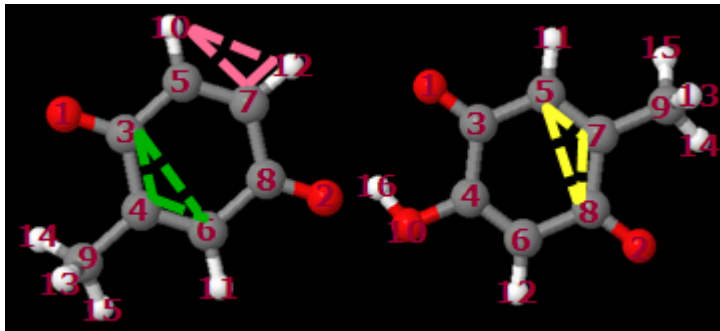


Figure 3.3: Two molecules to be compared, 2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ ) (left) and 2-hydroxy-5-methylbenzo-1,4-quinone ( $C_7H_6O_3$ ) (right). For illustration, the atoms within each structure are arbitrarily numbered.

The mapping information regarding which atom in one triplet corresponds to which atom in the other triplet is recorded in set  $M_{T_A=T_B}$ . For example, let us use  $A_1, A_2, A_3$  to represent three atom types in one molecule  $A$ ,  $B_1, B_2, B_3$  to represent three atom types in the other molecule  $B$  and  $d()$  to represent the distance between two atoms. If

$$d(A_1, A_2) = d(B_1, B_2),$$

$$d(A_2, A_3) = d(B_2, B_3),$$

$$d(A_1, A_3) = d(B_1, B_3),$$

$$A_1 = B_1,$$

$$A_2 = B_2,$$

$$A_3 = B_3,$$

then the algorithm will record the mapping information as  $A_1$  corresponds to  $B_1$ ,  $A_2$  corresponds to  $B_2$ , and  $A_3$  corresponds to  $B_3$ . If

$$d(A_1, A_2) = d(B_2, B_3),$$

$$d(A_2, A_3) = d(B_1, B_2),$$

$$d(A_1, A_3) = d(B_1, B_3),$$

$$A_1 = B_1,$$

$$A_2 = B_2,$$

$$A_3 = B_3,$$

$$A_1 \neq A_2 \neq A_3,$$

this is not a mapping. Although every distance mapped, the atom types did not. If

$$d(A_1, A_2) = d(B_2, B_3),$$

$$d(A_2, A_3) = d(B_1, B_2),$$

$$d(A_1, A_3) = d(B_1, B_3),$$

$$A_1 = B_3,$$

$$A_2 = B_2,$$

$$A_3 = B_1,$$

then the algorithm will record the mapping information as  $A_1$  corresponds to  $B_3$ ,  $A_2$  corresponds to  $B_2$  and  $A_3$  corresponds to  $B_1$ . When recording matching information, all possible mappings for triplets should be saved. For example, if

$$d(A_1, A_2) = d(B_1, B_2) = d(A_2, A_3) =$$

$$d(B_2, B_3) = d(A_1, A_3) = d(B_1, B_3),$$

$$A_1 = B_1 = A_2 = B_2 = A_3 = B_3,$$

then for every atom in  $A_1, A_2, A_3$ , the corresponding atom could be any atom in  $B_1, B_2, B_3$ . Thus, all six possible mappings should be saved as different mappings.

For every mapping of triplets in  $M_{T_A=T_B}$ , the first triplet’s center is mapped to the center of the other triplet. Then, the three atoms in the triplet of molecule  $A$  are used as the origin position and the three atoms in the triplet of molecule  $B$  is used as the final position to find the rotation that makes the two triplets fully overlap. The molecule is rotated based on the calculated rotation. Following the rotation, molecule  $A$  is then translated such that the centre of mass of the two molecules overlap. After that, the mapping of the atoms that minimizes the sum of all the distances between the corresponding atoms of the two molecules is found for the current orientation. This mapping is considered to be a candidate best mapping of all the atoms between the two molecules. One molecule is then translated until the centers of the mapped atoms are overlapped and the sum of all the distances between the corresponding atoms of the two molecules at that position is saved. Because the algorithm is only finding the minimum sum of all the distances between mapped atoms in two molecules, it has to ensure that the centers of the matched atoms are overlapped in order to avoid having the distance between the centers of mapped atoms in the two molecules affecting the sum value. The method used to find the rotation does not include the translation.

The above steps are applied to all mapped triplets in  $M_{T_A=T_B}$ . During this iteration, once a mapping with a smaller sum of the distances between the corresponding atoms of the two molecules has been found, the saved value and the mapping information is replaced. After the above steps are applied to every mapped triplet, the saved mapping is considered to be the best mapping between the atoms in the two molecules.

### 3.2.2.2 Finding the best superimposition

The basic idea of comparing the two chemical structures is to try to find the best superimposition between these two molecules based on the best mapping of the atoms. The popular way of doing this is to rotate one of the molecules until the best overlap is achieved. There are many available methods for parameterizing rotation, such as the arbitrary Eulerian rotation matrix [10], the polar-angle representation [11] and the quaternion-based representation [12]. Quaternions include information about the rotation axis and the rotation angle while other methods require several parameters to represent a rotation. In addition, quaternions are much more computationally efficient since multiplication between two quaternions only requires sixteen multiplications and additions while a  $3 \times 3$  rotation matrix requires twenty-seven operations. Therefore, the more rotation that occurs, the greater the benefit in using quaternion representation. In computer graphics fields such as computer games, conventional methods usually find several key orientations for a rotating object and then use interpolation to obtain every step of the animation. Eulerian rotation matrices may produce gimbal lock such that two rotational axes are pointing in the same direction, and many object positions will not be available for representation, due to the loss of one rotation dimension [18]. Therefore, the object will not rotate how it ought to rotate. This will have a negative impact on the interpolation of the rotation. In addition, in three-dimensional space, orientations and angles are represented by two independent values. It is impossible to normalize them. However, for quaternions, rotation axes and rotation angles are in the same dimensional space, and can be normalized easily. The interpolation results using quaternions are much more reliable. Thus, the quaternion-based representation is more suitable for our needs.

Quaternions are a number system created by mathematician William Rowan Hamilton in 1843 [19,20]. A quaternion is defined as  $q = w + xi + yj + zk$  where  $w$  is a real

number and  $x, y, z$  are pure imaginary. It can be also represented as  $q = (w, \mathbf{v})$ , where  $\mathbf{v}$  is a vector. Let two quaternions be denoted by  $q_1 = w_1 + x_1i + y_1j + z_1k = (w_1, \mathbf{v}_1)$ , and  $q_2 = w_2 + x_2i + y_2j + z_2k = (w_2, \mathbf{v}_2)$ . The multiplication between them is defined as follows:

1. Normal representation:

$$\begin{aligned} q_1 * q_2 = & (w_1w_2 - x_1x_2 - y_1y_2 - z_1z_2) + (w_1w_2 + x_1w_2 + y_1z_2 - z_1y_2)i + \\ & (w_1y_2 - x_1z_2 + y_1w_2 + z_1x_2)j + (w_1z_2 - x_1y_2 - y_1x_2 + z_1w_2)k. \end{aligned} \quad (3.4)$$

2. Vector representation:

$$q_1 * q_2 = (w_1w_2 - \mathbf{v}_1 \cdot \mathbf{v}_2, w_1\mathbf{v}_2 + w_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2). \quad (3.5)$$

If we consider  $w = \cos(\frac{\theta}{2})$ ,  $x = a_x \sin(\frac{\theta}{2})$ ,  $y = a_y \sin(\frac{\theta}{2})$ ,  $z = a_z \sin(\frac{\theta}{2})$ , the quaternion  $q$  represents a rotation about the vector  $(a_x, a_y, a_z)$  through  $\theta$  degrees. As we can see, a quaternion itself can easily represent a rotation with a simple and easy to read form. It is a clearer representation compared to other methods, and the cost of storing or computing the quaternion is much lower as well.

Note that the input files for the algorithm presented in this thesis contain the coordinates of all the atoms of the molecules to be compared. For example, this algorithm is trying to compare molecule  $A$  and molecule  $B$ . It reads in the coordinates of all the atoms in these two molecules. For a certain atom in molecule  $A$ , its Cartesian coordinate can be represented as a vector  $\mathbf{p}_A(x^A, y^A, z^A)$ . A rotation is desired that will overlap  $\mathbf{p}_A$  with its correspondent atom  $\mathbf{p}_B(x^B, y^B, z^B)$  in the other molecule.  $\mathbf{p}_A$  and  $\mathbf{p}_B$  are converted from vectors to quaternions. When converting vectors to four-dimensional space, as in a quaternion, the representation  $q = (w, \mathbf{v})$  mentioned

above can be used. The real part  $w$  is set to zero while the quaternion's imaginary part is set to the coordinates of vector  $\mathbf{p}_{A/B}$ . After two quaternions are represented by  $\mathbf{P}_A(0, \mathbf{p}_A)$  and  $\mathbf{P}_B(0, \mathbf{p}_B)$ , a rotation is sought, after which  $\mathbf{p}_A$  will be overlapped with  $\mathbf{p}_B$ . This rotation can be represented by  $\mathbf{Q}$  in the following equation:

$$\mathbf{P}_B = \mathbf{Q}^{-1} \mathbf{P}_A \mathbf{Q}, \quad (3.6)$$

where  $\mathbf{Q}$  is the quaternion that represents the rotation. Note that  $\mathbf{Q}$  has to be a unit quaternion [9], and  $\mathbf{Q}^{-1}$  is the conjugate quaternion of  $\mathbf{Q}$ . If a quaternion  $\mathbf{q} = w + xi + yj + zk$ , then its conjugate quaternion is  $\mathbf{q}^{-1} = w - xi - yj - zk$  [9]. The desired rotation is found by solving Equation (3.6) for  $\mathbf{Q}$ .

The above case is only for one atom in two molecules. Obviously, a chemical structure will not consist of a single atom. If Equation (3.6) is applied to different atom pairs between the two molecules, by solving these equations, different quaternions representing different rotations may be obtained. For a given pair of molecules, the algorithm seeks a single rotation to be applied to all the atoms, which results in maximum overlap of the two molecules. In other words, one particular quaternion that can minimize the total distance between all the atom pairs is sought.

Consider the case where the molecules  $A$  and  $B$  mentioned in the above example each have  $N$  atoms, the coordinate information of all the atoms in molecule  $A$  is represented as a vector set  $\{(x_1^A, y_1^A, z_1^A), (x_2^A, y_2^A, z_2^A) \dots (x_N^A, y_N^A, z_N^A)\}$  and the coordinate information of all the atoms in molecule  $B$  is represented as a vector set  $\{(x_1^B, y_1^B, z_1^B), (x_2^B, y_2^B, z_2^B) \dots (x_N^B, y_N^B, z_N^B)\}$ . Recall that the best superimposition is achieved by minimizing Equation (3.2). In order to find the rotation to achieve this minimum value, we need to convert the vector sets that represent the coordinate values of all the atoms in the two molecules to quaternion representation. All the atoms

in molecule A can be denoted as  $\{\mathbf{v}_i^{\mathbf{A}}\}(i = 1...N)$ . The same representation for atoms in molecule B can be given as  $\{\mathbf{v}_i^{\mathbf{B}}\}(i = 1...N)$ . The quaternion representation is then given by  $(0, \mathbf{v}_i^{\mathbf{A}})(i = 1...N)$  and  $(0, \mathbf{v}_i^{\mathbf{B}})(i = 1...N)$ . The rotation to superimpose two molecules is represented by  $\mathbf{Q} = (q_1, q_2, q_3, q_4)$ . This rotation  $\mathbf{Q}$  will not make the atom postions in one molecule exactly match all the atom position vectors in the other molecule. Consequently, the residual (distance) between the  $i^{th}$  corresponding atom pair is defined as a vector  $\mathbf{e}_i$  [21]:

$$(0, \mathbf{e}_i) = (0, \mathbf{v}_i^{\mathbf{A}}) - \mathbf{Q}^{-1}(0, \mathbf{v}_i^{\mathbf{B}})\mathbf{Q}. \quad (3.7)$$

So if an atom pair is fully overlapped after rotation, the residual distance will be zero, otherwise, it will have a positive value. As discussed in the previous section, the rotation  $\mathbf{Q}$  is determined by minimizing  $d$  in Equation (3.2). It is also determined by minimizing  $d^2$ , which here is the sum of the squared magnitudes of these residual vectors  $\sum_{i=1}^N |\mathbf{e}_i|^2$ , which is equals  $\sum_{i=1}^N |(0, \mathbf{e}_i)|^2$ . In order to make the problem tractable, we first left multiply Equation (3.7) through by  $\mathbf{Q}$  to give:

$$\mathbf{Q}(0, \mathbf{e}_i) = \mathbf{Q}(0, \mathbf{v}_i^{\mathbf{A}}) - (0, \mathbf{v}_i^{\mathbf{B}})\mathbf{Q}. \quad (3.8)$$

Since  $\mathbf{Q} = (q_1, q_2, q_3, q_4) = (q_1, \mathbf{q})$ , where  $\mathbf{q} = (q_2, q_3, q_4)$ , a slightly modified least squares residual function is constructed by expanding polynomials in Equation (3.8) according to Equations (3.4) and (3.5) [21]:

$$\begin{aligned} \varepsilon &= \sum_{i=1}^N |\mathbf{Q}(0, \mathbf{e}_i)|^2 = \sum_{i=1}^N |\mathbf{Q}|^2 |\mathbf{e}_i|^2 = \sum_{i=1}^N |(-\mathbf{q} \cdot (\mathbf{v}_i^{\mathbf{A}} - \mathbf{v}_i^{\mathbf{B}}), q_1(\mathbf{v}_i^{\mathbf{A}} - \mathbf{v}_i^{\mathbf{B}}) + \mathbf{q} \times (\mathbf{v}_i^{\mathbf{A}} + \mathbf{v}_i^{\mathbf{B}}))|^2 \\ &= \sum_{i=1}^N \{[q_2(x_i^{\mathbf{A}} - x_i^{\mathbf{B}}) + q_3(y_i^{\mathbf{A}} - y_i^{\mathbf{B}}) + q_4(z_i^{\mathbf{A}} - z_i^{\mathbf{B}})]^2 + [q_1(x_i^{\mathbf{A}} - x_i^{\mathbf{B}}) + q_3(z_i^{\mathbf{A}} + z_i^{\mathbf{B}}) - q_4(y_i^{\mathbf{A}} - y_i^{\mathbf{B}})]^2 \} \end{aligned}$$

$$+[q_1(y_i^A - y_i^B) + q_4(x_i^A - x_i^B) - q_2(z_i^A - z_i^B)]^2 + [q_1(z_i^A - z_i^B) + q_2(y_i^A - y_i^B) - q_3(x_i^A - x_i^B)]^2\}. \quad (3.9)$$

Now, if  $q_1, q_2, q_3, q_4$  can be calculated from Equation (3.9), the quaternion for the rotation is obtained. Values of  $q_1, q_2, q_3, q_4$  that minimize  $\varepsilon$  such that  $q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$  are sought. Lagrange multipliers can be used to solve this problem given the following function from Equation (3.9):

$$\begin{aligned} \varepsilon = F(q_1, q_2, q_3, q_4, \lambda) = & \sum_{i=1}^N \{ [q_2(x_i^A - x_i^B) + q_3(y_i^A - y_i^B) + q_4(z_i^A - z_i^B)]^2 \\ & + [q_1(x_i^A - x_i^B) + q_3(z_i^A - z_i^B) - q_4(y_i^A - y_i^B)]^2 + [q_1(y_i^A - y_i^B) + q_4(x_i^A - x_i^B) - q_2(z_i^A - z_i^B)]^2 \\ & + [q_1(z_i^A - z_i^B) + q_2(y_i^A - y_i^B) - q_3(x_i^A - x_i^B)]^2 \} + \lambda(q_1^2 + q_2^2 + q_3^2 + q_4^2 - 1). \end{aligned} \quad (3.10)$$

Computing the partial derivatives of the unknowns of  $F(q_1, q_2, q_3, q_4, \lambda)$  and setting them equal to zero gives the following system of 5 equations containing 5 unknowns:

$$\left\{ \begin{array}{l} \frac{\partial F(q_1, q_2, q_3, q_4, \lambda)}{\partial q_1} = 0 \\ \frac{\partial F(q_1, q_2, q_3, q_4, \lambda)}{\partial q_2} = 0 \\ \frac{\partial F(q_1, q_2, q_3, q_4, \lambda)}{\partial q_3} = 0 \\ \frac{\partial F(q_1, q_2, q_3, q_4, \lambda)}{\partial q_4} = 0 \\ \frac{\partial F(q_1, q_2, q_3, q_4, \lambda)}{\partial \lambda} = 0 \end{array} \right. . \quad (3.11)$$

Letting  $x_i^m = x_i^A - x_i^B$ ,  $x_i^p = x_i^A + x_i^B$ ,  $y_i^m = y_i^A - y_i^B$ ,  $y_i^p = y_i^A + y_i^B$ ,  $z_i^m = z_i^A - z_i^B$ ,  $z_i^p = z_i^A + z_i^B$ , the problem can be organized as the following eigenvalue problem [21]:



$$\begin{pmatrix}
\sum_{i=1}^N (x_i^m)^2 + (y_i^m)^2 + (z_i^m)^2 & \sum_{i=1}^N y_i^p z_i^m - y_i^m z_i^p & \sum_{i=1}^N x_i^m z_i^p - x_i^p z_i^m & \sum_{i=1}^N x_i^p y_i^m - x_i^m y_i^p \\
\sum_{i=1}^N y_i^p z_i^m - y_i^m z_i^p & \sum_{i=1}^N (y_i^p)^2 + (z_i^p)^2 + (x_i^m)^2 & \sum_{i=1}^N x_i^m y_i^m - x_i^p y_i^p & \sum_{i=1}^N x_i^m z_i^m - x_i^p z_i^p \\
\sum_{i=1}^N x_i^m z_i^p - x_i^p z_i^m & \sum_{i=1}^N x_i^m y_i^m - x_i^p y_i^p & \sum_{i=1}^N (x_i^p)^2 + (z_i^p)^2 + (y_i^m)^2 & \sum_{i=1}^N y_i^m z_i^m - y_i^p z_i^p \\
\sum_{i=1}^N x_i^p y_i^m - x_i^m y_i^p & \sum_{i=1}^N x_i^m z_i^m - x_i^p z_i^p & \sum_{i=1}^N y_i^m z_i^m - y_i^p z_i^p & \sum_{i=1}^N (x_i^p)^2 + (y_i^p)^2 + (z_i^m)^2
\end{pmatrix}$$

$$* \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix} = \lambda \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix}. \quad (3.12)$$

One of the solutions for this system of equations will give us the values of  $q_1, q_2, q_3, q_4$  and  $\lambda$  as required. Thus, we can obtain the rotation that leads to the best superimposition of the two molecules, and the minimum value of  $\varepsilon$ , which is also the value that describes the difference between the two molecules geometrically. The rotation matrix for the best superimposition of molecules  $A$  and  $B$  is defined as follows [22]:

$$\begin{pmatrix}
q_1^2 + q_2^2 - q_3^2 - q_4^2 & 2(q_2 q_3 + q_1 q_4) & 2(q_2 q_4 - q_1 q_3) \\
2(q_2 q_3 - q_1 q_4) & q_1^2 + q_3^2 - q_2^2 - q_4^2 & 2(q_3 q_4 + q_1 q_2) \\
2(q_2 q_4 - q_1 q_3) & 2(q_3 q_4 - q_1 q_2) & q_1^2 + q_4^2 - q_2^2 - q_3^2
\end{pmatrix}. \quad (3.13)$$

### 3.2.3 Summary

By applying the above algorithm to any two molecules, a value measuring the similarity between the two molecules will be obtained. This algorithm can also be applied repeatedly to search a database of molecules for the molecule most similar to a query molecule by finding the minimum similarity score among all the molecule comparisons.

