A COMPUTATIONAL APPROACH TO CHEMICAL STRUCTURE MATCHING BASED ON CARTESIAN ATOMIC COORDINATES

by Mykola Zotko

Master Thesis

Supervised by: Prof. Dr. Max C. Holthausen

Institute of Inorganic and Analytic Chemistry

Goethe University Frankfurt am Main

29 July 2018

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Masterarbeit mit dem Titel

“A Computational Approach to Chemical Structure Matching Based on Cartesian Atomic Coordinates ”

selbstständig angefertigt und mich keiner anderen Hilfsmittel als den darin angegebenen bedient habe, insbesondere, dass schriftliche Entlehnungen nicht stattgefunden haben, soweit sie in der Arbeit nicht ausdrücklich als solche mit Angabe der entsprechenden Schrift bezeichnet sind. Die Arbeit wurde bisher in keinem Studiengang als Prüfungsleistung verwendet.

Frankfurt am Main, den 29. Juli 2018

_____

Mykola Zotko

Acknowledgements

I would like to first say a very big thank you to my supervisor Prof. Dr. Max C. Holthausen for the interesting research topic, the patient guidance and advice he has provided. I am particularly grateful for the assistance given by Dr. Mathias Lein of the School of the Chemical and Physical Sciences Victoria at the University of Wellington. I would also like to thank all the members of the Holthausen working group who helped me in my supervisor's absence. I am also grateful to the scholarship received from the Goethe University Frankfurt am Main and making it possible for me to concentrate totally on the work on this thesis. Finally, I wish to thank my family for their support and encouragement throughout my study.

Zusammenfassung

Eine chemische Reaktion ist ein Vorgang, bei dem ein Satz von Molekülen (Edukte) in einen anderen Satz (Produkte) umgewandelt wird. Während der Reaktion ändern Atome ihre relative Position zueinander und Bindungen werden gebildet oder gebrochen. Die Identifizierung von strukturellen Ähnlichkeiten und die automatische Zuordnung von Atomen ist ein nicht-triviales Problem der *computational molecular sciences*, welches als "atom mapping problem" bekannt ist. Aufgrund der potenziell großen Anzahl der an Bindungstransformationen beteiligten Atome und der gegebenfalls komplexen Stereochemie von Molekülen ist eine manuelle Durchführung des *atom mappings* ein mühsamer, zeitaufwändiger und fehlerträchtiger Prozess. Es ist daher von großer Bedeutung zuverlässige Algorithmen für das *atom-atom mapping* zwischen Edukt- und Produktstrukturen zu entwickeln. Zur Lösung dieses Problems existieren bereits seit 1960[1,2] zahlreiche Verfahren, wie beispielsweise *maximal clique* oder *backtracking* basierte Methoden, die allerdings individuelle Schwächen und Beschränkungen aufweisen. Ziel dieser Masterarbeit war es, geeignete Algorithmen für *atom mapping* in chemischen Reaktionen zu finden, sie zu bewerten und zu implementieren. Die drei Hauptanforderungen an die Entwicklung von atom mapping Algorithmen sind: (i) die Fähigkeit alle Atome zweier Strukturen einander zuzuordnen, (ii) das Erkennen von an der Reaktion beteiligten Atomen unter Berücksichtigung der Stereochemie und (iii) die Identifizierung einer effizienten Rechenmethode auch für moderat große Moleküle (bis zu 100 Atome).

Wir haben zwei Algorithmen als am besten geeignet für die Lösung der oben aufgeführten Aufgaben ausgewählt: den *canonical labeling for clique approximation* (CLCA) Algorithmus und den A* *atom mapping* Algorithmus. Der CLCA-Algorithmus basiert auf der *extended-connectivity* Methode, die mittels Primfaktorzerlegung Informationen über die Position eines Atoms im gesamten topologischen Raum eines Moleküls kodiert. Der A* *atom mapping* Algorithmus basiert auf der populären A*-Suchtechnik, wie sie auch in der künstlichen Intelligenz verwendet wird. A* verwendet eine intelligente Heuristik, die durch den Raum des *atom mappings* führt, um optimale Ergebnisse zu finden. CLCA und A* Algorithmen sind komplementär und können in einem Programmierskript verwendet werden. Der Hauptvorteil von CLCA ist die schnelle Identifizierung von ausge-

dehnten substrukturen zweier Moleküle. Das *partial mapping*, das mit CLCA berechnet wird, verringert die Rechenzeit und liefert den Startpunkt für den A*-Algorithmus. Für beide Algorithmen ist die die Stereochemie schwierig zu handhaben. Daher müssen die möglichen Zuordnungen visuell überprüft werden.

Contents

1. Introduction

Molecules with similar structures often tend to have similar properties and functions[3]. For example, the function of a newly discovered RNA molecule can be estimated by comparing its similarity to known RNA species.[4] Molecular similarity plays an essential role in the analysis of molecular data sets[5], query retrieval in molecular databases[6], molecular modelling[7] and the study of reaction mechanisms[8]. Comparing similarities of different chemical compounds and matching of corresponding atoms are very challenging tasks in molecular sciences.[9] The most common type of similarity searching is the detection of common structural fragments or substructures that are shared by two molecules.[10] Existing atom matching algorithms can be classified into two broad categories: algorithms for exact matching and algorithms for approximate matching. Exact matching algorithms require a strict bijection between the matched molecules or at least between their substructures. Approximate matching algorithms can be applied to any two molecules even if they have different, to some extent, structure[11], such as conformational isomers or reactants and products in chemical reactions. A chemical reaction is a process in which one set of molecules (reactants) is transformed into another set (products). During the reaction, atoms change their relative positions and bonds are formed or broken. The tracing and identification of corresponding atoms between reactants and products is a non-trivial problem in computational molecular sciences, well known as ”Atom Mapping Problem”.[12] Due to the potentially large number of atoms in the reaction, the rich stereochemistry of molecules and large sets of reactions in databases a manual curation of atom mapping is a laborious and time-consuming process.[13] It is of great importance to have reliable algorithms to establish the atom-atom mapping relationship between reactant and product structures. Numerous methods have been proposed to solve this problem.[13] Popular algorithms are either based on extended-connectivity or maximum common substructure. Most of the novel methods are based on some kind of an optimisation procedure. Their aim is finding optimal mappings with the minimum number of broken and formed bonds.[12] However, many methods have weaknesses and limitations, e.g. hydrogen atoms are ignored[14] or algorithms cannot handle stereochemistry[13]. The goal of this master thesis was to find, evaluate and implement suitable algorithms for atom mapping in chemical reactions.

2. Theoretical background

2.1 The Molecular graph

Chemical structures in computer analysis are often represented as molecular or chemical graphs. The properties of graphs, as well as networks in general, are studied by graph theory. This well-established branch of mathematics has many applications in modern computer sciences. The father of graph theory is the pioneering Swiss mathematician Leonhard Euler (1707-1782). Graph theory is considered to have begun in 1736 when Euler published the paper with his solution of the famous "Seven Bridges of Königsberg" problem.[14]

A graph is an abstract structure that consists of a set of vertices or nodes connected by edges. It does not matter what shape or size the nodes have or how long the edges are. The positions of the nodes do not matter as well. Hence, one graph may be drawn in many different ways. Figure 2.1 shows an example of different representations of the same graph.

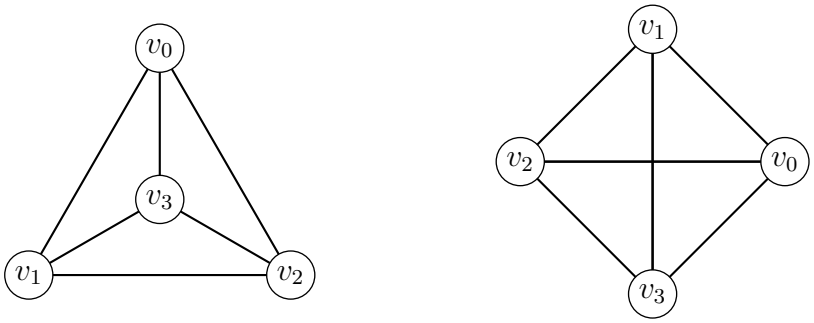


Figure 2.1: Two forms of presentation of one graph.

Mathematically, each graph G is defined as $G = (V, E)$, where V denotes a set of vertices and E denotes a set of edges. If two nodes v_1 and v_2 are connected then $(v_1, v_2) \in E$. There exist two types of graphs: directed and undirected. The edges of a directed graph have a particular direction, indicated by an arrow. In contrast, the edges of undirected graphs are indicated by a line and have no direction.[15] Since chemical bonds in general have no explicit direction, undirected graphs are used for the representation of molecules. The molecular graph is a two-dimensional representation of a molecule, where atoms are represented as nodes (vertices) and bonds as edges.

A graph represents only the topology of a molecular structure, e.g. the way the atoms are connected to each other. Hydrogen atoms are usually excluded from the graph. Nodes and edges of a molecular graph may have associated properties. Each node has an associated element or atomic number of the corresponding atom. Additionally, edges may have associated bond order of corresponding bonds.[16] An example of the molecular graph of 1,3,4-oxadiazole is shown in Figure 2.2.

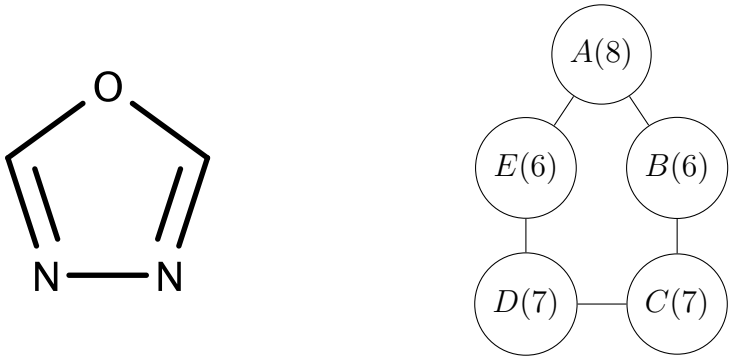


Figure 2.2: Molecular graph of 1,3,4-oxadiazole. Numbers in parentheses represent atomic numbers.