## Chapter 4

# Qualitative performance of the algorithm

### 4.1 Objectives

In this section, some examples demonstrating the performance of the algorithm will be discussed. The series of tests and the data sets to which they were applied will be presented. The main objectives of these tests are: 1. test if the algorithm can find the correct mapping between a query molecule and a target molecule (The correct mapping of an atom in a query molecule is an atom in a target molecule that is closest to the atom in the query molecule when the RMSD of the two molecules is minimized.); 2. test if the algorithm can find similar molecules for a query molecule from a collection of molecules based on the similarity score defined in Section 3.2.1; 3. test the algorithm's performance on different sets of molecules: a small set of molecules provided by Dr. Raymond Poirier of the Chemistry Department of Memorial University and a much larger data set downloaded from the National Cancer Institute (NCI) database [23] and preprocessed by Mark Staveley to compute atom types [13].

## 4.2 Application

In order to test the performance of the algorithm, a java-based application was implemented. The application is able to read structural files and display them in 3D. It can read two molecules in a standard file format and compute the similarity between them using the algorithm presented in Chapter 3. It will output the similarity score and the root mean square deviation (RMSD in Equation (2.7)) of the pair of molecules. Then, it will overlap 3D images of the two molecules based on their best superimposition calculated by the algorithm. The application can also search for the best match to a query molecule from many different molecules and provide a ranked list of all possible matches and similarity scores.

The application is implemented as an extension of Jmol [14]. Jmol is a free open source application that can display molecules in 3D. In Jmol the user can rotate the molecules, do many calculations on the molecules such as calculate distances, angles and torsion angles, and make use of many tools/functions that will not be discussed here. Almost all chemists are familiar with Jmol, which is one reason for using it as a platform. The application uses some of Jmol’s code as a foundation, such as its code for rendering, finding rotations and animation. Therefore, the user can also use Jmol’s scripts directly within the application.

It is worth mentioning that when implementing the algorithm to compare the atom triplets (as introduced in Section 3.2.2.1), a residual value is used to check if the distances are equal. As long as the difference between the two distances being compared is less than this residual value, the application considers these two distances to be equal. The Cartesian coordinates in a structural file may differ even for the same structure depending on the method(s) use to create it. Due to the round-off error in computers, even two structural files that describe the same molecule may vary several angstroms in the distance between two atoms. In the application developed

here, 0.01 Angstrom was arbitrarily chosen as the tolerance. If one wants to test the algorithm on a different data set, a different tolerance value might be more appropriate. All of the tests presented henceforth are run on a personal computer with an AMD Athlon™ dual core processor 4850e 2.5GHz CPU, 4 GB RAM and the 64-bit Windows 7 operating system.

### 4.3 Data sets

In order to check if the algorithm works as desired, tests were performed with two data sets. First, the algorithm was tested on a small set of molecules provided by Dr. Raymond Poirier of the Chemistry Department of Memorial University. There are 737 structures in the data set originating from a number of different sources [24–26]. The number of atoms in each structure in this data set ranges from 17 to 27. The data set consists of tens of molecules and their isomers, along with several transition states. Hence, every molecule in this data set has several similar molecules within the set. This data set is henceforth referred to as the 737-data-set and was used to test the functioning and performance of the algorithm. The algorithm was subsequently tested with a much larger data set downloaded from the National Cancer Institute (NCI) database [23] and preprocessed by Mark Staveley to compute atom types [13]. The version of the database that Dr. Staveley downloaded and processed was: Release 3 Files - September 2003. The NCI database contains 260,071 structures, combined from DTP (Developmental Therapeutics Program) releases from Oct. 1999, Aug. 2000, Feb. 2003, and Sep. 2003. Some molecules in the NCI database are described in 2D, as projections of the actual 3D structure. Although the algorithm described in this thesis is capable of comparing such 2D structures, they were omitted as test cases since their spatial extent is not fully described, making superimposition

of structures less meaningful. Following the removal of all unsuitable files, 163,652 structures remained. These 163,652 molecules are henceforth referred to as the sub-NCI database. The smallest molecule in this data set has 4 atoms while the largest has 270 atoms. The distribution of the structure size of this data set (in terms of number of atoms) is presented in Figure 4.1.

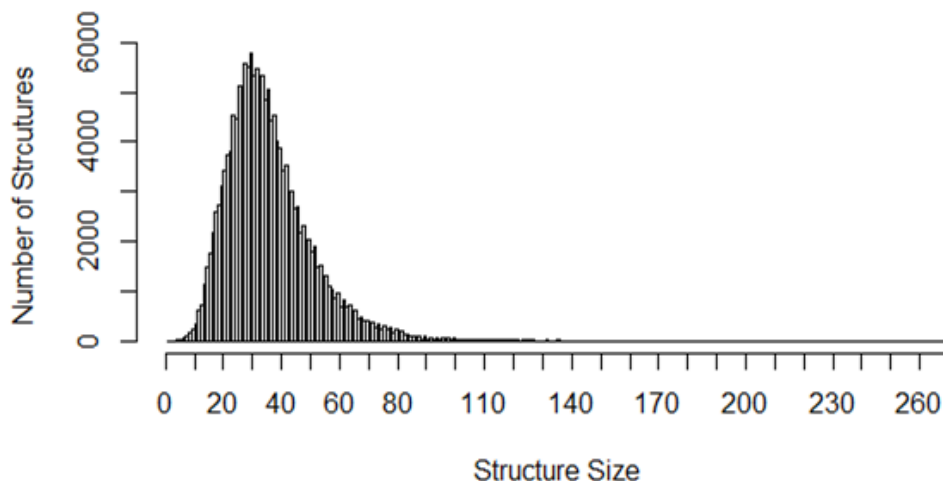


Figure 4.1: The distribution of size (number of atoms) of the test molecules in the sub-NCI database.

In the following section, the results of testing the algorithm functionality and performance on these two data sets are presented and discussed.

### 4.3.1 Test results

#### 4.3.1.1 Finding similar molecules

In order to test the performance of the algorithm with similar molecules, each molecule of the 737-data-set is used as a query to search all of the molecules in the 737-data-set for similar molecules. The query molecule was kept in the target molecules as a distinct benchmark of the test, since the algorithm should always find the query molecule itself as the most similar molecule. The search result of every molecule in the

737-data-set is in the folder named “Result of testing with the 737-data-set” in the zip file attached to this thesis. For all of the molecules in the 737-data-set, each of them found itself to be the most similar one. Owing to the length of the thesis, among all the search results, only the six most similar molecules of five query molecules that the algorithm found are listed in Table 4.1. The details of the comparison results (output of the application) shown in Table 4.1 are listed in Appendix A. The first column of the table contains the query molecules. The other columns are the top six most similar molecules (in ranked order) the algorithm found according to the calculated similarity score in Equation (3.1). For each molecule, its 3D image, its molecular formula and its structural filename are listed. It can be seen that for each of the five molecules in the table, the algorithm found the same molecule as the query molecule to be the most similar one. Although for molecule  $C_5H_{12}N_2O_4$  (in the third row of Table 4.1), the most similar molecule’s filename (in the third row second column of Table 4.1) is not the same as the query molecule’s filename, while the same structural filename is listed as the second most similar one (in the third row third column of Table 4.1). These two files describe the same molecule in the 737-data-set and the same similarity score ( $8.656624 \times 10^{-8}$ ) and RMSD ( $8.656624 \times 10^{-8}$ ) is found for each. For some structures in the 737-data-set, there are several structural files describing the same molecule that have been computed using different methods. The ranking for these kind of structural files with the same similarity score is based on the comparison order of structures. For these cases, in the result list, the query structural file itself may not be in first place. However, this does not affect the fact that the most similar molecule found by the algorithm is still the query molecule itself. As illustrated by the images in Table 4.1, the other top ranked molecules found by the algorithm, are all similar to the query molecules. Thus, the algorithm performs as desired for the 737-data-set.

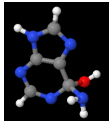
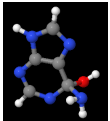
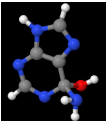
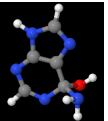
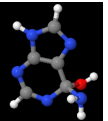
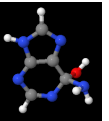
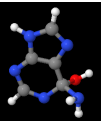
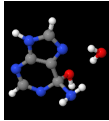
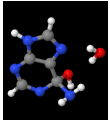
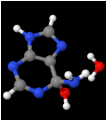
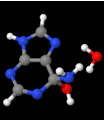
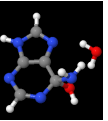
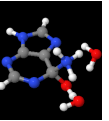
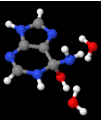
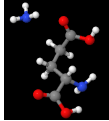
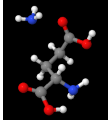
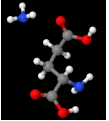
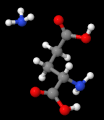
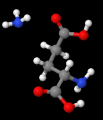
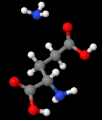
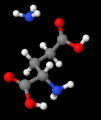
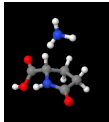
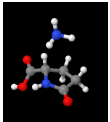
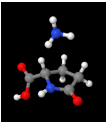
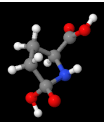
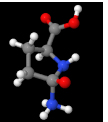
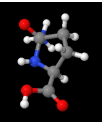
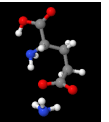
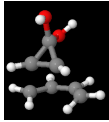
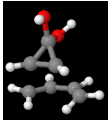
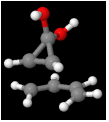
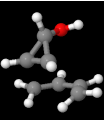
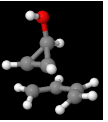
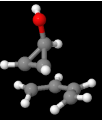
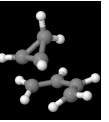
Query	NO. 1	NO. 2	NO. 3	NO. 4	NO. 5	NO. 6
 $C_5H_6N_5O$ (GS_Ade_ PathwayA_ I1A_B3LYP_ 631Gd)	 $C_5H_6N_5O$ (GS_Ade_ PathwayA_ I1A_B3LYP_ 631Gd)	 $C_5H_6N_5O$ (GS_Ade_ PathwayA_ I1A_G3MP2B3)	 $C_5H_6N_5O$ (GS_Ade_ PathwayA_ I1A_G3B3)	 $C_5H_6N_5O$ (GS_Ade_ PathwayA_ I1A_CBSQB3)	 $C_5H_6N_5O$ (GS_Ade_ PathwayA_ I1A_MP2_ 631Gd)	 $C_5H_6N_5O$ (GS_Ade_ PathwayA_ I1A_B3LYP_ 631pGd)
 $C_5H_8N_5O_2$ (GS_Ade_ PathwayB_ I1B_B3LYP_ 631pGd)	 $C_5H_8N_5O_2$ (GS_Ade_ PathwayB_ I1B_B3LYP_ 631pGd)	 $C_5H_8N_5O_2$ (GS_Ade_ PathwayB_ I1B_HF_ 631Gd)	 $C_5H_8N_5O_2$ (GS_Ade_ PathwayB_ I1B_MP2_ 631Gd)	 $C_5H_8N_5O_2$ (GS_Ade_ PathwayB_ I1B_CBSQB3)	 $C_5H_{10}N_5O_3$ (TS_Ade_ PathwayC_ TS2C_ G3MP2B3)	 $C_5H_{11}N_5O_3$ (GS_Ade_ PathwayF_ I2F_HF_ 631Gd)
 $C_5H_{12}N_2O_4$ (GS_DHA_P1_ B3LYP_631Gd)	 $C_5H_{12}N_2O_4$ (GS_DHB_P1_ B3LYP_631Gd)	 $C_5H_{12}N_2O_4$ (GS_DHA_P1_ B3LYP_631Gd)	 $C_5H_{12}N_2O_4$ (GS_DHA_P1N_ G3MP2B3)	 $C_5H_{12}N_2O_4$ (GS_DHB_P1_ G3MP2B3)	 $C_5H_{12}N_2O_4$ (TS_IRCF_P_ DHA_TS1_ B3LYP_631Gd)	 $C_5H_{12}N_2O_4$ (TS_IRCF_P_ DHB_TS2_ B3LYP_631Gd)
 $C_5H_{10}N_2O_3$ (GS_GDA_P1b_ B3LYP_ 631pGdp_SMD)	 $C_5H_{10}N_2O_3$ (GS_GDA_P1b_ B3LYP_ 631pGdp_SMD)	 $C_5H_{10}N_2O_3$ (GS_GDA_P1_ B3LYP_ 631pGdp_PCM)	 $C_5H_8NO_4$ (TS_IRCF_ I2b_PHC_TS3_ B3LYP_ 631pGdp)	 $C_5H_{10}N_2O_3$ (GS_GDA_P1_ B3LYP_ 631pGdp_SMD)	 $C_5H_{10}N_2O_3$ (TS_IRCP_P_ DGA_TS1_ B3LYP_ 631pGdp_SMD)	 $C_5H_{11}N_2O_4$ (GS_DHD_P1_ B3LYP_631Gd)
 $C_7H_{10}O_2$ (TS_C7H1002_ endo3B_ HF631ppGd)	 $C_7H_{10}O_2$ (TS_C7H1002_ endo3B_ HF631ppGd)	 $C_7H_{10}O_2$ (TS_C7H1002_ endo_3B_ B3LYP631ppGd)	 $C_7H_{10}O$ (TS_C7H100_ endo_syn_G_ HF631ppGd)	 $C_7H_{10}O$ (TS_C7H100_ endo_anti_S_ HF631ppGd)	 $C_7H_{10}O$ (TS_C7H100_ endo_anti_S_ HF631Gd)	 $C_7H_{10}$ (TS_ C7H10_endo_ HF631ppGd)

Table 4.1: Top six similar molecules for five sample query structures searching the 737-data-set.

#### 4.3.1.2 Changes in orientation and labeling order

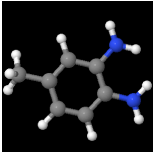
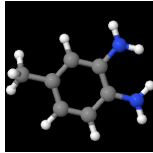
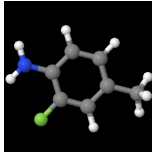
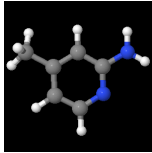
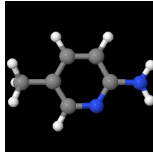
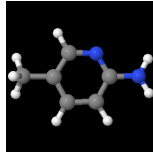
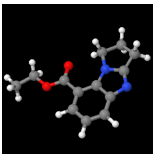
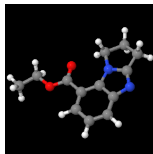
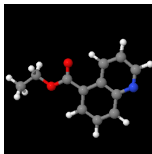
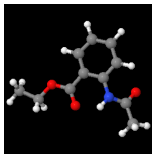
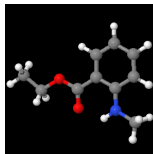
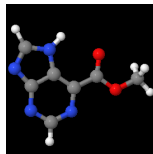
In order to test whether the algorithm can find similar molecules to a query molecule in spite of the orientation in the structural files, a new data set was generated by randomly rotating each molecule in the 737-data-set to a different orientation. The labeling order of each structural file does not need to be changed randomly, because the algorithm pairs the atoms automatically regardless of the labeling order. This new data set is referred to as the 737r-data-set. Each molecule of the 737-data-set is used as a query to search all the molecules in the 737r-data-set for similar molecules. The details of the comparison results (output of the application) of the same molecules as in Table 4.1 are listed in Appendix B. Although the ranking order of some structures has changed (Appendix B), all of the same structural files were in the top ranked list. The search results of using every molecule in the 737-data-set to search all the molecules in the 737r-data-set are in the folder named “Result of testing with the 737r-data-set” in the zip file attached to this thesis. When randomly rotating a molecule, minor roundoff errors are introduced into the Cartesian coordinates, thus, the similarity scores of these structures were changed, leading to a change of the ranking order of these files. Regardless of which filename was listed first, the molecule that these files represent is still the same. Therefore, using the algorithm on the randomly perturbed structures produced the same results as listed in Table 4.1, demonstrating that the algorithm could find the best superimposition even though the molecules are not in the same orientation and do not have the same labeling order.