Two molecules are equal if their molecular graphs are isomorphic and the properties of the nodes and edges are equal to the properties of corresponding nodes and edges respectively. Two graphs G_1 and G_2 are called isomorphic if there exists a bijective (one-to-one) correspondence between the nodes of both graphs. Therefore if two nodes in G_1 a connected by an edge, then the corresponding nodes in G_2 must be connected as well.

For a pair of molecules with different sizes, the smaller molecule can be a substructure of the larger one. In this case, there exists a subgraph isomorphism between both molecules. For example, the graph of benzene is a subgraph of toluene graph. Although two graphs may have many common subgraphs, the largest common subgraphs are of the most interest for comparison of chemical structures.[1,17] The largest common subgraph between two graphs is called maximal common subgraph (MCS). Maximal common subgraphs are divided into connected or disconnected subgraphs. In a connected MCS, each node is connected to other nodes by a path through the MCS. A disconnected MCS has two or more disconnected groups of nodes.

An example of connected and disconnected subgraphs is shown in Figure 2.3.[18]

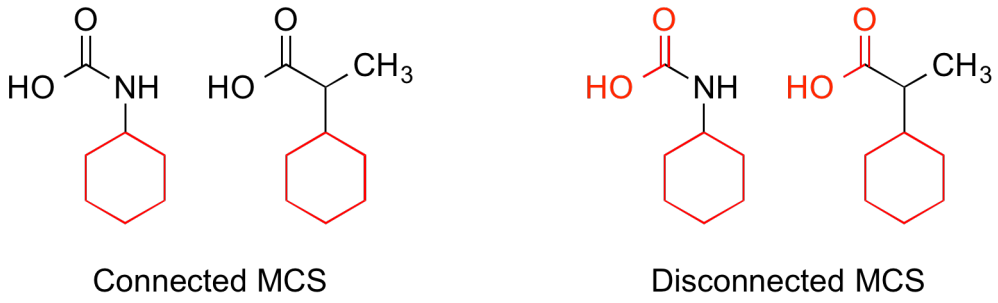


Figure 2.3: Connected versus disconnected maximal common subgraphs .[18]

The comparison and enumeration of the molecular graphs are of great importance and have a long history in computational molecular sciences. According to Akutsu et al. the most essential problems for pattern recognition in molecular graphs are the following: determining whether two graphs are isomorphic, determining whether one molecular graph is a subgraph of another, finding the maximum common subgraph of two graphs, detecting a reaction atom mapping, enumerating of stereoisomers and enumerating of possible molecular graphs satisfying given constraints.[19]

2.2 Molecular graph construction from atomic coordinates

Approach taken in this thesis relies on the most straightforward method for determination of atomic connectivity in molecules, which is based on interatomic distances and atomic covalent radii. In a first step, the distance matrix between all possible atom combinations is created, which is of size $n \times n$, where n is the number of atoms in the molecule. Each element a_{ij} in the distance matrix represents the distance between atoms a_i and a_j in the three-dimensional Cartesian space. Next, each interatomic distance is compared to the sum of covalent radii[20] r_i and r_j of both atoms. If the distance between two atoms is within the range $d = 1.3(r_i + r_j)$, that is the sum of covalent radii plus thirty per cent, then an edge connecting the two nodes is added to the molecular graph.

3. Reaction Mapping

A chemical reaction is the process of transforming reactant molecules into product molecules by breaking or creating chemical bonds. It is essential in the present context to have reliable methods at hand to establish the bijections between the atoms before and after the reaction, known as reaction mapping, as manual curation of atom mapping is time-consuming and error-prone. According to Chen et al. existing methods for reaction mapping can be classified into three main groups: fragment-assembly based methods, common-substructure based methods, and optimisation based methods (Figure 3.1).[12]

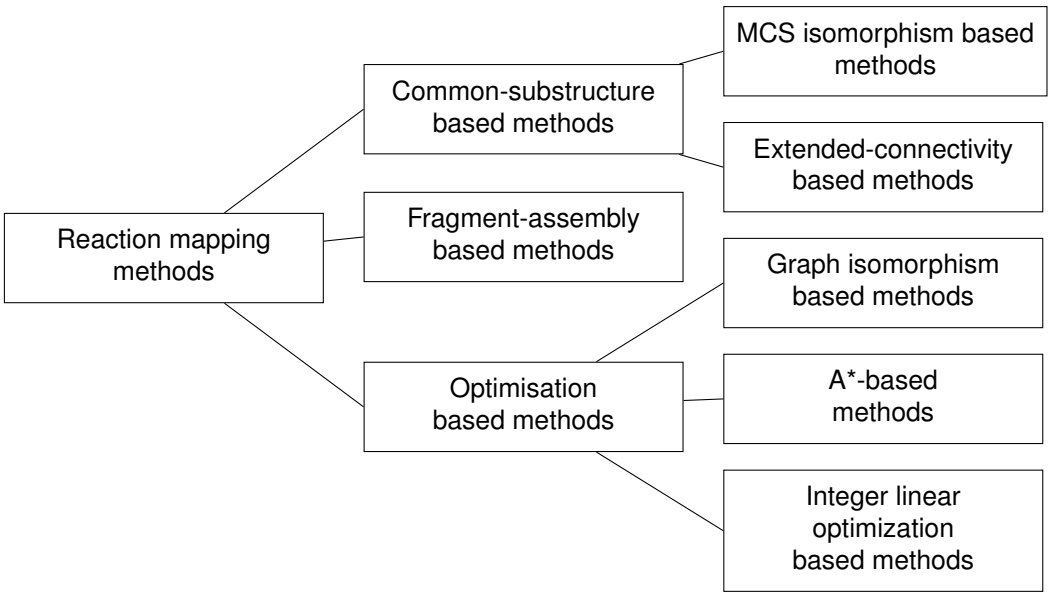


Figure 3.1: Classification of reaction mapping methods.[12]

The common-substructure based methods are classified into the extended-connectivity based methods[21,22] and maximal common subgraph (MCS) isomorphism based methods[23,24,25]. The latter can be classified further into maximal clique-based[26], backtracking[27,28,29] and dynamic programming methods[30,31].[18] The extended-connectivity approach is used in Canonical Labeling Click Approximation (CLCA) algorithm, which will be presented in Chapter 4 of this work. The optimisation-based methods are classified into graph isomorphism[32,33] based methods, integer linear optimisation based methods[34] and A*-based methods. The A*-based atom mapping algorithm will be presented in Chapter 5 of this work.

3.1 Fragment-assembly based methods

The technique based on fragment-assembly was used by Lynch et al.[35,36] in their algorithm for analysing the difference in chemical structures involved in a chemical reaction. The first step in the fragment-assembly method is the breaking of reactants and products down into small fragments. In the next step, the fragments of the reactants are matched with the fragments of the products to identify the identical species on both sides of the reaction, which are then removed in a stoichiometric fashion. Finally, the remaining components are assembled. The structures obtained in this way identify the overall structural changes in chemical reaction. Figure 3.2 illustrates the workflow of the fragment-assembly based algorithm. Due to ambiguities during the fragmentation process, it is obviously not always possible to identify the exact location of the reaction sites in both molecules, which represents the main shortcoming of this method.[12]

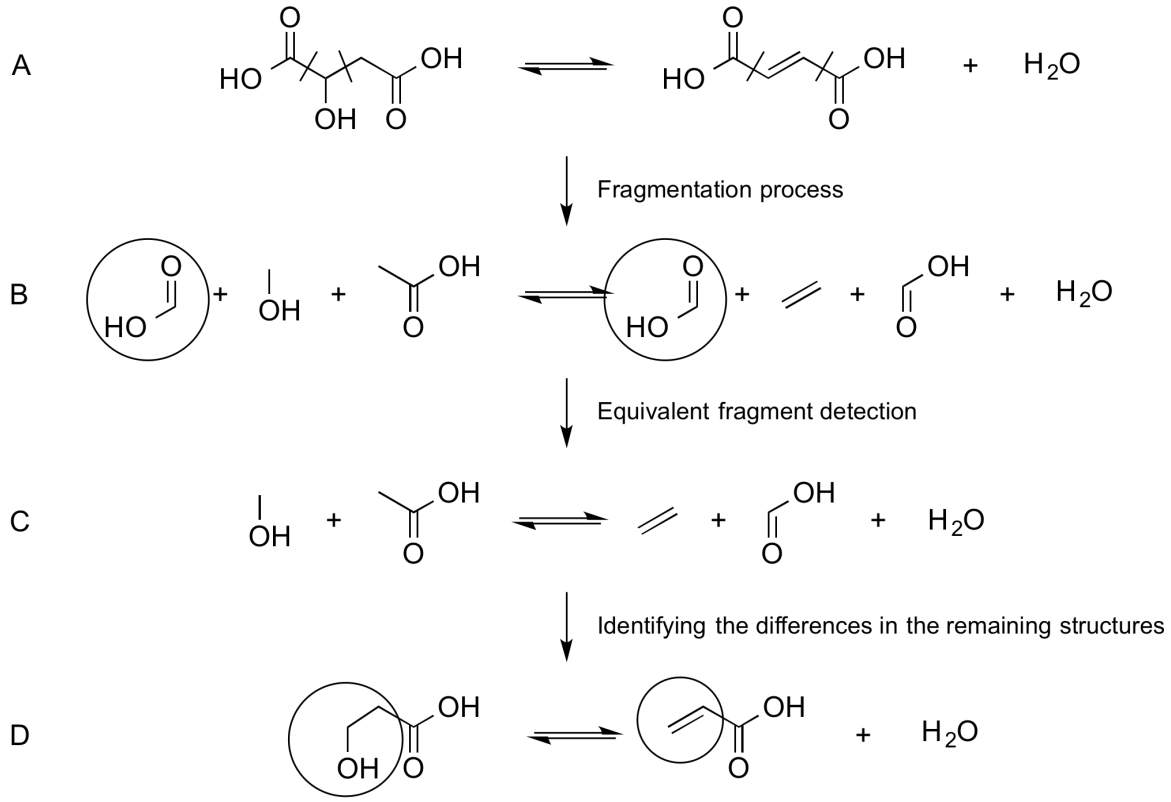


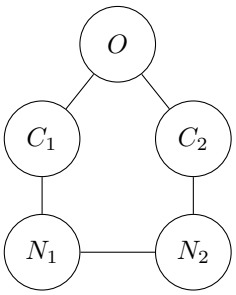
Figure 3.2: Fragment-assembly based algorithm. (A) Fragmentation of molecules. (B) Identification and removing of equal fragments on both sides of the reaction (black circles). (C) Remaining fragments. (D) Fragment assembly to identify the overall structural changes.[37]