#### 4.3.1.3 Searching the sub-NCI database

Due to the size of the sub-NCI data set, searching every molecule in the whole database for its similar molecules will be quite time consuming. Thus, a random selection of a number of molecules were chosen as query structures. According to the distribution



of the molecule size in Figure 4.1, most molecules have 20 – 80 atoms. Therefore, the molecules were categorized into 5 groups: less than 20 atoms, 21 – 40 atoms, 41 – 60 atoms, 61 – 80 atoms and more than 80 atoms. Then, 5 molecules from each category were randomly chosen and used to compare to all the molecules in the sub-NCI database. The query molecules were still kept in the target molecules as a benchmark of the test. For all the structures being tested, the algorithm found the same molecule as the query molecule to be the most similar. Other similar molecules were molecules with the same geometric structure but different atoms, or molecules with similar geometric structure. Five of the test results are listed in Table 4.2. The detailed results (output of the application) of each molecule in Table 4.2 are listed in Appendix C. The complete search results of these molecules are in the folder named “Result of testing with the sub-NCI database” in the zip file attached to this thesis. Based on the results of these randomly selected structures, the algorithm proved successful in finding similar structures in the sub-NCI database.

Query	NO. 1	NO. 2	NO. 3	NO. 4	NO. 5
 4-methyl-1,2-benzenediamine $C_7H_{10}N_2$ (1487_NCI)	 4-methyl-1,2-benzenediamine $C_7H_{10}N_2$ (1487_NCI)	 2-fluoro-4-methylaniline $C_7H_8NF$ (111019_NCI)	 4-methyl-2-pyridinamine $C_6H_8N_2$ (1482_NCI)	 5-methyl-2-pyridinamine $C_6H_8N_2$ (1481_NCI)	 5-methyl-2-pyridinamine $C_6H_8N_2$ (76019_NCI)
 ethyl 2,3-dihydro-1H-pyrrolo[1,2-a]benzimidazole-8-carboxylate $C_{13}H_{14}N_2O_2$ (108547_NCI)	 ethyl 2,3-dihydro-1H-pyrrolo[1,2-a]benzimidazole-8-carboxylate $C_{13}H_{14}N_2O_2$ (108547_NCI)	 ethyl 5-quinolinecarboxylate $C_{12}H_{11}NO_2$ (105780_NCI)	 ethyl 2-(acetylamino)benzoate $C_{11}H_{13}NO_3$ (99250_NCI)	 ethyl 2-(methylamino)benzoate $C_{10}H_{13}NO_2$ (199847_NCI)	 methyl 9H-purine-6-carboxylate $C_7H_6N_4O_2$ (205011_NCI)

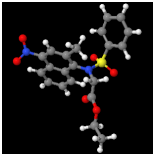
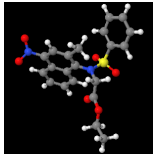
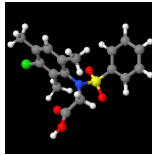
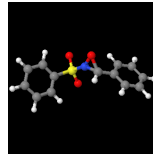
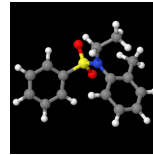
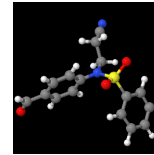
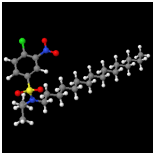
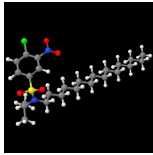
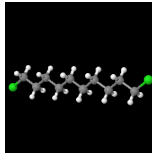
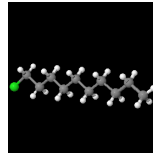
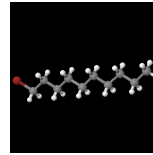
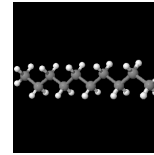
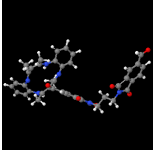
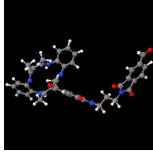
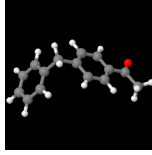
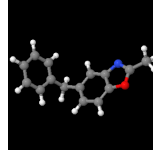
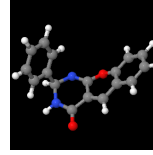
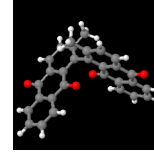
 <p>ethyl ((4-(hydroxy(oxido)amino)-2-methyl-1-naphthyl)(phenylsulfonyl)amino)acetate  <math>C_{21}H_{20}N_2O_6S</math>  (118972_NCI)</p>	 <p>ethyl ((4-(hydroxy(oxido)amino)-2-methyl-1-naphthyl)(phenylsulfonyl)amino)acetate  <math>C_{21}H_{20}N_2O_6S</math>  (118972_NCI)</p>	 <p>(3-chloro-2,4,6-trimethyl(phenylsulfonyl)anilino)acetic acid  <math>C_{17}H_{18}NO_4SCl</math>  (116735_NCI)</p>	 <p>3-phenyl-2-(phenylsulfonyl)-1,2-oxaziridine  <math>C_{13}H_{11}NO_3S</math>  (190006_NCI)</p>	 <p>N-ethyl-N-(2-methylphenyl)benzenesulfonamide  <math>C_{15}H_{17}NO_2S</math>  (122069_NCI)</p>	 <p>N-(2-cyanoethyl)-N-(4-formylphenyl)benzenesulfonamide  <math>C_{16}H_{14}N_2O_3S</math>  (222312_NCI)</p>
 <p>4-chloro-N-hexadecyl-3-(hydroxy(oxido)amino)-N-isopropylbenzenesulfonamide  <math>C_{25}H_{43}N_2O_4SCl</math>  (113567_NCI)</p>	 <p>4-chloro-N-hexadecyl-3-(hydroxy(oxido)amino)-N-isopropylbenzenesulfonamide  <math>C_{25}H_{43}N_2O_4SCl</math>  (113567_NCI)</p>	 <p>1,10-dichlorodecane  <math>C_{10}H_{20}Cl_2</math>  (9324_NCI)</p>	 <p>1-chlorodecane  <math>C_{10}H_{21}Cl</math>  (6004_NCI)</p>	 <p>1-bromodecane  <math>C_{10}H_{21}Br</math>  (8638_NCI)</p>	 <p>decane  <math>C_{10}H_{22}</math>  (8639_NCI)</p>
 <p>4,4-bis(4-(((4-(phenyldiazenyl)anilino)carbonyl)oxy)phenyl)pentanoic acid  <math>C_{43}H_{36}N_6O_6</math>  (206308_NCI)</p>	 <p>4,4-bis(4-(((4-(phenyldiazenyl)anilino)carbonyl)oxy)phenyl)pentanoic acid  <math>C_{43}H_{36}N_6O_6</math>  (206308_NCI)</p>	 <p>1-(4-benzylphenyl)ethanone  <math>C_{15}H_{14}O</math>  (69547_NCI)</p>	 <p>5-benzyl-2-methyl-1,3-benzoxazole  <math>C_{15}H_{13}NO</math>  (92804_NCI)</p>	 <p>2-phenyl-2,3-dihydro-4H-chromeno[2,3-d]pyrimidin-4-one  <math>C_{17}H_{12}N_2O_2</math>  (157385_NCI)</p>	 <p>Dichinyl  <math>C_{30}H_{18}O_4</math>  (7126_NCI)</p>

Table 4.2: Top five similar molecules for five sample query structures searching the sub-NCI data set.

#### 4.3.1.4 Test conclusion

Based on the test results presented, the algorithm is shown to be functional and effective in comparing two molecules as well as searching a database for similar molecules.

### 4.3.2 Some Features of the algorithm

Some concrete examples are presented in order to give a vivid image of how the algorithm works. Here is an example discovered when testing the algorithm with the sub-NCI database. The molecule on the left in Figure 4.2 (2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ )) was used to search the whole sub-NCI database for the most similar molecule. The molecule on the right (2-hydroxy-5-methylbenzo-1,4-quinone ( $C_7H_6O_3$ )) is the second most similar one found by the algorithm. The most similar one is the same molecule as the query but in a different orientation.

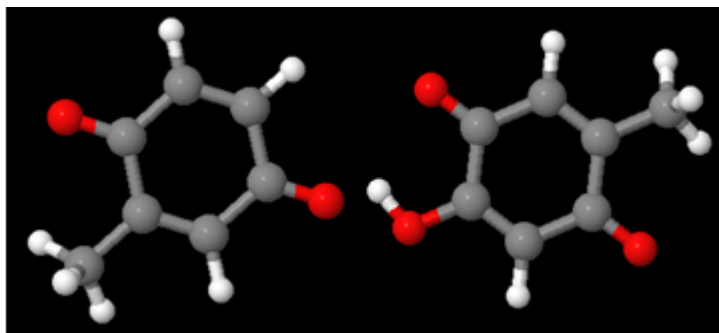


Figure 4.2: Two similar molecules in the NCI database, the left one is 2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ ) and the right one is 2-hydroxy-5-methylbenzo-1,4-quinone ( $C_7H_6O_3$ ).

Figure 4.3 shows the labeling order of the atoms in the structural files of these two molecules.

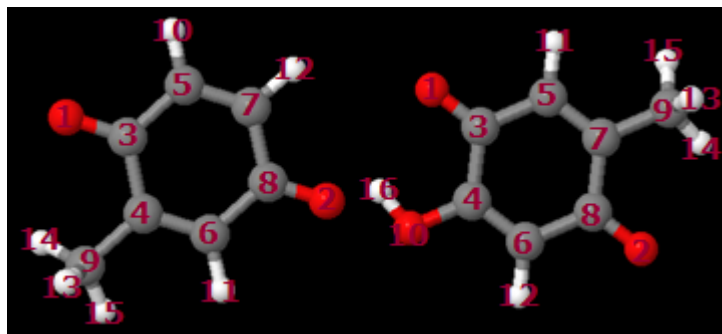


Figure 4.3: The labeling order in the structural files of 2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ ) (left) and 2-hydroxy-5-methylbenzo-1,4-quinone ( $C_7H_6O_3$ ) (right) in the NCI database.

It can be seen that the labeling order in these two structural files does not correspond to the correct mapping of the atoms. Without this thesis work, if these two structural files are used directly to find the best superimposition, the result is not the maximum overlap of these two structures. Figure 4.4 demonstrates the best superimposition Jmol obtained based on the labeling order in the two structural files. The RMSD for this superimposition was 2.09 Angstroms. It is obvious that these two molecules can be overlapped in a better position than the one in Figure 4.4.

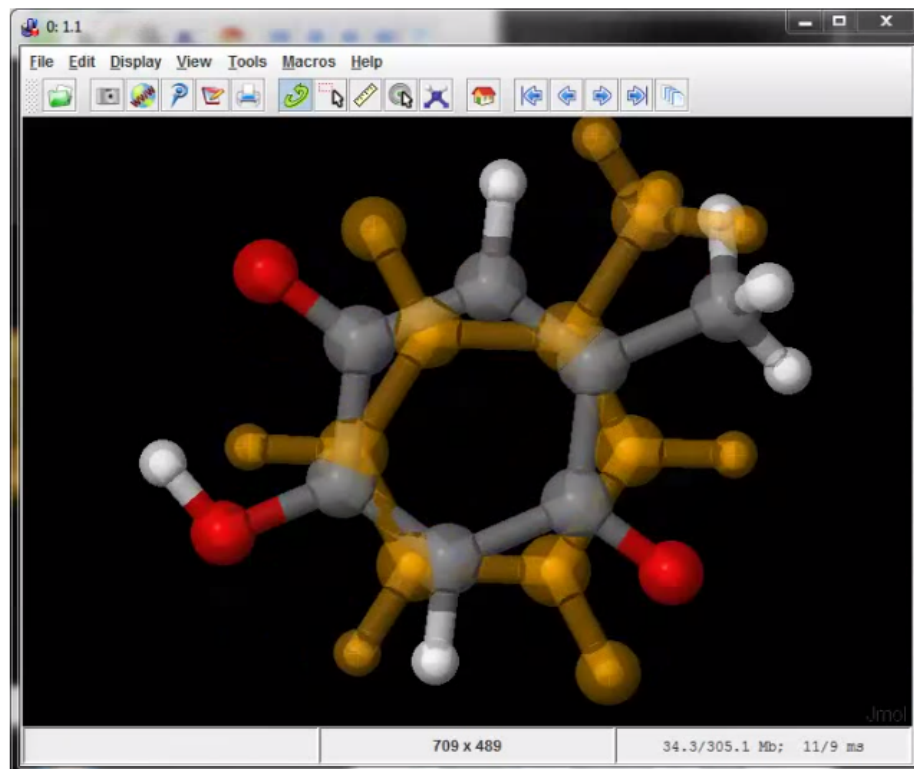


Figure 4.4: Comparison result in Jmol based on the structural files of 2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ ) and 2-hydroxy-5-methylbenzo-1,4-quinone ( $C_7H_6O_3$ ) (in transparent yellow) in the NCI database.

Using the algorithm developed for the thesis, the result shown in Figure 4.5 is obtained, where molecule 2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ ) is able to fully overlap with molecule 2-hydroxy-5-methylbenzo-1,4-quinone ( $C_7H_6O_3$ ).

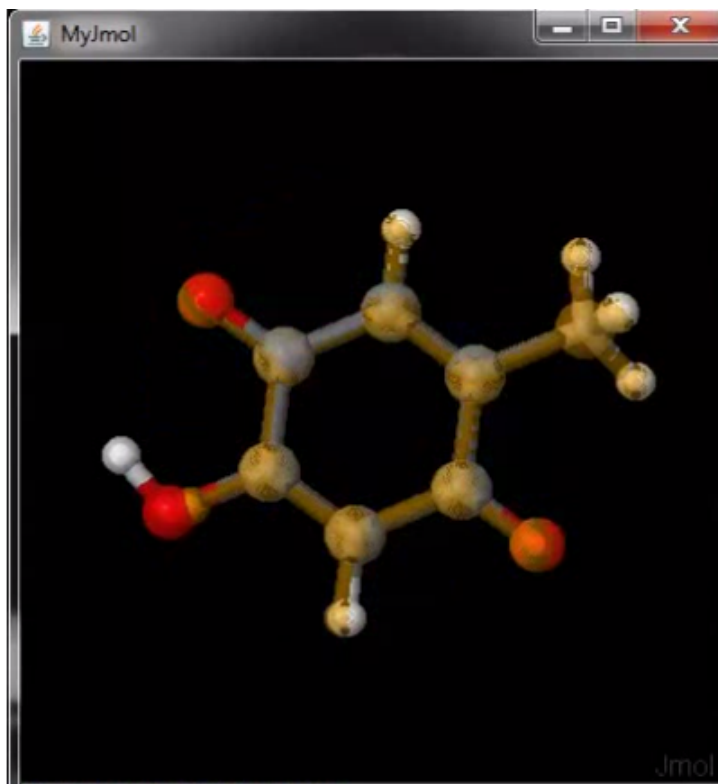
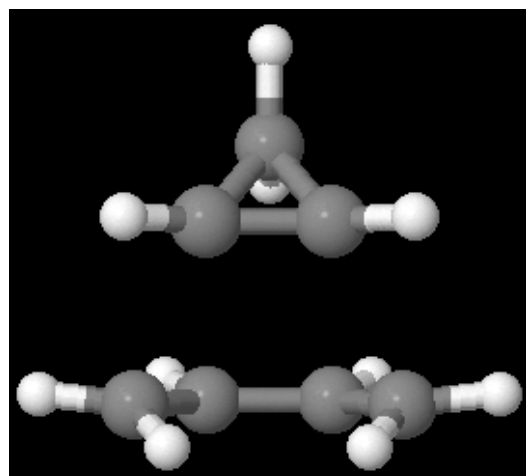


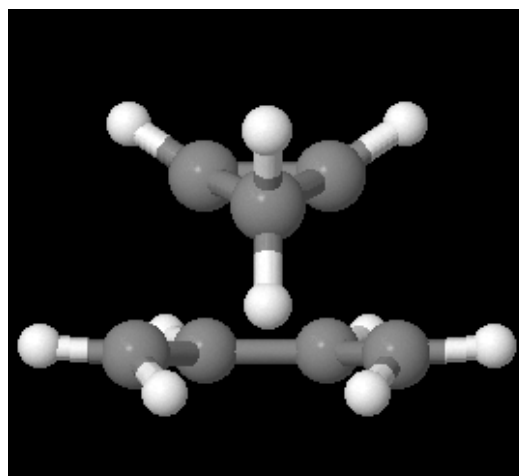
Figure 4.5: Comparison result of the algorithm based on the structural files of 2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ ) and 2-hydroxy-5-methylbenzo-1,4-quinone ( $C_7H_6O_3$ ) (in transparent yellow) in the NCI database.

It is important to note that, based on the algorithm, the best superimposition calculated may not make the largest substructure of the two molecules fully overlap as in the algorithm in [8]. This is because the algorithm is trying to find the best superimposition of the two molecules based on the minimum value of the RMSD between the corresponding atoms. For some cases, there may be large substructures that are identical in the two molecules to be compared, but in order to make the distances between the atoms outside of the substructures close enough, the identical substructures will not fully overlap. To illustrate this phenomenon, molecule  $C_7H_{10}$  and its isomer are chosen from the 737-data-set (shown in Figure 4.6).

(a) Molecule  $C_7H_{10}$ 

(filename:

TS\_C7H10\_endo\_B3LYP631ppGd.log1.ar)

(b) An isomer of molecule  $C_7H_{10}$ 

(filename:

TS\_C7H10\_exo\_B3LYP631ppGd.log1.ar)

Figure 4.6: Molecule  $C_7H_{10}$  and its isomer.

These two molecules contain the same two components. The top component consists of 7 atoms and the bottom component consists of 10 atoms. Each component is identical to its counterpart in the other structure, however, one component is inverted relative to the other. When applying the algorithm to these two molecules, the best superimposition obtained is shown in Figure 4.7.