3.2 Maximal-clique based methods

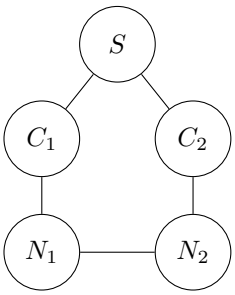
The problem of identification of maximal common substructures between two molecules can be solved by finding the so-called maximal clique in a compatibility graph of both structures. A clique of a graph is a subgraph, in which each node is connected to all other nodes. Consequently, a maximal clique of a graph is the clique with the maximal number of vertices, and graph may have multiple maximal cliques. If all edges ($e_1, e_2, e_3...$) of a graph G are represented as a set E and all vertices ($v_1, v_2, v_3...$) are represented as a set V, a compatibility graph of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is a graph G_c, in which each vertex consists of a combination of vertices from G_1 and G_2. For example, if the graph G_1 has the node v_1 representing a carbon atom and the node v_2, which represents a carbon atom in G_2, both nodes are compatible and can be mapped to each other. Therefore the node in the compatibility graph will get a label v_1v_2. Two nodes v_1v_2 and u_1u_2 in the compatibility graph are adjacent if, and only if v_1 is connected to u_1 in the first graph and v_2 is connected to u_2 in the second graph or, vice versa, if v_1 is not connected to u_1 in G_1 and v_2 is not connected to u_2 in G_2. To summarise, two nodes v_1v_2 and u_1u_2 in the compatibility graph are connected, if:

$$v_1u_1 \in E_1 \; and \; v_2u_2 \in E_2$$
$$or \; v_1u_1 \notin E_1 \; and \; v_2u_2 \notin E_2$$

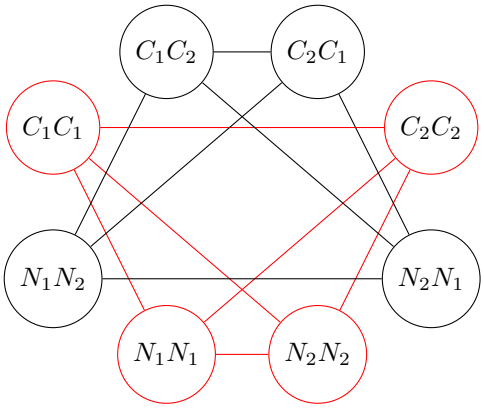
Figure 3.3 illustrates the compatibility graph of molecular graphs of 1,3,4-oxadiazole and 1,3,4-thiadiazole. The compatibility graph has two maximal cliques which can be determined with special algorithms. The most popular algorithms for this task are the Bron-Kerbosch[38] and the Carraghan-Pardalos[39] algorithms. Both methods can identify disconnected MCS, however, most clique-detection algorithms can be modified for the identification of connected MCS. The main shortcoming of clique-detection algorithms is that the computing time increases exponentially with the number of nodes and edges in the compatibility graph.[40,41]



(a) 1,3,4-oxidiazole



(b) 1,3,4-thiadiazole

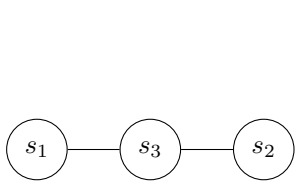


(c) Compatibility graph of (a) and (b)

Figure 3.3: Molecular graphs of 1,3,4-oxidiazole (a) and 1,3,4-thiadiazole (b) and their compatibility graph (c). Two maximal cliques, which represent two possible mappings, are easily identified by visual inspection.

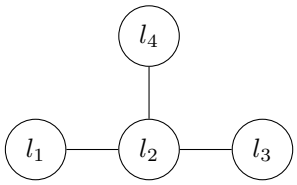
3.3 Backtracking methods: The Ullmann algorithm

The Ullmann algorithm[42] is one of the oldest and most widely applied algorithms[2,43,44] for substructure search in molecular graphs. The algorithm determines subgraph isomorphism rather than MCS. Nevertheless, it is worth mentioning because it was the fastest algorithm at its time and it forms the basis for many subgraph isomorphism algorithms developed later.[45] Subgraph isomorphism algorithms determine if a chemical structure represents a substructure of a larger one. If it is the case, the algorithm generates all possible mappings between both molecules. The workflow of the Ullmann algorithm is illustrated using matrices. In the first step, the so-called adjacency matrices of both structures are created (Figure 3.4).



$$\begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{ccc} s_1 & s_2 & s_3 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array}$$

(a) Propane



$$\begin{array}{c} \\ l_1 \\ l_2 \\ l_3 \\ l_4 \end{array} \begin{array}{c} \begin{array}{cccc} l_1 & l_2 & l_3 & l_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

(b) Isobutane

$$\begin{array}{c} \\ s_1 \\ s_2 \\ s_3 \end{array} \begin{array}{c} \begin{array}{cccc} l_1 & l_2 & l_3 & l_4 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

(c) Initial atom mapping matrix

Figure 3.4: Molecular graphs and adjacency matrices of propane (a) and isobutane (b). (c) Initial atom mapping matrix generated from (a) and (b). The atom s_3 with two neighbours can be mapped only to the atom l_2 with three neighbours.

An adjacency matrix (a and b in Figure 3.4) is a matrix of the size $n \times n$ where n is the number of atoms (nodes) of a chemical structure (graph). Depending on whether atoms a_i and a_j are connected or not, the matrix element a_{ij} is assigned 1 or 0 respectively. In

the second step, an atom mapping matrix (c in Figure 3.4) of size $n_1 \times n_2$ is created, where n_1 and n_2 are the number of atoms of both structures. Depending on whether two atoms a_i and a_j are mapped or not the corresponding matrix element is assigned 1 or 0 respectively. Mapping is accomplished in the following way: all elements of the matrix are initially set to 1, that is, all mapping combinations are allowed. Next, the Ullmann algorithm identifies nodes which cannot be mapped to each other. To this end, the algorithm compares the node connectivities[1] of the smaller structure ($c_{s1}, c_{s2}, c_{s3}...$) with the node connectivities of the larger structure ($c_{l1}, c_{l_2}, c_{l3}...$). Two nodes s_i and l_j cannot be mapped if $c_{si} > c_{lj}$, thus the corresponding elements in the mapping matrix are set to zero. For example, atom s_3 in propane cannot be mapped to atoms l_1, l_3 and l_4 in isobutane. So the atom s_3 can be mapped only to the atom l_2. At this stage the initial matrix from Figure 3.4 (c) is obtained. Next, the algorithm starts to generate all possible mapping permutations starting from the first raw of the mapping matrix (s. Figure 3.5).

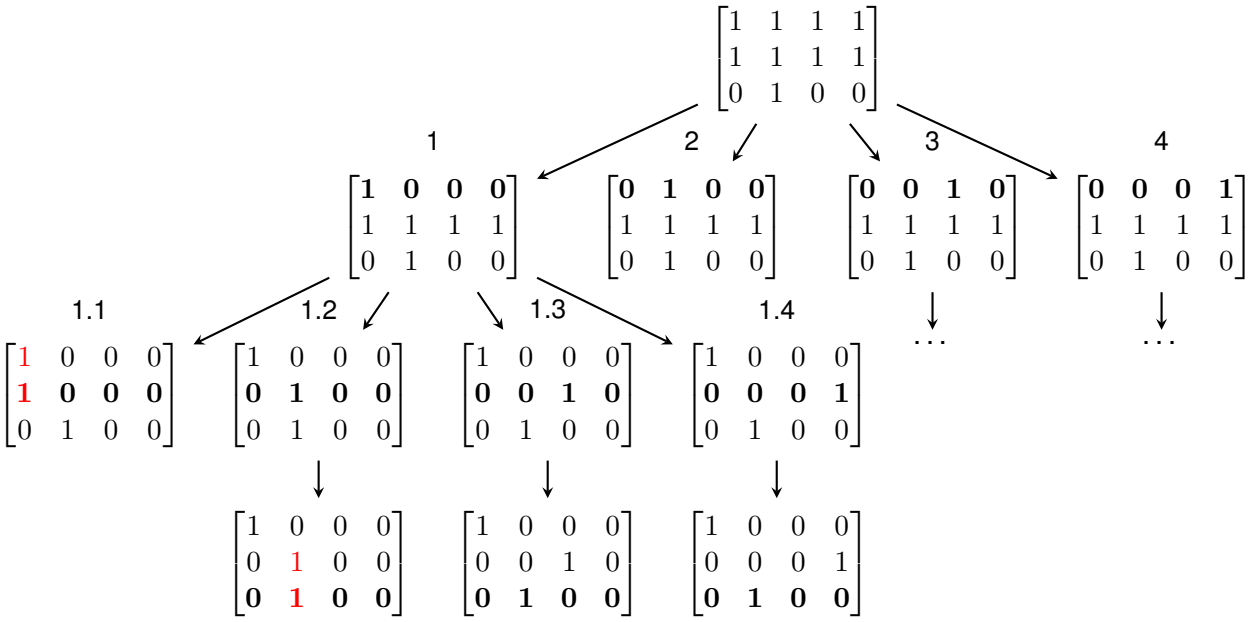


Figure 3.5: Workflow of the Ullmann algorithm. The row under consideration is shown in bold, mapping conflicts in red. Due to space limitations, the matrices 2, 3 and 4 are not expanded.