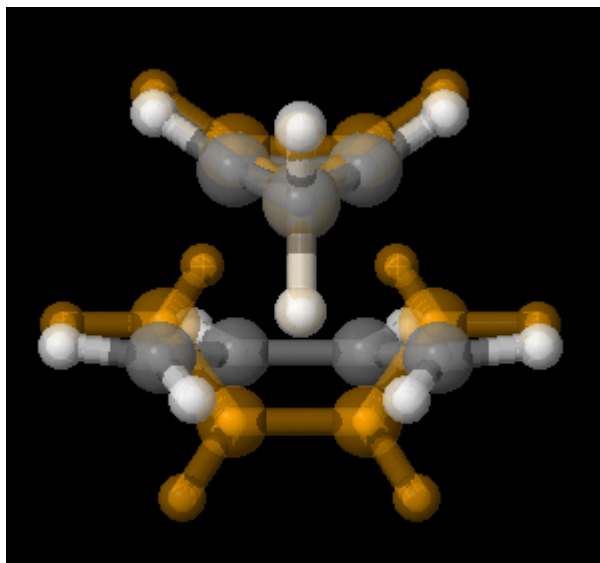


Figure 4.7: Comparison result of  $C_7H_{10}$  and one of its isomers (in yellow).

If the largest substructures (bottom component) fully overlap, the distances between the atoms in the top component will make the total RMSD larger than if the top component was fully superimposed. In fact, these kinds of cases do not only happen with the algorithm presented here. As long as the superimposing method uses the RMSD to measure the best superimposition, such cases may occur.



# Chapter 5

## Run-time analysis

In this chapter, the run-time analysis of the algorithm is presented to determine for a particular pair of structures, how long the algorithm would take to compute the similarity score, as well as to determine which factors may affect the run-time of the algorithm. First, the time complexity analysis of the algorithm is introduced. Second, test data is used to analyze the run-time. Linear regression is used to analyze these data in order to obtain the factors that affect the run-time performance of the algorithm. Finally, some concrete examples of the run-time of the algorithm to search the sub-NCI database are provided.

### 5.1 Time complexity analysis of the algorithm

#### 5.1.1 Time complexity

Time complexity is widely used to describe the run-time of an algorithm in computer science. Often a function ( $f(a)$ ) is used to represent the number of steps that an algorithm uses on any input of length  $a$  [28]. There are commonly two ways to analyze the time complexity: worst-case analysis and average-case analysis. In worst-

case analysis, the longest running time for input of a particular length is considered. In average-case analysis, the average of the running times for inputs of a particular length is considered. Sometimes it is difficult to obtain the function  $f(a)$  representing the worst-case or average-case of an algorithm. Instead, a function  $g(a)$  can be obtained, where  $f(a) < c * g(a)$  for suitably large  $a$  and constant  $c$  [28]. This is called the upper bound analysis.

The exact running time of an algorithm may be a complex expression. Since the main interest in understanding the running time of the algorithm is typically when it is run on large inputs, one convenient form of estimation, called asymptotic analysis, is often used to describe the running time of an algorithm. Only the highest order term of the expression for the running time of the algorithm need be considered. Both the coefficient of that term and any lower order terms are disregarded, because the highest order term dominates the other terms on large inputs [28].

For example, the function  $f(a) = 6a^3 + 7a^2 + 15a + 99$  describing the run-time of an algorithm has four terms and the highest order term is  $6a^3$ . Disregarding the coefficient 6, the time complexity of this algorithm is  $O(a^3)$ . This notation is called big-O notation ( $O()$ ) and it is the most common way to represent upper bounds. Big-O notation represents the idea that one function is asymptotically no more than another. If the worst-case time complexity of an algorithm is  $O(a^3)$ , then the run-time of the worst case of this algorithm can be represented by a polynomial with a highest order term of  $a^3$ .

### 5.1.2 Time complexity calculation

In this section, the time complexity of the algorithm presented in this thesis is analyzed step-by-step based on the flow chart of the algorithm in Figure 3.1. If the probability of every case of an algorithm can be obtained, an average-case analysis of time complexity

can be calculated. Unfortunately, every molecule is distinct. There are no patterns or rules that describe a distribution of molecules, from which an average can be determined. Thus, the worst-case analysis is applied here. As introduced in the last section, time complexity is a function ( $f(a)$ ) representing the maximum number of steps that an algorithm uses on any input of length  $a$  [28].

The input of the program is two structure files representing two molecules. Accordingly, the parameter of the function  $f()$  should be the length  $a$  of these two structure files. In Appendix D, an example of the structure file that the algorithm of this thesis uses as input is shown. It can be seen that the file contains two main parts. Each line in the first part lists the atomic number, Cartesian coordinates, labeling order and atom type of an atom in the structure. The number of characters of the atomic number is 4, the number of characters of each coordinate of the Cartesian coordinates and the atom type are 11, and the number of characters of the labeling order is 3. The number of characters of the interval between every two fields is 3 except the one between the  $z$  coordinates and the labeling order is 2. Thus, the number of characters of first part is  $65 * (n + 1)$ , where  $n$  represents the number of atoms in the structure, and 1 represents the header line of the first part. The second part of the file lists the connectivity information of the atoms in the structure with the labeling of the atoms indicated in the first part. For every listed atom, two lines are used to indicate the connectivity information. The first line lists the labeling order of an atom and the labeling order of all the other atoms bonded to it. The number of characters of the labeling order of the listed atom is 5 and the number of characters of each of all the other atoms bonded to it is 4. The number of characters between every two fields is 1. The second line lists the atom type of all the atoms in the first line. The number of characters of the atom type of the listed atom is 13, the number of characters of the atom type of the bonded atoms is 14 and the number

of characters between every two fields is 1. Unfortunately, the number of characters of each line in the second part varies because different atoms may have a different number of bonded atoms. The actual number of characters of every line cannot be predicted. However, an atom can be bonded to at most 15 atoms [29]. Therefore, the upper bound can be used here. The upper bound of the number of characters of the first line in the line-triple that represents the connectivity information of an atom is  $5 + 15 * 4 + 7 = 72$ . The upper bound of the number of characters of the second line is  $13 + 14 * 15 + 7 = 230$ . The number of characters of the header of the second part is 39. Thus, the upper bound of the number of characters of the second part is  $72*n + 230*n + 39 = 302*n + 39$ . Therefore, the upper bound of the number of characters of a structure file is  $65*(n+1) + 302*n + 39 = 367*n + 104$ , where  $n$  is the number of atoms in the structure. The algorithm of this thesis takes two structure files as the input, thus, the length  $a$  of the input is  $367*n_1 + 104 + 367*n_2 + 104 = 367*(n_1 + n_2) + 208$ , where  $n_1$  is the number of atoms in molecule A and  $n_2$  is the number of atoms in molecule B. According to the usual expression of time complexity, this length  $a$  should be used as the function parameter. However, it can be seen that this length  $a$  is dependent on the number of atoms in the two molecules. Therefore, the number of atoms in the two molecules can be used as the input of the function representing the run-time of the algorithm. Since the upper bound analysis is applied here, a function  $g(n_1, n_2)$  rather than  $f(n_1, n_2)$  is sought in the following analysis (for ease of illustration,  $n_1$  and  $n_2$  are henceforth used to denote the the number of atoms in molecule A and B, respectively, where  $n_1 \leq n_2$ ), where  $f(n_1, n_2)$  is bounded by  $g(n_1, n_2)$ .

The algorithm actually starts in Step 2 of the flow chart in Figure 3.1 (Step 1 just indicates the start of the algorithm). Step 2 reads in structural information of two molecules that includes Cartesian coordinates, type of each atom, and connectivity information. As calculated in the last paragraph, the length of the input is  $367 *$

$(n_1 + n_2) + 208$ . Considering the reading in of a single character as a unit step, the algorithm takes  $367 * (n_1 + n_2) + 208$  steps to read in the two structure files. The algorithm needs to save the  $x, y, z$  coordinates, atom type and the connectivity information. The algorithm gets each of these values of an atom in a structure by getting the substrings of a line. Thus, the algorithm takes 8 steps (4 steps to get the  $x, y, z$  values and the atom type, and 4 steps to save these 4 values) to save the information of the  $x, y, z$  coordinates and the atom type of an atom, and at most 16 steps (1 step to get the information of the atoms, at most 7 steps to get the information of all the atoms bonded to it and 8 steps to save these values) to save the connectivity information. The algorithm takes at most  $24 * n_1 + 24 * n_2$  steps to save all the necessary information. Therefore, Step 2 of the algorithm takes at most  $367 * (n_1 + n_2) + 208 + 24 * n_1 + 24 * n_2 = 391(n_1 + n_2) + 208$  steps.

Step 3 finds all atom triplets in each molecule. For each atom triplet, three atoms have to be bonded in the molecule and cannot be co-linear in geometrical space. For an atom in a molecule, it chooses its two bonded atoms to form an atom triplet. The number of atom triplets in a molecule depends on its structure. A precise number cannot be calculated just based on the number of atoms in a molecule. However, the number of atoms bonded to an atom is not likely to exceed 4 in our data sets. In Figure 5.1, the  $x$  axis is the the number of atoms in the structures in the sub-NCI database and the  $y$  axis is the number of triplets the algorithm found in a structure. The red line represents the function  $y = 4x$ , which means for a certain structure with  $x$  atoms the number of the triples found in this structure is  $4x$ . It can be seen that all the dots in the plot are under this line. Thus, for all the structures in the sub-NCI database, the number of triplets actually found in each structure is no more than the four times the number of atoms in the structure.

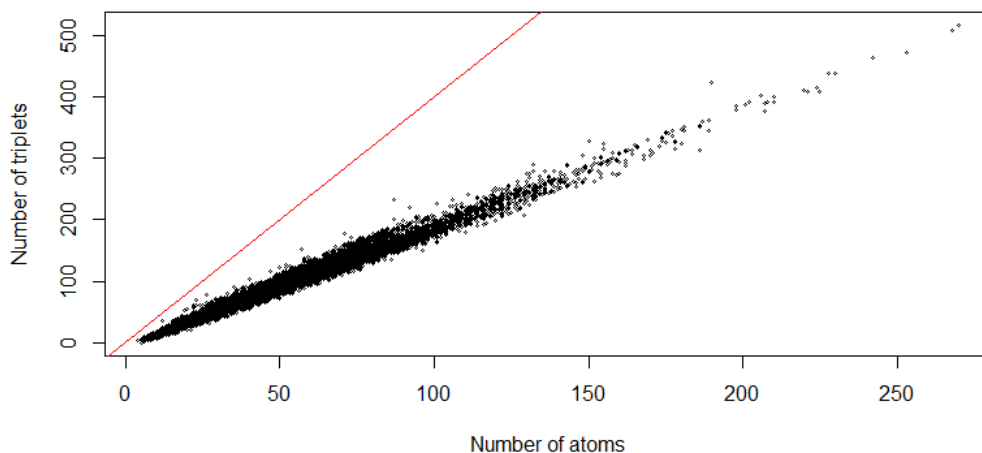


Figure 5.1: Number of atoms vs. the number of triplets found in the structure for all the structures in the sub-NCI database.

As a result, for a molecule with  $n$  atoms, it will have at most  $4 * n$  atom triplets. For every atom triplet, the algorithm has to check if the three atoms are co-linear. To do so, it calculates the distances between every two atoms, which requires 3 steps. If the length of the two short edges is equal to the length of the longest edge, the three atoms are co-linear. This requires at most 4 steps (2 steps to sort the edges, 2 steps to compare the length) to check if three atoms are co-linear. Then, the algorithm requires 3 steps to save the information of every atom and 3 steps to save the length of every edge. Consequently, the algorithm requires no more than  $4 * n + 4 * n * (3 + 4 + 3 + 3) = 56 * n$  steps to find and save all the atom triplets in a structure. Therefore, the algorithm requires at most  $56 * n_1 + 56 * n_2 = 56 * (n_1 + n_2)$  steps to find and save all the atom triplets in two molecules.

Step 4 initializes the variable used to save the sum of all the distances between the mapped atoms of two molecules, as well as the variable used to save the potential correct mapping of atoms between the two molecules, which requires 2 steps.

Step 5 compares every triplet from molecule A to every triplet from molecule B without replacement. The number of steps will be the number of atom triplets in molecule A multiplied by the number of atom triplets in molecule B. As calculated in Step 3, the upper bound of the number of atom triplets in a molecule is  $4*n$ . Thus, the upper bound of the steps to compare all the atom triplets is  $4*n_1*4*n_2 = 16*n_1*n_2$ .

Step 6 compares a triplet in molecule A to a triplet in molecule B to see if the atoms in the two triplets are mapped. If the atoms in the two triplets are mapped, continue to Step 7. Otherwise go to Step 13. Step 6 compares the atom types of the three atoms in two atom triplets, which requires 3 steps. It then compares the lengths between every two atoms in two atom triplets, which also requires 3 steps. Thus, the algorithm requires 6 steps to compare a triplet in molecule A to a triplet in molecule B.

Step 7 computes the mapping of each atom for matched triplets. The worst case happens when two atom triplets form two identical regular triangles with six identical atoms. In that case, every atom in one triplet can be mapped to any atom in the other triplet. This requires 6 steps to save all the mapping information. Thus, it requires at most 12 steps for one comparison of two atom triplets.

Step 8 computes the rotation that makes the atoms of the two triplets overlap. The system of equations in Equation (3.12) requires  $6 * 16 * N$  steps (6 steps to compute  $x_i^m, x_i^p, y_i^m, y_i^p, z_i^m, z_i^p$ , 16 steps to compute the  $4 \times 4$  matrix for one pair of points, where  $N$  represents the number of paired points) to construct, and  $4 * 4$  steps to solve. Here the number of paired points is 3. Thus, the algorithm requires  $6 * 16 * 3 + 16 = 304$  steps to compute the rotation.

Step 9 applies the computed rotation in Step 8 to molecule A, which means it applies the rotation matrix in Equation (3.13) to every atom in A. For ease of calculation, every entry in the matrix is considered to be computed in 1 step. Thus, it requires

9 steps to compute the rotation matrix and then it requires 15 steps to compute the coordinates after rotation. Thus, the algorithm requires  $9 + 15 * n_1$  steps to apply the computed rotation to molecule A. After that, the algorithm also needs to translate molecule A in order to overlap the centers of molecule A and B. It requires  $n_1 * 3$  steps to get the center of molecule A,  $n_2 * 3$  steps to get the center of molecule B, and 3 steps to get the amount by which the atoms in molecule A need to be translated. Then, the algorithm requires  $3 * n_1$  steps to apply this translation to molecule A. Thus, the algorithm requires  $9 + 15 * n_1 + 3 * n_1 + 3 * n_2 + 3 + 3 * n_1 = 15 + 21 * n_1 + 3 * n_2$  steps to rotate and translate molecule A.

Step 10 runs the Hungarian algorithm (Section 3.2.2.1) to get the best mapping of all the atoms in the two molecules for the current structural orientation. For a  $N \times N$  cost matrix, the Hungarian algorithm requires  $N$  steps to subtract the smallest entry in each row from all the entries of the row and  $N$  steps to subtract the smallest entry in each column from all the entries of the column. Then, it requires at most  $N^2$  steps to determine which rows and columns to exclude so that all the zero entries of the cost matrix are covered and the minimum number of such exclusions are used (ie. determine where to draw lines). If the minimum number of lines drawn is  $N$ , an optimal assignment of zeros is possible and the algorithm is finished. If the minimum number of lines is less than  $N$ , the algorithm determines the smallest entry not covered by any line. This entry is subtracted from each uncovered row, and then added to each covered column, which requires  $2 * N$  steps. The algorithm repeats this process until the optimal assignment is found. The worst case happens when every loop only makes one of the  $N$  rows all zero. In which case, the algorithm needs to be repeated  $N$  times. Thus, the Hungarian algorithm requires at most  $(N + N + N^2 + 2 * N) * N = N^3 + 4N^2$  steps. In the algorithm of this thesis,  $N = n_2$ , so at most  $n_2^3 + 4n_2^2$  steps are required to get the best mapping of all the atoms.



Step 11 calculates the sum of all the distances between the mapped atoms in two molecules, if it is smaller than the saved one, it proceeds to the next step. Otherwise, it goes to Step 13. It requires  $3 * n_1 + n_1 - 1$  steps to calculate the sum of all the distances between the mapped atoms and 1 step to compare it to the saved one. Thus, Step 11 requires  $4 * n_1$  steps to finish the process.

Step 12 replaces the saved sum of all the distances between the mapped atoms, as well as the mapping information with the newly calculated ones, which requires  $1 + n_1$  steps.

Step 13 checks whether all the triplets of two molecules have been compared. If so, it continues, otherwise, it goes to Step 5. This requires 1 step.

Step 14 checks if the variable for saving the correct mapping is NULL (which means there is no mapping between all the triplets in the two molecules). If so, it goes to Step 10 to obtain the mapping based on the original orientation of the two molecules. Otherwise, it goes to the next step. This requires 1 step.

Step 15 gets the rotation that minimizes the RMSD between the two molecules based on the best mapping of atoms achieved from the above steps. As calculated in Step 8 and 9, it requires  $96 * n_1 + 16$  steps to get the rotation and  $9 + 15 * n_1 + 3 * n_1 + 3 * n_2 + 3 + 3 * n_1 = 15 + 21 * n_1 + 3 * n_2$  to rotate and translate molecule A. Thus, Step 15 requires  $96 * n_1 + 16 + 15 + 21 * n_1 + 3 * n_2 = 117 * n_1 + 3 * n_2 + 31$  steps in all.

Step 16 calculates the similarity score, which requires  $3 * n_1$  steps to get the RMSD and 1 step to calculate the similarity score from the RMSD, so  $3 * n_1 + 1$  steps in all.

Step 6 to Step 13 is a loop, the number of iterations depends on the number of mapped atom triplets found in Step 6. Thus, the upper bound of the number of mapped atom triplets is  $16 * n_1 * n_2$ , when all the compared atom triplets in Step 5 are mapped.