[1]The connectivity of an atom is the sum of all elements in the corresponding row or column of the adjacency matrix.

In the first step of this example, four possible combinations for a mapping of atom s_1 are created. After expanding of the matrix 1, there are four possibilities for the mapping of the atom s_2. The matrix 1.1 is pruned from the search as atom l_1 is already mapped to atom s_1. After expanding the remaining matrices (1.2, 1.3, 1.4) and moving to the last row, three possible combinations are determined. After removal of one combination where two nodes are mapped to one, the algorithm returns two possible atom mappings for both molecules. Combination 2 will be pruned from the search because atom l_2 is already mapped to atom s_3.

4. The CLCA algorithm

4.1 Base algorithm

The Canonical Labeling for Clique Approximation (CLCA) algorithm is an extended-connectivity algorithm developed by Kumar et al.[46] CLCA uses prime number factorisation to generate canonical labels for bonds and atoms. It is an algorithm for the approximate solution of the maximum clique problem[1], which is a typical NP-complete problem and cannot be solved in polynomial-time[1].[47] CLCA has a polynomial time complexity, but it can identify only unambiguous atom-atom mappings. The key steps of CLCA are summarized in Figure 4.4 later in this subchapter.

The CLCA algorithm works as follows:

(1) First, two chemical structures are converted into molecular graphs.

(2) Second, the algorithm generates a canonical label for each atom in both compounds. The labels are assigned to corresponding nodes as properties. Canonical labels are represented as strings, which encode the information about an atom's location in the full topological space of the molecule. These labels can include different atom properties (if available), such as the atomic numbers of the atom itself and its neighbours, the sign of charge, the absolute charge, stereochemical information, etc. In our implementation, the atom strings are created by concatenation (merging) of an atom's atomic number with the atomic numbers of adjacent atoms. The latter are sorted in ascending order. For example, the carbon atom in formaldehyde, CH_2O, would be associated the string "6118" (Figure 4.1).

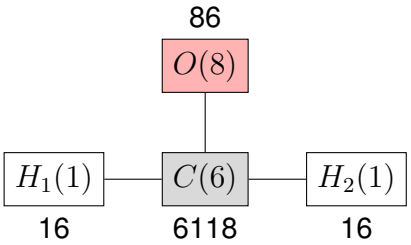


Figure 4.1: Creating of atom strings from atomic numbers (in parentheses) in the molecular graph of formaldehyde.

[1] An algorithm is said to finish in polynomial time if the number of steps required to complete the algorithm is equal to n^k where k is some nonnegative integer, and n is the size of the input.

The first digit ”6” represents the atomic number of carbon. The rest of the string, ”118”, represents the atomic numbers of carbon’s neighbouring atoms, i.e. two hydrogens and one oxygen. The remaining atoms, two hydrogens and oxygen, have the atom strings ”16”, ”16” and ”86” respectively, because they have only carbon as their neighbour.
(3) In the next step, CLCA searches in the string lists of both structures for singular strings, i.e. strings that appear only once. Atoms with singular strings are unique to one structure and cannot be mapped to any atom in the other. Therefore, singular nodes are not considered further as mapping candidates and are assigned a value of ’1’.
(4) Next, the algorithm searches for atom strings which are unique in either structure, but appear in both structures. A pair of such strings is considered a match between two atoms in both molecules. These atoms are marked as mapped and are not considered further. The first pair of mapped atoms is assigned the prime number ”2” and the remaining pairs are assigned next higher prime numbers respectively (the assignment order is irrelevant).
(5) The remaining nodes are divided into groups with identical strings. Each group is assigned subsequent prime numbers in the same fashion.
(6) In the following step, the prime number products (PNP) for atoms that are neither singular nor mapped are calculated. The PNPs for the formaldehyde molecule from the previous example are calculated as follows (Figure 4.2): if, for example, the carbon atom, both hydrogens and oxygen have prime numbers 2, 5, 5, 3 respectively, the PNP of carbon is equal to 150 ($3 \times 5 \times 5 \times 2 = 150$). In this way the PNP propagates the topological information from a node to its neighbors.

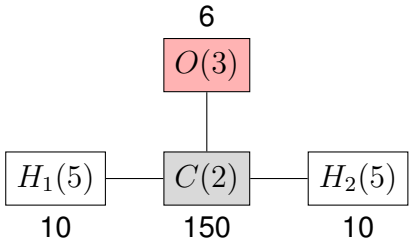


Figure 4.2: Calculation of prime number products (PNP) in the molecular graph of formaldehyde, associated prime numbers for step 5 are given in parentheses.

(7) Subsequently, the PNP is concatenated with the corresponding atom strings from step (2). Consequently, the carbon atom string of formaldehyde will be ”6118150” (”6118” + ”150”, Figure 4.3).