The upper bound of all the steps of the algorithm is  $391 * (n_1 + n_2) + 208 + 56 * (n_1 + n_2) + 2 + 16 * n_1 * n_2 + 16 * n_1 * n_2 * (6 + 12 + 304 + 15 + 21 * n_1 + 3 * n_2 + n_2^3 + 4n_2^2 + 4 * n_1 + 1 + n_1 + 1) + 117 * n_1 + 3 * n_2 + 31 + 3 * n_1 + 1$ .  $g(n_1, n_2) = 16 * n_1 * n_2^4 + 64 * n_1 * n_2^3 + 48 * n_2^2 * n_1 + 416 * n_1^2 * n_2 + 5440 * n_1 * n_2 + 450 * n_2 + 567 * n_1 + 242$ . Therefore, the time complexity of the algorithm is  $O(n_1 * n_2^4)$ .

During the above time complexity analysis, the worst case of many steps was not obtained, and an upper bound was applied instead. Time complexity obtained by the upper bound analysis describes the upper bound of the run-time for an arbitrary input size of the algorithm. In general the upper bound overestimates the actual run-time of the algorithm. A good estimate for a run-time function would have a growth rate as close to the actual run-time as possible, without unnecessary polynomial terms. According to the above analysis, the term that dominates the run-time upper bound is  $n_1 * n_2^4$ , which means that the number of atoms in the two molecules are the two main factors that affect the upper bound run-time of the algorithm. However, this upper bound is obtained by assuming that all the triplets are mapped, which never happens in practice. If the number of mapped triplets is zero or much less than  $16 * n_1 * n_2$ , the run-time is dramatically over estimated by this upper bound. Thus, the number of mapped triplets may also be an important factor that affects the run-time of the algorithm, which is not included in this time complexity analysis. Conventional run-time complexity parameterizes the run-time function  $f(a)$  with the size of the input, and does not distinguish different input structures, such as correspondence between molecules as used in this thesis. If such factors influence program run-time, they cannot be captured in these kind of complexity functions without characterizing them as sub-problems of their own. Therefore, this time complexity analysis is not able to describe the performance of the algorithm in terms of mapped triplets. It cannot describe the effect on the run-time of the different structures to be compared. The

next section investigates the effect of the number of mapped triplets on the run-time of the algorithm using some empirical run-time data of the algorithm.

## 5.2 Obtaining the run-time data

One hundred structure pairs to be compared are randomly selected from the sub-NCI database to gain one hundred samples of run-time. In order to make sure these one hundred samples cover structures with different numbers of atoms, these structures are not picked entirely randomly. Structures are chosen with the following arbitrary categories based on number of atoms: 18 – 22, 38 – 42, 58 – 62, 78 – 82, 98 – 102, 118 – 122, 138 – 142, 158 – 162, 178 – 182 and 198 – 202. One query structure is randomly chosen from each set and compared to a randomly chosen target structure from the other sets. Then, this process is repeated until a query structure is selected from each set and compared to a target structure from every other set. Each structure can only be selected once as a query structure or a target structure. If, for a certain set, all the structures with that number of atoms have been selected, the range of the number of atoms of that set can be increased by one in order to obtain a unique structure to compare. For every comparison result, in addition to the run-time of the algorithm, the number of atoms and the number of mapped atom triplets between each pair of structures are also recorded. For every comparison, the CPU time of running the algorithm is recorded as the run-time.

The CPU time is captured by the following java code:

```
import java.lang.management.*;

/** Get CPU time in nanoseconds. */
public long getCpuTime( ) {
```

```

ThreadMXBean bean = ManagementFactory.getThreadMXBean( );
return bean.isCurrentThreadCpuTimeSupported( ) ?
    bean.getCurrentThreadCpuTime( ) : 0L;
}

```

Because the single run-times of some comparisons are too short to be captured, each comparison is repeated until the total run-time is more than 1 second, then the run-time of a single run is obtained by dividing the total run-time by the number of times the comparison was run. The run-times of these 100 comparisons are shown in the scatter plot in Figure 5.2. The  $x$  axis is the index of the structure pair to be compared and the  $y$  axis is its run-time (in seconds). The range of the run-time is from 0.00241973s to 1565.111s. The average run-time is 59.42111s and the standard deviation of all these 100 run-times is 202.1817s. The details of all the test results are listed in Appendix E.

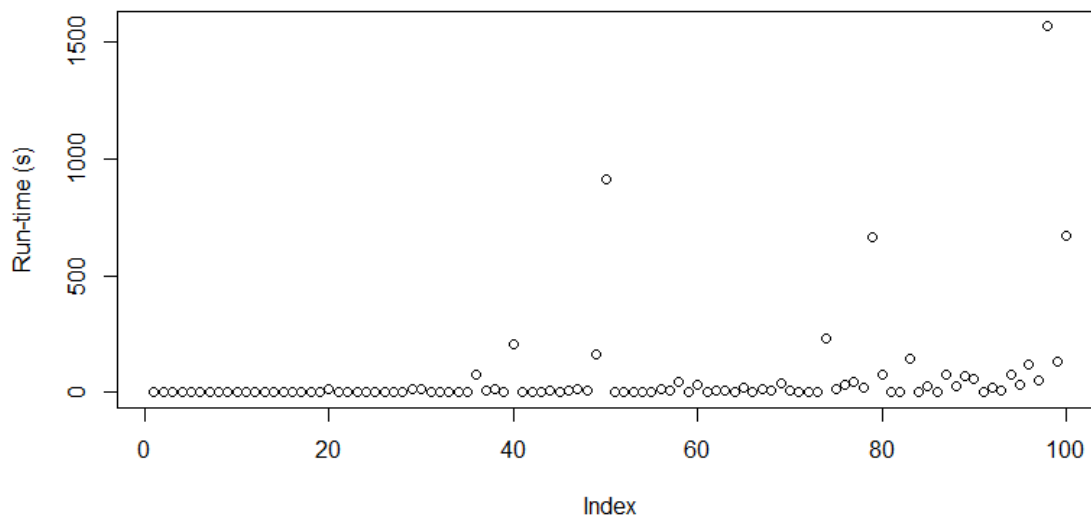


Figure 5.2: The run-time of 100 randomly selected pairs of comparing structures in the sub-NCI database.

### 5.3 Run-time analysis based on the empirical run-time

Linear regression is used to analyze the run-time data obtained in the last section. Linear regression is an approach for modeling the relationship between a scalar dependent variable  $y$  and one or more explanatory variables denoted  $X$ . The data are modeled using a linear model provided by the user, and the coefficients of the model are calculated from the data by minimizing the sum of the least square distances from all the data points to the curve represented by the model. If the model fits all the data points well, the  $R^2$  (multiple  $R^2$ ) value will be close to 1 (if the regression contains more than one explanatory variable  $X$ , the  $R^2$  value is called the multiple  $R^2$  value.).  $R^2$  (multiple  $R^2$ ) is a statistical measure of how close the data are to the fitted regression line, and is always between 0 and 100%. 0 indicates that the model explains none of the variability of the response data around its mean, and 100% indicates that the model explains all the variability of the response data around its mean.

First, the model based on the time complexity analysis is used to fit the 100 run-time data points. The statistical software package R is used to apply the linear regression. To explore the explanatory power of the function  $g(n_1, n_2) = 16 * n_1 * n_2^4 + 64 * n_1 * n_2^3 + 48 * n_2^2 * n_1 + 416 * n_1^2 * n_2 + 5440 * n_1 * n_2 + 450 * n_2 + 567 * n_1 + 242$  obtained in the time complexity analysis, the linear model is described by a polynomial containing the linear terms:  $n_1 * n_2^4$ ,  $n_1 * n_2^3$ ,  $n_2^2 * n_1$ ,  $n_1^2 * n_2$ ,  $n_1 * n_2$ ,  $n_2$ ,  $n_1$ . Thus, the model “run-time  $\sim n_1 * n_2^4 + n_1 * n_2^3 + n_2^2 * n_1 + n_1^2 * n_2 + n_1 * n_2 + n_2 + n_1$ ” is applied to the 100 run-time data points. The calculated multiple  $R^2$  of this model is 0.2578, indicating that this model only explained 25.78% of the variability of the response data around its mean, which indicates the function  $g(n_1, n_2)$  poorly describes the run-time of the actual cases. The number of atoms in the molecules vs. the run-time are plotted in

Figures 5.3 and 5.4. It can be seen that there is no recognizable correlation between the run-time and the sizes of the molecules to be compared.

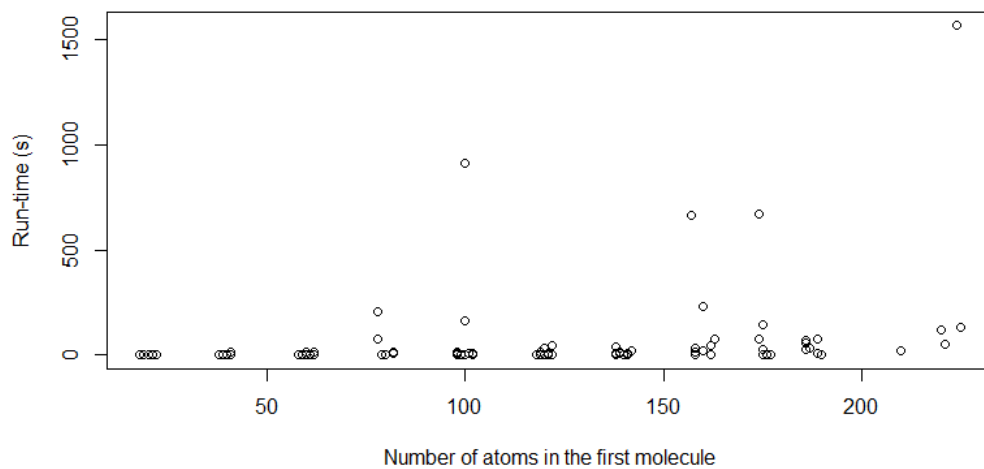


Figure 5.3: Number of atoms in the first molecule vs. the run-time

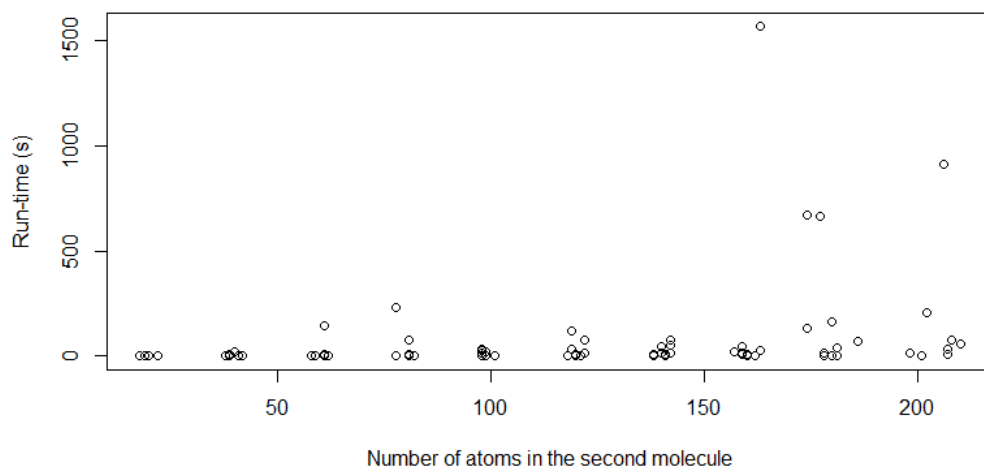


Figure 5.4: Number of atoms in the second molecule vs. the run-time

Thus, the upper bound of the run-time obtained by complexity analysis is of limited value. The run-time is not dominated by the number of atoms in the molecules being

compared, the parameter associated with classical complexity analysis.

Next the number of mapped triplets is used in a linear model. The number of mapped triplets vs. the run-time is plotted in Figure 5.5. An increase in the run-time is observed with an increase in the number of mapped triplets.

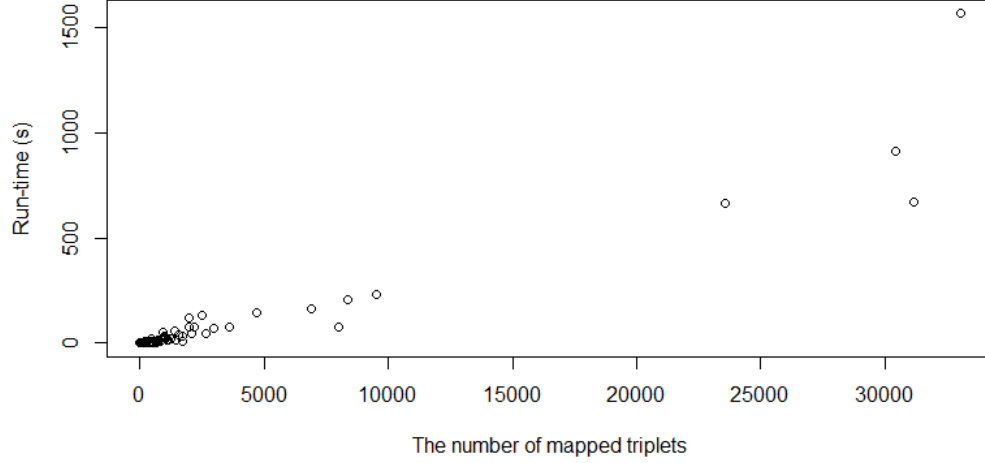


Figure 5.5: The number of mapped triplets vs. the run-time

Adding the variable for the number of mapped triplets ( $k$ ) in the function  $g(n_1, n_2)$  from time complexity analysis, by replacing the number of mapped triplets ( $16 * n_1 * n_2$ ) with  $k$ , a function  $g(n_1, n_2, k)$  can be obtained as  $k * n_2^3 + 4 * k * n_2^2 + 3 * k * n_2 + 26 * k * n_1 + 16 * n_1 * n_2 + 339k + 431 * n_1 + 314 * n_2 + 242$ . Thus, the model “run-time  $\sim k * n_2^3 + k * n_2^2 + k * n_2 + k * n_1 + n_1 * n_2 + k + n_2 + n_1$ ” is applied to the 100 run-time data points. By running the linear regression, the multiple  $R^2$  is 0.9931, which means that 99.31% of the variability of the run-time around its mean can be described by this model. The detailed results of the linear regression generated by R are listed in Table 5.1 and the residual plot is in Figure 5.6. It can be seen that there are no apparent patterns of bias in the residual plot, which indicates no extra factors are needed to explain any bias in the run-time data.

Term	Coefficients	Standard Error	P-value
Intercept	$1.806 * 10^1$	$1.051 * 10^1$	0.0890
$k$	$-3.227 * 10^{-1}$	$7.032 * 10^{-2}$	$1.42 * 10^{-5}$
$n_2$	$-1.118 * 10^{-1}$	$7.233 * 10^{-2}$	0.1258
$n_1$	$-2.229 * 10^{-1}$	$1.770 * 10^{-1}$	0.2111
$k * n_2^3$	$8.060 * 10^{-8}$	$1.393 * 10^{-8}$	$1.00 * 10^{-7}$
$k * n_2^2$	$-3.876 * 10^{-5}$	$7.337 * 10^{-6}$	$8.65 * 10^{-7}$
$k * n_2$	$6.267 * 10^{-3}$	$1.264 * 10^{-3}$	$3.29 * 10^{-6}$
$k * n_1$	$2.634 * 10^{-5}$	$1.103 * 10^{-5}$	0.0191
$n_1 * n_2$	$1.511 * 10^{-3}$	$1.010 * 10^{-3}$	0.1382

Table 5.1: The linear regression results generated by R.

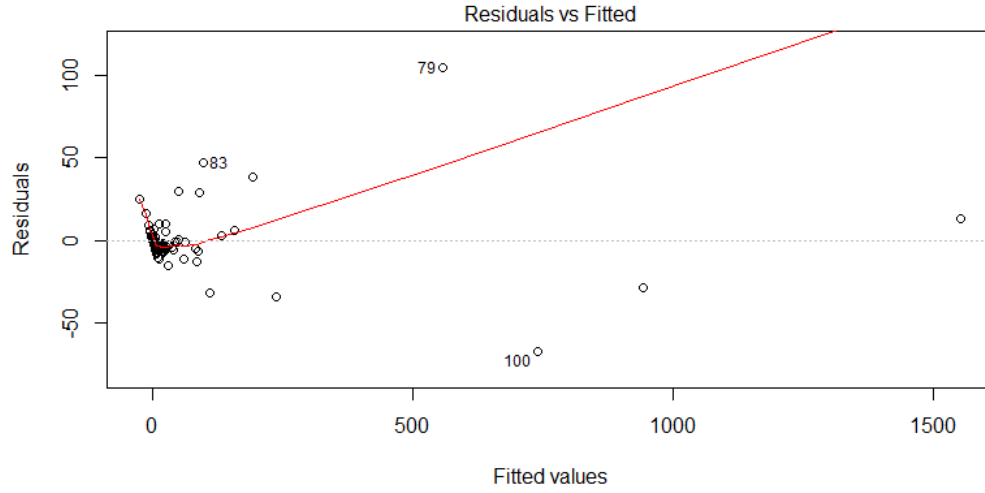


Figure 5.6: The residual plot of the linear regression generated by R.

As can be seen from Table 5.1, five terms that contain the number of mapped triplets ( $k$ ) have the five lowest p-values among all the other variables. The p-value



for each term tests the null hypothesis that the term does not explain any variance in the run-time. A low p-value ( $< 0.05$ ) indicates that the null hypothesis can be rejected, which indicates that this term is likely to be a meaningful addition to the model because changes in this term's value are related to changes in the run-time. With the non-significant factors ( $n_1$ ,  $n_2$ ,  $n_1 * n_2$ ) dropped, the  $R^2$  is 0.9928. Thus, the number of mapped triplets is a dominant variable that affects the performance of the algorithm.

It should be noted that the linear regression result obtained here is a polynomial that fits the trend of the run-time of the algorithm. If this polynomial is used to predict the actual run-time for a specific case, the result may not be accurate. In addition, the data points are spread out over a large range of values in Figure 5.2, which also indicates the prediction of this kind of data with large standard deviation may result in large error. Furthermore, the number of mapped triplets cannot be obtained before running the algorithm. Therefore, this polynomial cannot reliably predict the run-time of the algorithm for specific cases.

## 5.4 Some examples of the empirical run-time of the algorithm

Different tests of the run-time are presented in this section. Although the specific molecules influence the exact run-time, users can obtain an estimate of how long the algorithm may take by using these run-time data as a reference. All of the tests here are run on the same personal computer as the one being used for Chapter 4 with an AMD Athlon™ dual core processor 4850e 2.5GHz CPU, 4 GB RAM and the 64-bit Windows 7 operating system.

Figure 5.7 shows the run-time distribution of 3900 randomly selected pairs of struc-

tures (selected by the same method introduced in Section 5.2), ranging from 18 to 225 atoms. The run-time of all these 3900 comparisons are in the folder named “3900 run time test result” in the zip file attached to this thesis.

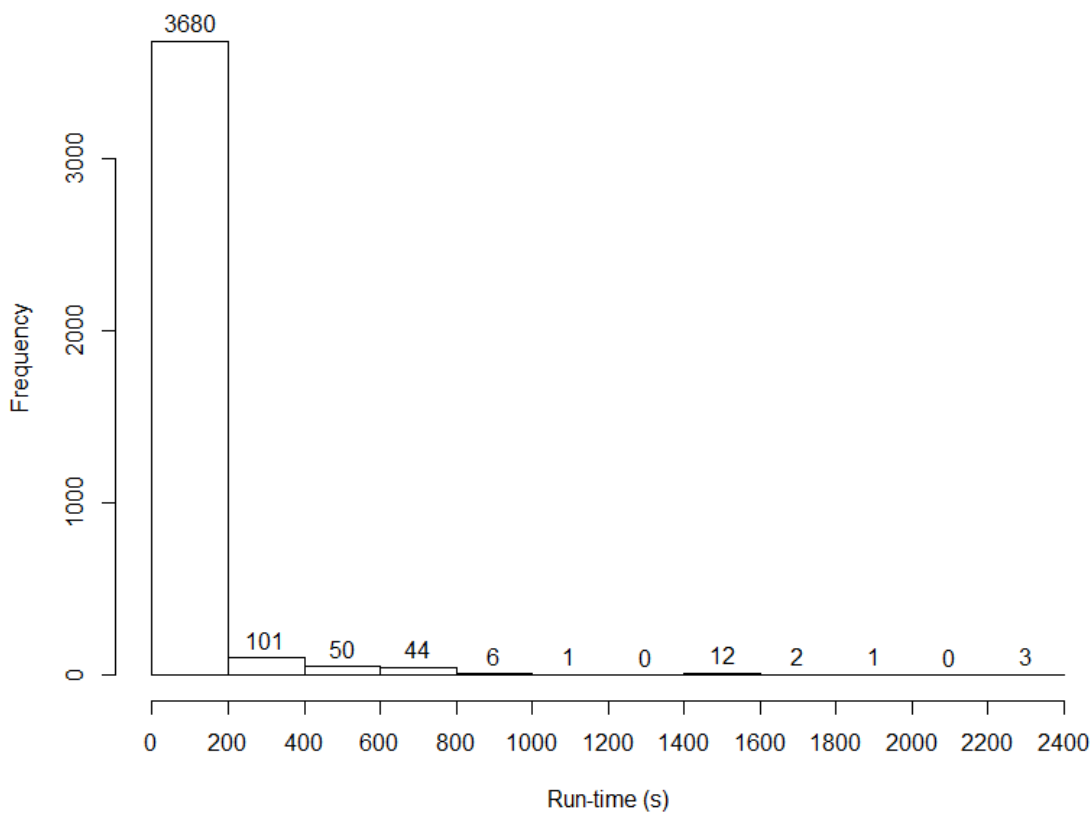


Figure 5.7: The run-time distribution of 3900 randomly selected pairs of structures in the sub-NCI database.

The run-time of 94.4% of cases (3680 out of 3900) is under 200s. Only 0.64% of comparisons (25 out of 3900) take over 800s to finish the comparison. For example, the case of the longest run-time in the test data is structure  $C_{70}H_{138}N_8O_4S_4$  (Figure 5.8) compared to itself (exact same file).

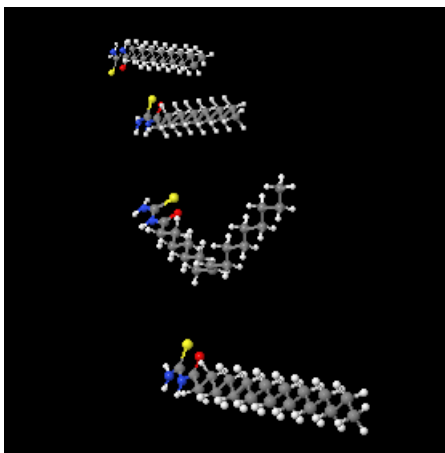


Figure 5.8: Chemical structure  $C_{70}H_{138}N_8O_4S_4$ .

It took 2362.432s to run the algorithm. This structure has 224 atoms and contains many identical three-atom substructures. When the algorithm is comparing this structure to itself, each triplet is mapped to all the identical triplets in the other copy of the structure. This situation is close to the worst case mentioned in the time complexity calculation (as discussed in Section 5.1.2). Thus, the number of mapped triplets for this comparison is extremely large: 52984. As mentioned in the linear regression analysis (Section 5.3), the number of mapped triples is the most significant term that affects the run-time of the algorithm. This is the main reason why the run-time of this comparison is so long.

For 25 molecules randomly selected from the sub-NCI database, with structures in the 4 to 100 atom range, the run-time data to search the entire sub-NCI database for a similar structure is listed in Table 5.2. The details of the result files are in the folder named “Run time results of searching the sub-NCI data set” in the zip file attached to this thesis. The sub-NCI database is categorized into 5 groups: 0 – 20, 21 – 40, 41 – 60, 61 – 80, and > 80 atoms. Five molecules are randomly selected from each category to search the whole sub-NCI database. These run-times can provide a rough run-time prediction as a reference for those who want to use the algorithm to search

Query Structure	Number of atoms	Run-time(s)	Run-time(h)
$C_7H_{10}N_2$	19	4294.9883318	1.19
$C_8H_8O_2$	18	3889.4325321	1.08
$C_6H_9O_2Br_3$	20	5921.2831567	1.64
$C_7H_9NO$	18	4280.7610406	1.19
$C_4H_4N_6S$	15	506.2076449	0.14
$C_{10}H_{21}BO_2$	34	16383.6930230	4.55
$C_{16}H_{16}O_2$	34	16173.1228732	4.49
$C_{13}H_{14}N_2O_2$	31	13100.4023764	3.64
$C_7H_6N_4O_6$	23	1622.2231988	0.45
$C_{16}H_{14}N_2O_6S$	38	17656.6767831	4.90
$C_{18}H_{18}O_4S$	41	19759.3206615	5.49
$C_{17}H_{26}N_2O_2Cl_4$	51	13472.8235637	3.74
$C_{21}H_{20}N_2O_6S$	50	37706.0345038	10.47
$C_{19}H_{21}NO_2$	43	31774.9232841	8.83
$C_{23}H_{18}N_6OS_2$	50	38831.8313204	10.79
$C_{25}H_{43}N_2O_4SCl$	76	180604.7289152	50.17
$C_{26}H_{41}N_2O_4Cl$	74	90847.9779551	25.24
$C_{36}H_{27}N$	64	122826.8569466	34.12
$C_{23}H_{38}O_5$	66	127223.3395290	35.34
$C_{36}H_{25}N_2P$	64	109882.4711703	30.52
$C_{31}H_{56}O_2$	89	342824.8959814	95.23
$C_{32}H_{38}N_6O_2Cl_4$	82	80888.5153128	22.47
$C_{43}H_{36}N_6O_6$	91	175906.1971966	48.86
$C_{34}H_{29}N_6O_{20}NaS_6$	96	96468.1323815	26.80
$C_{36}H_{41}N_3O_4$	84	165219.9258953	45.89

Table 5.2: 25 randomly selected structures to search the sub-NCI database.

the NCI database.

Although most of the run-times in Table 5.2 are several hours, this is not unusual in computational chemistry. Furthermore, when using this algorithm to search the database, a chemist could set some filtering options first, which will be discussed in more detail in the next chapter. Thus, the actual run-time for searching the database could be significantly improved.

## 5.5 Conclusion

The analysis of the run-time in this chapter reveals that the number of mapped triplets, and the number of atoms in each structure are the terms that affect the run-time of the algorithm. However, the number of mapped triplets is the dominant of these three. The more mapped triplets found, the longer time the algorithm takes.

## Chapter 6

### Conclusion and future work

This thesis details an algorithm for the automated comparison of chemical structures. The algorithm solves the traditional labeling problem in superimposition methods. Thus, the algorithm is not only able to compute a similarity score for two chemical structures, but is also able to search a large collection of chemical structures for similar ones. Using data from two data sets, the algorithm is shown to work well qualitatively. This thesis also analyses the time complexity of the algorithm and presents a practical average run-time analysis based on the data sets. Based on the test run-time data, the algorithm may take less than 200 seconds for a single comparison (3680 out of 3900 test cases (94.4%) are under 200s, 2367 out of 3900 test cases (60.7%) are under 10s), only 0.64% of comparisons (25 out of 3900) took over 800 seconds. If using the algorithm to search the sub-NCI database that contains 163652 structures, it took at least 506.21 seconds and at most 342824.9 seconds to finish the search owing to different query structures. These run-times are all captured by testing the algorithm on a personal computer with an AMD Athlon™ dual core processor 4850e 2.5GHz CPU, 4 GB RAM and the 64-bit Windows 7 operating system. Therefore, this algorithm is functional and practical for use.

However, the algorithm still has the potential to be improved. First, since the algorithm only compares chemical structures mainly on a geometric basis, if two structures are isomers or are in the same cluster, it cannot output a result to notify the user that these two structures are isomers or the same molecules. Although there is a method available to check if two structures are isomers using the SMILES (Simplified Molecular-Input Line-Entry System) strings (a specification in the form of a line notation for describing the structure of chemical molecules using short ASCII strings.), its results are not reliable. Thus, a method of comparing isomers and molecules in the same cluster according to geometry would be a good improvement for the algorithm. Second, the result of using the algorithm to search the entire sub-NCI database shows that it takes hours to finish, owing to the algorithm comparing every structure in the database one by one. In practical applications of this algorithm, the user could add some filters based on the particular needs. For example, if a chemist is looking for a structure in the NCI database that is similar to a 20-atom query structure, he may not be interested in the structures with sizes that are twice as large as the query structure. Thus, the algorithm does not need to search the structures with more than 40 atoms. Or maybe the chemist is only interested in structures that consist of certain atoms; then the algorithm does not need to search the structures that consist of other atoms. Consequently, the run-time can be significantly reduced. Third, the user also can add more conditions to the algorithm to obtain more specific comparison or search results. For example, the user can add a condition that considers chlorine equal to fluorine. Thus, the molecules with the same structure but different in chlorine or fluorine atoms will have the same metrics in the results. Finally, in the run-time analysis section, the thesis presents a polynomial to predict the run-time. However, the polynomial has a term that is the number of mapped atom-triplets between the two structures, for which it is hard to obtain a value before running the algorithm.

Maybe it is possible to estimate this number from a more thorough characterization of chemical structures.

Overall, the algorithm has many practical applications, not just limited to using it as introduced in this thesis. This algorithm has proved to be useful and worth developing to additional applications in the future.



# Appendix A

## Detailed results of Table 4.1

The test results of using molecules  $C_5H_6N_5O$ ,  $C_5H_8N_5O_2$ ,  $C_5H_{12}N_2O_4$ ,  $C_5H_{10}N_2O_3$  and  $C_7H_{10}O_2$  in Table 4.1 as the query molecule to search the whole 737-data-set are listed in this appendix. The results are text files generated by the application for the test. The first line of the file contains the query molecule’s filename, the number of atoms and the atom triplets found in it. The remainder is the molecules the algorithm found that are similar to the query molecule in a most-to-least-similar order. These molecules are listed in a tab-delimited table with eight columns. Each row represents a molecule. Column “Structure” lists the filename of this molecule and the directory of the file. Column “score” is the similarity score the algorithm calculated when comparing this molecule to the query molecule. Column “rmsd” is the RMSD between this molecule and the query molecule. Column “size” is the number of the atoms in this molecule. Column “same” is the number of identical atoms between this molecule and the query molecule. Column “all” is the larger of the number of atoms of this molecule and the query molecule. Column “triangles” is the number of atom triplets the algorithm found in this molecule. Column “matched” is the number of mapped atom triplets between the query molecule and this molecule.

The following is the first eight rows of the output file generated by the application when using molecule  $C_5H_6N_5O$  to search the entire 737-data-set (the first row in Table 4.1):

```
GS_Ade_PathwayA_I1A_B3LYP_631Gd.out1.ar: (17atoms, 28triangles)
structure      score      rmsd      size  same  all  triangles  matched
D:\\structures\\GS\\GS_Ade_PathwayA_I1A_B3LYP_631Gd.out1.ar  7.824592E-8  7.824592E-8  17   17   17   28        36
D:\\structures\\GS\\GS_Ade_PathwayA_I1A_G3B3.out1.ar        2.3375972E-7  2.3375972E-7  17   17   17   28        36
D:\\structures\\GS\\GS_Ade_PathwayA_I1A_G3MP2B3.out1.ar     2.3375972E-7  2.3375972E-7  17   17   17   28        36
D:\\structures\\GS\\GS_Ade_PathwayA_I1A_CBSQB3.out1.ar      0.013064672  0.013064672  17   17   17   28        36
D:\\structures\\GS\\GS_Ade_PathwayA_I1A_MP2_631Gd.out1.ar   0.022226611  0.022226611  17   17   17   28        36
D:\\structures\\GS\\GS_Ade_PathwayA_I1A_B3LYP_631pGd.out1.ar 0.031945907  0.031945907  17   17   17   28        36
...(more comparison results of other 731 molecules)
```

The following is the first eight rows of the output file generated by the application when using molecule  $C_5H_8N_5O_2$  to search the entire 737-data-set (the second row in Table 4.1):

```
GS_Ade_PathwayB_I1B_B3LYP_631pGd.out1.ar: (20atoms, 29triangles)
structure      score      rmsd      size  same  all  triangles  matched
D:\\structures\\GS\\GS_Ade_PathwayB_I1B_B3LYP_631pGd.out1.ar 7.751953E-8  7.751953E-8  20   20   20   29        38
D:\\structures\\GS\\GS_Ade_PathwayB_I1B_HF_631Gd.out1.ar     0.08340267  0.08340267  20   20   20   29        36
D:\\structures\\GS\\GS_Ade_PathwayB_I1B_MP2_631Gd.out1.ar    0.1279925  0.1279925  20   20   20   29        38
D:\\structures\\GS\\GS_Ade_PathwayB_I1B_CBSQB3.out1.ar      0.13690503  0.13690503  20   20   20   29        38
D:\\structures\\TS\\TS_Ade_PathwayC_TS2C_G3MP2B3.out1.ar     0.3397378  0.25111055  23   17   23   37        26
D:\\structures\\GS\\GS_Ade_PathwayF_I2F_HF_631Gd.out1.ar     0.38504344  0.30482605  24   19   24   32        28
...(more comparison results of other 731 molecules)
```

The following is the first eight rows of the output file generated by the application when using molecule  $C_5H_{12}N_2O_4$  to search the entire 737-data-set (the third row in Table 4.1):

```
GS_DHA_P1_B3LYP_631Gd.log1.ar: (23atoms, 32triangles)
structure      score      rmsd      size  same  all  triangles  matched
D:\\structures\\GS\\GS_DHB_P1_B3LYP_631Gd.log1.ar          8.656624E-8  8.656624E-8  23   23   23   32        76
D:\\structures\\GS\\GS_DHA_P1_B3LYP_631Gd.log1.ar          8.656624E-8  8.656624E-8  23   23   23   32        76
D:\\structures\\GS\\GS_DHA_P1N_G3MP2B3.log1.ar             1.825114E-7  1.825114E-7  23   23   23   32        76
D:\\structures\\GS\\GS_DHB_P1_G3MP2B3.log1.ar             1.825114E-7  1.825114E-7  23   23   23   32        76
D:\\structures\\TS\\TS_IRCF_P_DHA_TS1_B3LYP_631Gd.log1.ar   7.410906E-4  7.410906E-4  23   23   23   32        76
D:\\structures\\TS\\TS_IRCF_P_DHB_TS2_B3LYP_631Gd.log1.ar   8.335496E-4  8.3354954E-4  23   23   23   32        76
...(more comparison results of other 731 molecules)
```

The following is the first eight rows of the output file generated by the application

when using molecule  $C_5H_{10}N_2O_3$  to search the entire 737-data-set (the fourth row in Table 4.1):

GS\_GDA\_P1b\_B3LYP\_631pGdp\_SMD.log1.ar: (20atoms,31triangles)

structure	score	rmsd	size	same	all	triangles	matched
D:\\structures\\GS\\GS_GDA_P1b_B3LYP_631pGdp_SMD.log1.ar	7.23501E-8	7.23501E-8	20	20	20	31	68
D:\\structures\\GS\\GS_GDA_P1_B3LYP_631pGdp_PCM.log1.ar	0.16244033	0.16244033	20	20	20	31	68
D:\\structures\\TS\\TS_IRCF_I2b_PHC_TS3_B3LYP_631pGdp.log1.ar	1.2166932	0.9125199	18	15	20	32	38
D:\\structures\\GS\\GS_GDA_P1_B3LYP_631pGdp_SMD.log1.ar	1.2433121	1.1189809	20	18	20	37	39
D:\\structures\\TS\\TS_IRCR_P_DGA_TS1_B3LYP_631pGdp_SMD.log1.ar	1.2438437	1.1194594	20	18	20	37	39
D:\\structures\\GS\\GS_DHD_P1_B3LYP_631Gd.log1.ar	1.251775	0.85348296	22	15	22	37	53

...(more comparison results of other 731 molecules)

The following is the first eight rows of the output file generated by the application when using molecule  $C_7H_{10}O_2$  to search the entire 737-data-set (the fifth row in Table 4.1):

TS\_C7H10O2\_endo3B\_HF631ppGd.log1.ar: (19atoms,36triangles)

structure	score	rmsd	size	same	all	triangles	matched
D:\\structures\\TS\\TS_C7H10O2_endo3B_HF631ppGd.log1.ar	7.648167E-8	7.648167E-8	19	19	19	36	92
D:\\structures\\TS\\TS_C7H10O2_endo_3B_B3LYP631ppGd.log1.ar	0.044035513	0.044035513	19	19	19	36	63
D:\\structures\\TS\\TS_C7H10O_endo_syn_G_HF631ppGd.log1.ar	0.13386537	0.11272874	18	16	19	35	60
D:\\structures\\TS\\TS_C7H10O_endo_anti_S_HF631ppGd.log1.ar	0.17247841	0.14524497	18	16	19	35	60
D:\\structures\\TS\\TS_C7H10O_endo_anti_S_HF631Gd.log1.ar	0.1786975	0.1504821	18	16	19	35	60
D:\\structures\\TS\\TS_C7H10_endo_HF631ppGd.log1.ar	0.23298095	0.171670	17	14	19	34	60

...(more comparison results of other 731 molecules)

## Appendix B

### Detailed results of using molecules in Table 4.1 to search the 737r-data-set

The test results of using molecules  $C_5H_6N_5O$ ,  $C_5H_8N_5O_2$ ,  $C_5H_{12}N_2O_4$ ,  $C_5H_{10}N_2O_3$  and  $C_7H_{10}O_2$  in Table 4.1 as the query molecule to search the entire 737r-data-set are listed in this appendix. The filename of every molecule in the 737r-data-set is the same filename of the molecule in the 737-data-set (before a random rotation). The results are text files generated by the application for the test. The first line of the file contains the query molecule’s filename, the number of atoms and the atom triplets found in it. The remainder is the molecules the algorithm found that are similar to the query molecule in a most-to-least-similar order. These molecules are listed in a tab-delimited table with eight columns. Each row represents a molecule. Column “Structure” lists the filename of this molecule and the directory of the file. Column “score” is the similarity score the algorithm calculated when comparing this molecule to the query molecule. Column “rmsd” is the RMSD between this molecule

and the query molecule. Column “size” is the number of the atoms in this molecule. Column “same” is the number of identical atoms between this molecule and the query molecule. Column “all” is the larger of the number of atoms of this molecule and the query molecule. Column “triangles” is the number of atom triplets the algorithm found in this molecule. Column “matched” is the number of mapped atom triplets between the query molecule and this molecule.

The following is the first eight rows of the output file generated by the application when using molecule  $C_5H_6N_5O$  in the 737-data-set to search the entire 737r-data-set:

```
GS_Ade_PathwayA_I1A_B3LYP_631Gd.out1.ar: (17atoms, 28triangles)
structure                                score          rmsd          size  same  all  triangles  matched
D:\\TestFileConvert\\GS\\GS_Ade_PathwayA_I1A_G3MP2B3.out1.ar    1.3036896E-7    1.3036896E-7    17   17   17   28         38
D:\\TestFileConvert\\GS\\GS_Ade_PathwayA_I1A_B3LYP_631Gd.out1.ar  1.6501266E-7    1.6501266E-7    17   17   17   28         38
D:\\TestFileConvert\\GS\\GS_Ade_PathwayA_I1A_G3B3.out1.ar        2.650705E-7    2.650705E-7    17   17   17   28         38
D:\\TestFileConvert\\GS\\GS_Ade_PathwayA_I1A_CBSQB3.out1.ar      0.013064702    0.013064702    17   17   17   28         38
D:\\TestFileConvert\\GS\\GS_Ade_PathwayA_I1A_MP2_631Gd.out1.ar    0.022226539    0.022226539    17   17   17   28         38
D:\\TestFileConvert\\GS\\GS_Ade_PathwayA_I1A_B3LYP_631pGd.out1.ar 0.031945888    0.031945888    17   17   17   28         38
...(more comparison results of other 731 molecules)
```

The following is the first eight rows of the output file generated by the application when using molecule  $C_5H_8N_5O_2$  in the 737-data-set to search the entire 737r-data-set:

```
GS_Ade_PathwayB_I1B_B3LYP_631pGd.out1.ar: (20atoms, 29triangles)
structure                                score          rmsd          size  same  all  triangles  matched
D:\\TestFileConvert\\GS\\GS_Ade_PathwayB_I1B_B3LYP_631pGd.out1.ar 1.9156103E-7    1.9156103E-7    20   20   20   29         40
D:\\TestFileConvert\\GS\\GS_Ade_PathwayB_I1B_HF_631Gd.out1.ar     0.08340269     0.08340269     20   20   20   29         40
D:\\TestFileConvert\\GS\\GS_Ade_PathwayB_I1B_MP2_631Gd.out1.ar    0.12799248     0.12799248     20   20   20   29         40
D:\\TestFileConvert\\GS\\GS_Ade_PathwayB_I1B_CBSQB3.out1.ar       0.13690506     0.13690506     20   20   20   29         40
D:\\TestFileConvert\\TS\\TS_Ade_PathwayC_TS2C_G3MP2B3.out1.ar    0.33973783     0.25111058     23   17   23   37         28
D:\\TestFileConvert\\GS\\GS_Ade_PathwayF_I2F_HF_631Gd.out1.ar    0.38504344     0.30482605     24   19   24   32         36
...(more comparison results of other 731 molecules)
```

The following is the first eight rows of the output file generated by the application when using molecule  $C_5H_{12}N_2O_4$  in the 737-data-set to search the entire 737r-data-set:

```

GS_DHA_P1_B3LYP_631Gd.log1.ar: (23atoms, 32triangles)
structure                                score          rmsd          size  same  all  triangles  matched
D:\\TestFileConvert\\GS\\GS_DHA_P1N_G3MP2B3.log1.ar  1.910293E-7    1.910293E-7    23    23    23    32        76
D:\\TestFileConvert\\GS\\GS_DHA_P1_B3LYP_631Gd.log1.ar  2.055007E-7    2.0550071E-7    23    23    23    32        76
D:\\TestFileConvert\\GS\\GS_DHB_P1_G3MP2B3.log1.ar  3.076233E-7    3.076233E-7    23    23    23    32        76
D:\\TestFileConvert\\GS\\GS_DHB_P1_B3LYP_631Gd.log1.ar  3.0786248E-7    3.0786248E-7    23    23    23    32        76
D:\\TestFileConvert\\TS\\TS_IRCF_P_DHA_TS1_B3LYP_631Gd.log1.ar  7.4109243E-4    7.4109243E-4    23    23    23    32        76
D:\\TestFileConvert\\TS\\TS_IRCF_P_DHB_TS2_B3LYP_631Gd.log1.ar  8.335505E-4    8.335506E-4    23    23    23    32        76
...(more comparison results of other 731 molecules)

```

The following is the first eight rows of the output file generated by the application when using molecule  $C_5H_{10}N_2O_3$  in the 737-data-set to search the entire 737r-data-set:

```

GS_GDA_P1b_B3LYP_631pGdp_SMD.log1.ar: (20atoms, 31triangles)
structure                                score          rmsd          size  same  all  triangles  matched
D:\\structures\\GS\\GS_GDA_P1b_B3LYP_631pGdp_SMD.log1.ar  7.23501E-8     7.23501E-8     20    20    20    31        68
D:\\structures\\GS\\GS_GDA_P1_B3LYP_631pGdp_PCM.log1.ar  0.16244033    0.16244033     20    20    20    31        68
D:\\structures\\TS\\TS_IRCF_I2b_PHC_TS3_B3LYP_631pGdp.log1.ar  1.2166932     0.9125199     18    15    20    32        38
D:\\structures\\GS\\GS_GDA_P1_B3LYP_631pGdp_SMD.log1.ar  1.2433121     1.1189809     20    18    20    37        39
D:\\structures\\TS\\TS_IRCR_P_DGA_TS1_B3LYP_631pGdp_SMD.log1.ar  1.2438437     1.1194594     20    18    20    37        39
D:\\structures\\GS\\GS_DHD_P1_B3LYP_631Gd.log1.ar  1.251775      0.85348296     22    15    22    37        53
...(more comparison results of other 731 molecules)

```

The following is the first eight rows of the output file generated by the application when using molecule  $C_7H_{10}O_2$  in the 737-data-set to search the entire 737r-data-set:

```

TS_C7H10O2_endo3B_HF631ppGd.log1.ar: (19atoms, 36triangles)
structure                                score          rmsd          size  same  all  triangles  matched
D:\\TestFileConvert\\GS\\GS_GDA_P1b_B3LYP_631pGdp_SMD.log1.ar  1.3220114E-7    1.3220114E-7    20    20    20    31        70
D:\\TestFileConvert\\GS\\GS_GDA_P1_B3LYP_631pGdp_PCM.log1.ar  0.16244034    0.16244034     20    20    20    31        70
D:\\TestFileConvert\\TS\\TS_IRCF_I2b_PHC_TS3_B3LYP_631pGdp.log1.ar  1.2166934     0.91252005     18    15    20    32        41
D:\\TestFileConvert\\GS\\GS_GDA_P1_B3LYP_631pGdp_SMD.log1.ar  1.2433121     1.1189809     20    18    20    37        41
D:\\TestFileConvert\\TS\\TS_IRCR_P_DGA_TS1_B3LYP_631pGdp_SMD.log1.ar  1.2438437     1.1194594     20    18    20    37        41
D:\\TestFileConvert\\GS\\GS_DHD_P1_G3MP2B3.log1.ar  1.251775      0.85348296     22    15    22    37        55
...(more comparison results of other 731 molecules)

```

# Appendix C

## Detailed results of Table 4.2

The test results of using molecules 4-methyl-1, 2-benzenediamine ( $C_7H_{10}N_2$ ); ethyl 2, 3-dihydro-1H-pyrrolo[1,2-a]benzimidazole-8-carboxylate ( $C_{13}H_{14}N_2O_2$ ); ethyl ((4-(hydroxy(oxido)amino)-2-methyl-1-naphthyl)(phenylsulfonyl)amino)acetate ( $C_{21}H_{20}N_2O_6S$ ); 4-chloro-N-hexadecyl-3-(hydroxy(oxido)amino)-N-isopropyl benzene-sulfonamide ( $C_{25}H_{43}N_2O_4SCl$ ) and 4,4-bis(4-(((4-(phenyldiazenyl)anilino)carbonyl)oxy)phenyl)pentanoic acid ( $C_{43}H_{36}N_6O_6$ ) in Table 4.2 as the query molecule to search the entire sub-NCI data set are listed in this appendix. The results are text files generated by the application for the test. The first line of the file contains the query molecule’s filename, the number of atoms and the atom triplets found in it. The remainder is the molecules the algorithm found that are similar to the query molecule in a most-to-least-similar order. These molecules are listed in a tab-delimited table with eight columns. Each row represents a molecule. Column “Structure” lists the filename of this molecule and the directory of the file. Column “score” is the similarity score the algorithm calculated when comparing this molecule to the query molecule. Column “rmsd” is the RMSD between this molecule and the query molecule. Column “size” is the number of the atoms in this molecule. Column “same” is the number of

identical atoms between this molecule and the query molecule. Column “all” is the larger of the number of atoms of this molecule and the query molecule. Column “triangles” is the number of atom triplets the algorithm found in this molecule. Column “matched” is the number of mapped atom triplets between the query molecule and this molecule.

The following is the first seven rows of the output file generated by the application when using molecule 4-methyl-1, 2-benzenediamine ( $C_7H_{10}N_2$ ) to search the entire sub-NCI data set (the first row in Table 4.2):

```
1487_NCI.lis:(19atoms,30triangles)
structure                score          rmsd          size  same  all  triangles  matched
D:\\NCI_Trim\\1400\\1487_NCI.lis    6.080277E-8  6.080277E-8   19   19   19   30         84
D:\\NCI_Trim\\111000\\111019_NCI.lis 0.015928725 0.01257531   17   15   19   27         59
D:\\NCI_Trim\\1400\\1482_NCI.lis    0.06428598  0.04398514   16   13   19   25         41
D:\\NCI_Trim\\1400\\1481_NCI.lis    0.069995396 0.047891587   16   13   19   25         40
D:\\NCI_Trim\\76000\\76019_NCI.lis   0.0700789   0.047948726   16   13   19   25         40
...(more comparison results of other 163647 molecules)
```

The following is the first seven rows of the output file generated by the application when using molecule ethyl 2, 3-dihydro-1H-pyrrolo[1,2-a]benzimidazole-8-carboxylate ( $C_{13}H_{14}N_2O_2$ ) to search the entire sub-NCI data set (the second row in Table 4.2):

```
108547_NCI.lis:(31atoms,59triangles)
structure                score          rmsd          size  same  all  triangles  matched
D:\\NCI_Trim\\108500\\108547_NCI.lis 1.02438456E-7 1.02438456E-7  31   31   31   59        125
D:\\NCI_Trim\\105700\\105780_NCI.lis 0.672143      0.43364063   26   20   31   44         76
D:\\NCI_Trim\\99200\\99250_NCI.lis   0.7822722    0.52992636   28   21   31   46         95
D:\\NCI_Trim\\199800\\199847_NCI.lis 0.7841758    0.5059199    26   20   31   43         77
D:\\NCI_Trim\\205000\\205011_NCI.lis 0.80119514    0.20676003   19    8   31   31          1
...(more comparison results of other 163647 molecules)
```

The following is the first seven rows of the output file generated by the application when using molecule ethyl ((4-(hydroxy(oxido)amino)-2-methyl-1-naphthyl) (phenylsulfonyl)amino)acetate ( $C_{21}H_{20}N_2O_6S$ ) to search the entire sub-NCI data set



(the third row in Table 4.2):

```
118972_NCI.lis: (50atoms,88triangles)
structure                score          rmsd          size  same  all  triangles  matched
D:\\NCI_Trim\\118900\\118972_NCI.lis  1.0505697E-7  1.0505697E-7   50   50   50   88       439
D:\\NCI_Trim\\116700\\116735_NCI.lis  0.9432973    0.62257624    42   33   50   73       335
D:\\NCI_Trim\\190000\\190006_NCI.lis  1.3381022    0.58876497    29   22   50   50       314
D:\\NCI_Trim\\122000\\122069_NCI.lis  1.3860861    0.69304305    36   25   50   63       374
D:\\NCI_Trim\\222300\\222312_NCI.lis  1.5278447    0.70280856    36   23   50   61       244
D:\\NCI_Trim\\116700\\116777_NCI.lis  1.8689374    0.82233244    36   22   50   63       377
...(more comparison results of other 163647 molecules)
```

The following is the first seven rows of the output file generated by the application when using molecule 4-chloro-N-hexadecyl-3-(hydroxy(oxido)amino)-N-isopropyl benzenesulfonamide ( $C_{25}H_{43}N_2O_4SCl$ ) to search the entire sub-NCI data set (the fourth row in Table 4.2):

```
113567_NCI.lis: (76atoms,144triangles)
structure                score          rmsd          size  same  all  triangles  matched
D:\\NCI_Trim\\113500\\113567_NCI.lis  1.7218684E-7  1.7218684E-7   76   76   76   144      3668
D:\\NCI_Trim\\9300\\9324_NCI.lis    0.22195421    0.0817726     32   28   76   60      1824
D:\\NCI_Trim\\6000\\6004_NCI.lis    0.22995637    0.08472077    32   28   76   60      1892
D:\\NCI_Trim\\8600\\8638_NCI.lis    0.2789677     0.10277758    32   28   76   60      1892
D:\\NCI_Trim\\8600\\8639_NCI.lis    0.28174788    0.103801854   32   28   76   60      1960
D:\\NCI_Trim\\5400\\5497_NCI.lis    0.28427857    0.093512684   30   25   76   55      1632
...(more comparison results of other 163647 molecules)
```

The following is the first seven rows of the output file generated by the application when using molecule 4,4-bis(4-(((4-(phenyldiazenyl)anilino)carbonyl)oxy)phenyl)pentanoic acid ( $C_{43}H_{36}N_6O_6$ ) to search the entire sub-NCI data set (the fifth row in Table 4.2):

206308\_NCI.lis: (91atoms, 164triangles)

structure	score	rmsd	size	same	all	triangles	matched
D:\\NCI_Trim\\206300\\206308_NCI.lis	1.7705688E-7	1.7705686E-7	91	91	91	164	1001
D:\\NCI_Trim\\69500\\69547_NCI.lis	4.555083	0.90100545	30	18	91	51	327
D:\\NCI_Trim\\92800\\92804_NCI.lis	4.9094367	0.8631977	30	16	91	53	271
D:\\NCI_Trim\\157300\\157385_NCI.lis	4.940801	0.8144177	33	15	91	59	261
D:\\NCI_Trim\\7100\\7126_NCI.lis	5.049523	1.49821	52	27	91	96	524
D:\\NCI_Trim\\249800\\249872_NCI.lis	5.0761538	1.227202	48	22	91	89	291

...(more comparison results of other 163647 molecules)

# Appendix D

## An example of the input structure file used by the algorithm

The following is the content of the structure file of molecule 2-methylbenzo-1,4-quinone ( $C_7H_6O_2$ ) used as the input of the algorithm.

!	Z	X	Y	Z	Atom	Atype
	8	-2.24230000	1.04180000	0.00180000	1	O(4,-1)
	8	2.75340000	-0.55940000	0.00110000	2	O(4,-1)
	6	-1.08580000	0.67110000	0.00190000	3	C(30,-1)
	6	-0.77300000	-0.77250000	-0.00020000	4	C(24,-1)
	6	0.00790000	1.66390000	-0.00180000	5	C(19,-1)
	6	0.50320000	-1.18150000	-0.00050000	6	C(19,-1)
	6	1.28410000	1.25480000	-0.00210000	7	C(19,-1)
	6	1.59700000	-0.18880000	0.00140000	8	C(30,-1)
	6	-1.88880000	-1.78530000	-0.00160000	9	C(10,0)
	1	-0.22080000	2.71930000	-0.00470000	10	H(4,0)
	1	0.73190000	-2.23700000	0.00200000	11	H(4,0)

1	2.08380000	1.98070000	-0.00080000	12	H(4,0)
1	-2.15810000	-2.03070000	1.02570000	13	H(4,0)
1	-2.75590000	-1.36990000	-0.51510000	14	H(4,0)
1	-1.55970000	-2.68790000	-0.51650000	15	H(4,0)

Atom to Atom(s) (Bond part of a ring)

```

1 : 3 ()
O(4,-1) : =C(30,-1)
2 : 8 ()
O(4,-1) : =C(30,-1)
3 : 4, 5 ()
C(30,-1) : -C(24,-1) , -C(19,-1)
4 : 6, 9 ()
C(24,-1) : =C(19,-1) , -C(10,0)
5 : 7, 10 ()
C(19,-1) : =C(19,-1) , -H(4,0)
6 : 8, 11 ()
C(19,-1) : -C(30,-1) , -H(4,0)
7 : 8, 12 ()
C(19,-1) : -C(30,-1) , -H(4,0)
9 : 13, 14, 15 ()
C(10,0) : -H(4,0) , -H(4,0) , -H(4,0)

```

# Appendix E

## Detailed test results of 100 randomly selected pairs in Figure 5.2

The test results of 100 randomly selected pairs of structures in Figure 5.2 are listed in this appendix. The results are text files generated by the application for the test. The results are listed in a tab-delimited table with ten columns. Each row represents comparison result of two molecules. Column “structure1” lists the filename of the first molecule of the comparison molecules. Column “n1” lists the number of atoms in the first molecule. Column “t1” lists the number of atom triplets found in the first molecule. Column “structure2” lists the filename of the second molecule. Column “n2” lists the number of atoms in the second molecule. Column “t2” lists the number of atom triplets found in the second molecule. Column “k” lists the number of matched triplets between the comparison molecules. Column “same” lists the number of identical atoms between the comparison molecules. Column “all” lists the number of the larger of the number of atoms in the comparison molecules. Column “runtime”

lists the run-time of the comparison molecules in seconds.

structure1	n1	t1	structure2	n2	t2	k	same	all	runtime
49834_NCI.lis	21	27	63912_NCI.lis	22	39	0	2	22	0.00241973
155648_NCI.lis	20	30	141829_NCI.lis	39	67	0	3	39	0.00385796
160854_NCI.lis	20	31	141650_NCI.lis	58	103	12	2	58	0.013389391
9103_NCI.lis	22	37	53903_NCI.lis	82	133	42	2	82	0.077520497
1632_NCI.lis	21	36	95189_NCI.lis	99	192	36	5	99	0.080246914
77389_NCI.lis	20	30	254562_NCI.lis	118	231	6	1	118	0.021853735
34963_NCI.lis	19	25	35396_NCI.lis	138	260	2	1	138	0.01572255
23736_NCI.lis	18	31	224460_NCI.lis	162	310	0	2	162	0.017691119
12318_NCI.lis	20	32	242358_NCI.lis	178	314	412	6	178	2.538916275
258243_NCI.lis	22	30	250869_NCI.lis	201	386	92	5	201	0.651304175
155314_NCI.lis	41	72	165142_NCI.lis	22	38	120	7	41	0.091047856
184316_NCI.lis	40	67	99846_NCI.lis	38	65	20	11	40	0.020481113
23406_NCI.lis	38	68	192414_NCI.lis	61	111	232	17	61	0.389402496
196698_NCI.lis	39	67	252964_NCI.lis	78	152	344	20	78	0.810005192
112813_NCI.lis	39	73	220211_NCI.lis	99	175	440	9	99	1.524352629
241174_NCI.lis	38	69	156948_NCI.lis	121	232	92	10	121	0.531224458
226946_NCI.lis	41	75	255693_NCI.lis	141	277	96	10	141	0.781205008
186916_NCI.lis	39	76	243642_NCI.lis	160	307	78	11	160	0.658129219
96406_NCI.lis	41	68	257742_NCI.lis	178	325	217	6	178	1.9656126
184537_NCI.lis	41	80	252207_NCI.lis	198	384	1152	8	198	13.3068853
12943_NCI.lis	59	107	49975_NCI.lis	20	31	52	4	59	0.092455639
30141_NCI.lis	60	110	59008_NCI.lis	39	69	450	19	60	0.782405015
127961_NCI.lis	59	110	106075_NCI.lis	59	101	400	21	59	0.886605683
133074_NCI.lis	58	97	12814_NCI.lis	81	161	176	17	81	0.637654088
208473_NCI.lis	58	102	60305_NCI.lis	98	186	98	17	98	0.491031719
237670_NCI.lis	61	106	168759_NCI.lis	120	217	182	22	120	1.620181814
84591_NCI.lis	59	101	253754_NCI.lis	141	252	162	10	141	1.838211783
248068_NCI.lis	62	113	43696_NCI.lis	162	291	168	20	162	2.4492157
128559_NCI.lis	60	108	226391_NCI.lis	178	328	872	16	178	12.3084789
188284_NCI.lis	62	110	226394_NCI.lis	198	379	768	13	198	13.4940865
100590_NCI.lis	79	151	202180_NCI.lis	19	30	0	1	79	0.01010622
118325_NCI.lis	79	148	239405_NCI.lis	42	85	654	13	79	2.44609568
253249_NCI.lis	80	153	149364_NCI.lis	62	110	640	26	80	2.546716325
216607_NCI.lis	79	149	248249_NCI.lis	81	144	228	21	81	0.950187909
252992_NCI.lis	80	161	188798_NCI.lis	99	187	660	27	99	3.6660235
126859_NCI.lis	78	148	63489_NCI.lis	122	234	7978	39	122	76.9240931

179370_NCI.lis	82	144	252409_NCI.lis	141	267	814	26	141	10.1556651
34084_NCI.lis	82	170	126271_NCI.lis	159	272	728	29	159	13.728088
74925_NCI.lis	80	150	132146_NCI.lis	181	350	92	32	181	2.06857326
238987_NCI.lis	78	146	244690_NCI.lis	202	392	8380	33	202	203.8777069
12747_NCI.lis	98	172	201661_NCI.lis	18	30	42	3	98	0.262001679
145773_NCI.lis	102	194	83063_NCI.lis	42	76	558	20	102	3.8220245
237893_NCI.lis	100	184	242212_NCI.lis	58	106	144	17	100	1.06704684
242802_NCI.lis	102	192	184787_NCI.lis	81	173	1682	34	102	9.94506375
80313_NCI.lis	99	173	1243_NCI.lis	101	187	82	32	101	0.75660485
259784_NCI.lis	101	182	201649_NCI.lis	120	234	504	37	120	6.15423945
196566_NCI.lis	98	192	220422_NCI.lis	140	259	1057	32	140	14.3832922
194639_NCI.lis	98	182	64965_NCI.lis	160	274	590	21	160	11.6220745
30822_NCI.lis	100	191	250831_NCI.lis	180	346	6880	54	180	162.3190405
148669_NCI.lis	100	195	45657_NCI.lis	206	402	30448	91	206	913.8070577
65006_NCI.lis	120	198	203909_NCI.lis	18	28	0	3	120	0.01918633
215213_NCI.lis	118	208	247205_NCI.lis	41	81	320	9	118	3.489222367
1244_NCI.lis	119	223	160502_NCI.lis	59	114	336	20	119	3.754424067
206482_NCI.lis	118	219	13816_NCI.lis	78	151	210	26	118	2.11849358
220502_NCI.lis	121	212	113644_NCI.lis	98	174	124	24	121	1.34940865
100607_NCI.lis	119	234	254396_NCI.lis	122	219	1468	30	122	16.9261085
238054_NCI.lis	121	230	148678_NCI.lis	138	258	816	38	138	11.3100725
229522_NCI.lis	122	227	248049_NCI.lis	159	298	2669	41	159	47.4867044
70998_NCI.lis	122	220	153537_NCI.lis	180	324	0	28	180	0.108697471
48485_NCI.lis	120	230	243059_NCI.lis	207	388	960	50	207	31.980205
13818_NCI.lis	141	262	234913_NCI.lis	20	38	26	7	141	0.438837596
248925_NCI.lis	138	262	249957_NCI.lis	39	67	636	16	138	10.2804659
220420_NCI.lis	138	258	43548_NCI.lis	61	112	384	21	138	6.18543965
147968_NCI.lis	140	254	255862_NCI.lis	78	138	102	29	140	1.69261085
254070_NCI.lis	142	259	45655_NCI.lis	99	193	1232	22	142	21.7465394
242395_NCI.lis	138	248	228639_NCI.lis	121	226	0	29	138	0.072388953
65722_NCI.lis	139	264	229518_NCI.lis	142	263	1090	45	142	17.9869153
254072_NCI.lis	139	253	216857_NCI.lis	159	295	590	39	159	11.5284739
96880_NCI.lis	138	260	250809_NCI.lis	181	345	1572	44	181	40.56026
245131_NCI.lis	141	265	223229_NCI.lis	207	377	232	48	207	9.43026045
244654_NCI.lis	158	274	241951_NCI.lis	18	29	116	0	158	2.577916525
175274_NCI.lis	162	288	84119_NCI.lis	38	72	144	8	162	3.629623267
223231_NCI.lis	162	296	255418_NCI.lis	58	106	1	17	162	0.060065056
170387_NCI.lis	160	309	121948_NCI.lis	78	147	9493	53	160	230.4914775
256128_NCI.lis	158	301	251111_NCI.lis	98	177	843	27	158	17.3941115

237503_NCI.lis	158	308	222854_NCI.lis	119	238	1728	45	158	32.9318111
252486_NCI.lis	162	313	53468_NCI.lis	140	266	2074	50	162	47.5491048
258320_NCI.lis	160	294	254074_NCI.lis	157	293	1000	25	160	19.8745274
111304_NCI.lis	157	297	160847_NCI.lis	177	336	23542	99	177	661.6314412
72377_NCI.lis	163	312	180885_NCI.lis	208	392	2176	68	208	77.6260976
250833_NCI.lis	177	345	93709_NCI.lis	22	38	54	5	177	1.631324743
49544_NCI.lis	176	340	226101_NCI.lis	41	75	48	10	176	1.548867071
250834_NCI.lis	175	340	148786_NCI.lis	61	113	4704	30	175	145.2213309
16882_NCI.lis	175	343	241653_NCI.lis	78	141	162	24	175	5.31183405
230493_NCI.lis	175	325	24230_NCI.lis	98	168	1012	28	175	29.5777896
256682_NCI.lis	175	340	248928_NCI.lis	120	217	120	32	175	3.614023167
220534_NCI.lis	174	332	11698_NCI.lis	142	266	3600	61	174	79.3265085
150270_NCI.lis	186	312	782_NCI.lis	163	312	972	39	186	28.2673812
252544_NCI.lis	186	354	252544_NCI.lis	186	354	2970	186	186	71.6668594
253516_NCI.lis	186	351	242397_NCI.lis	210	392	1400	40	210	60.4659876
220419_NCI.lis	190	423	45525_NCI.lis	20	38	0	5	190	0.060158469
226396_NCI.lis	210	400	234402_NCI.lis	40	69	434	4	210	22.1677421
223232_NCI.lis	189	345	90985_NCI.lis	61	105	186	17	189	7.1448458
250839_NCI.lis	189	362	16268_NCI.lis	81	171	1991	42	189	78.8897057
72378_NCI.lis	187	360	74842_NCI.lis	98	172	960	39	187	35.6150283
249991_NCI.lis	220	411	235419_NCI.lis	119	227	1991	35	220	118.3111584
15216_NCI.lis	221	407	252407_NCI.lis	142	272	924	53	221	49.8423195
48509_NCI.lis	224	414	75501_NCI.lis	163	312	33048	72	224	1565.111233
223233_NCI.lis	225	409	252542_NCI.lis	174	333	2478	53	225	135.2216668
195529_NCI.lis	174	327	195529_NCI.lis	174	327	31167	174	174	672.2707094



# Bibliography

- [1] Guy W. Bemis; Irwin D. Kuntz. "A fast and efficient method for 2D and 3D molecular shape descriptor description". *J. Comput.-Aided Mol. Des.*, 1992, 6, 607-628.
- [2] Andrew C. Good; Todd J. A Ewing; Daniel A. Gschwend; Irwin D. Kuntz. "New molecular shape descriptors: Application in database screening". *J. Comput.-Aided Mol. Des.*, 1995, 9, 1-12.
- [3] Randy J. Zauhar; Guillermo Moyna; LiFeng Tian; ZhiJian Li; William J. Welsh. "Shape signatures: a new approach to computer-aided ligand and receptor-based drug design". *J. Med. Chem.*, 2003, 46, 5674-5690.
- [4] Randy J. Zauhar. "SMART: a solvent-accessible triangulated surface generator for molecular graphics and boundary element applications". *J. Comput.-Aided Mol. Des.*, 1995, 9, 149-159.
- [5] Pedro J. Ballester; W. Graham Richards. "Ultrafast shape recognition to search compound databases for similar molecular shapes". *J. Comput. Chem.*, 2007, 3, 1711-1723.
- [6] J. W. M. Nissink; M. L. Verdonk; J. Kroon; T. Mietzner; G. Klebe. "Superposition of molecules: Electron density fitting by application of Fourier transforms". *J. Comput. Chem.*, 1997, 18, 638-645.

- [7] Paul L. A. Popelier. “Developing Quantum Topological Molecular Similarity (QTMS)”. *Quantum Biochemistry: Electronic Structure and Biological Activity*, 2010, 669-691.
- [8] Xavier Gironés; David Robert; Ramon Carbó-Dorca. “TGSA: A molecular superposition program based on topo-geometrical considerations”. *J. Comput. Chem.*, 2001, 22, 255-263.
- [9] Charles F. F. Karney. “Quaternions in molecular modeling”. *J. Mol. Graphics Modell.*, 2007, 25, 595-604.
- [10] Zong Jie Liu; Roland Van Rapenbusch. “A fast, direct algorithm for the least-squares fitting of two sets of atomic coordinates of macromolecular structures”. *J. Comp. Chem.*, 1988, 9, 596-599.
- [11] Arthur M. Lesk. “Least squares fitting of two structures”. *J. Mol. Biol.*, 1979, 128, 74-77.
- [12] José Manuel Vásquez-Pérez; Gabriel Ulises Gamboa Martínez; Andreas M. Köster; Patrizia Calaminici. “The discovery of unexpected isomers in sodium heptamers by Born-Oppenheimer molecular dynamics”. *J. Chem. Phys.*, 2009, 131, 124126.
- [13] Mark Sinclair Staveley. “Data representation scheme and similarity measures for a comprehensive computational chemistry database”. *PhD thesis*, 2009, Memorial University of Newfoundland.
- [14] Jmol: an open-source Java viewer for chemical structures in 3D.  
<http://www.jmol.org/>

- [15] H. W. Kuhm. "The Hungarian method for the assignment problem". *Naval Res. Logistics Quart.*, 1955, 2, 83-97.
- [16] James Munkres. "Algorithms for the assignment and transportation problems". *J. Soc. Ind. Appl. Math.*, 1957, 5, 32-38.
- [17] J. M. C. Marques; J. L. Llanio-Trujillo; P. E. Abreu; F. B. Pereira. "How different are two chemical structures". *J. Chem. Inf. Model.*, 2010, 50, 2129-2140.
- [18] Gimbal Lock, [http://en.wikipedia.org/wiki/Gimbal\\_lock](http://en.wikipedia.org/wiki/Gimbal_lock).
- [19] William Rowan Hamilton. "On a new species of imaginary quantities connected with a theory of quaternions". *Proc. R. Irish Acad.*, 1844, 2, 424-434.
- [20] William Rowan Hamilton. "On quaternions". *Proc. R. Irish Acad.*, 1847, 3, 1-16.
- [21] S. K. Kearsley. "On the orthogonal transformation used for structural comparisons". *Acta Crystallogr. Sect. A.*, 1989, 45, 208-210.
- [22] Shoemake Ken. "Animating rotation with quaternion curves". *Computer Graphics*, 1985, 19(3), 245-254.
- [23] National Cancer Institute (NCI) database. <http://cactus.nci.nih.gov/ncidb2.2/>.
- [24] Ahmad I. Alrawashdeh; Mansour H. Almatarneh; Raymond A. Poirier. "Computational study on deamination reaction of adenine with  $OH^-/nH_2O$  ( $n = 0, 1, 2, 3$ ) and  $3H_2O$ ". *Can. J. Chem.*, 2013, 91, 1-9.
- [25] Mohammad A. Halim; Mansour H. Almatarneh; Raymond A. Poirier. "Mechanistic Study of the Deamidation Reaction of Glutamine: A Computational Approach". *J. Phys. Chem. B*, 2014, 118(9), 2316-2330.

- [26] Pei-Ying Liu; Yong-Jin Wu; Cory C. Pye; Paul D. Thornton; Raymond A. Poirier; D. Jean Burnell. "Facial Selectivity in the Diels-Alder Reactions of 2,2-Disubstituted Cyclopent-4-ene-1,3-dione Derivatives and a Computational Examination of the Facial Selectivity of the Diels-Alder Reactions of Structurally Related Dienes and Dienophiles". *Eur. J. Org. Chem.*, 2012, 1186-1194.
- [27] Darren R. Flower. "Rotational superposition: A review of methods". *J. Mol. Graphics Modell.*, 1999, 17, 238-244.
- [28] Michael Sipser. "Introduction to the Theory of Computation, Third Edition". *Boston, MA, USA: Cengage Learning*, 2013, 276-277.
- [29] Andreas Hermann; Matthias Lein; Peter Schwerdtfeger. "The Search for the Species with the Highest Coordination Number". *Angew. Chem.*, 2007, 119, 2496-2499.