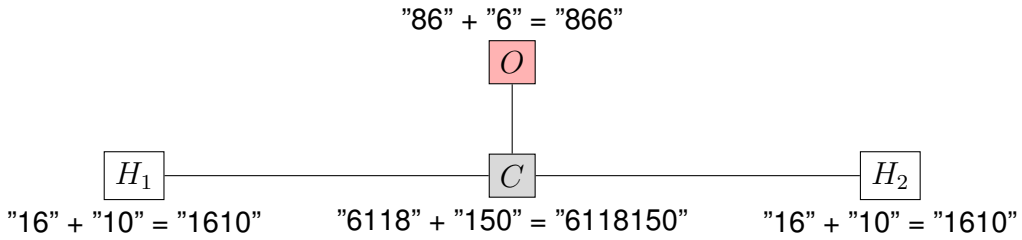


Figure 4.3: Creation of new atom strings with additional topological information in the molecular graph of formaldehyde. Atom strings (AS) from Figure 4.1 concatenated with prime number products (PNP) from figure 4.2

The new string encodes more specific information, e.g. information about the atom itself and its direct and indirect neighbors. In this way, new atom strings are created and the algorithm repeats steps (3), (4) and (5) to identify new mappings. If the highest assigned prime number after the second iteration has not changed, no new information can be retrieved. In this case, the algorithm stops and returns the mappings that have been accumulated thus far. Otherwise, the algorithm proceeds from the step (6).

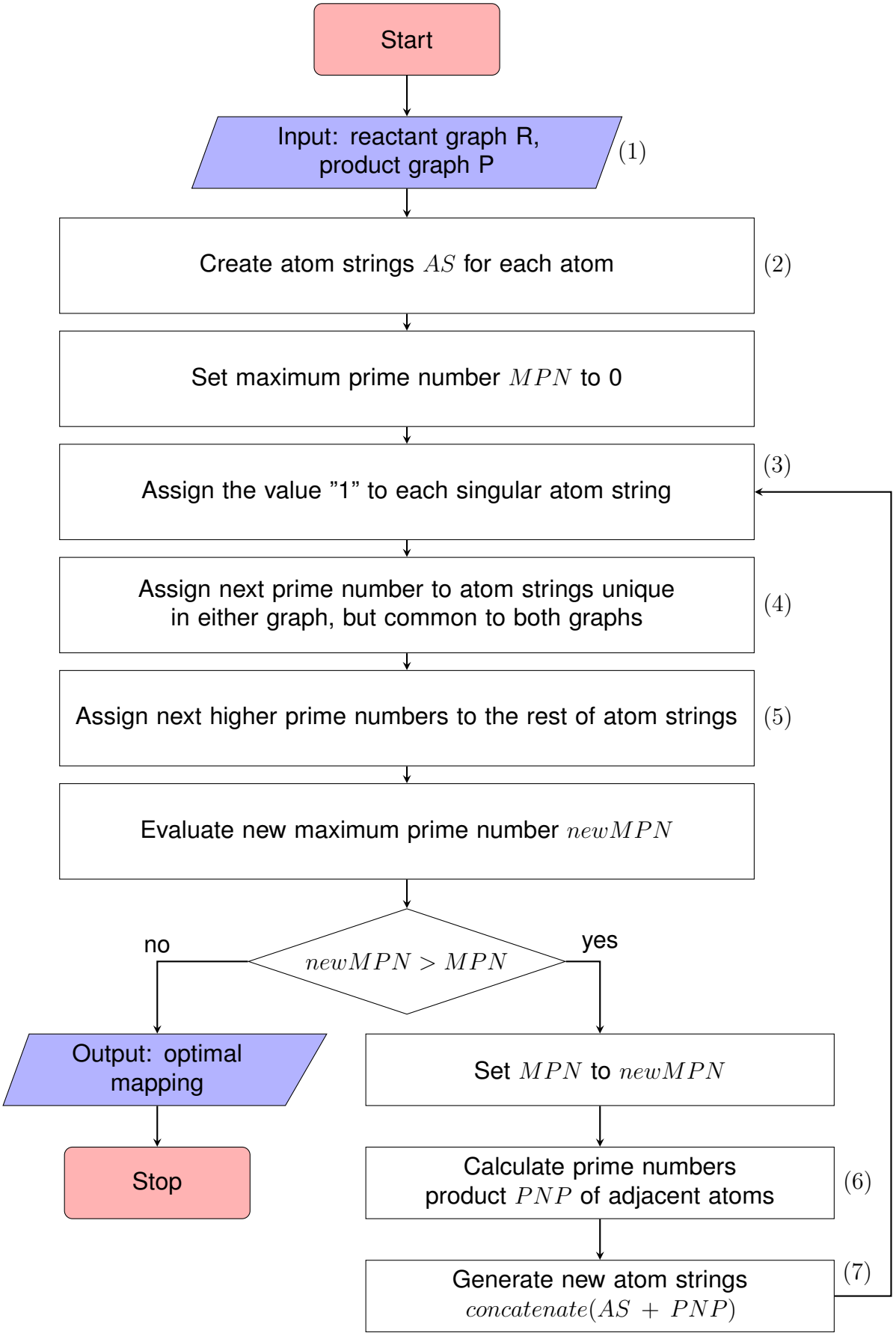


Figure 4.4: Flowchart diagram for the CLCA algorithm. Numbers in parentheses refer to indexes in the text.

The workflow of CLCA is illustrated for a simple example with butyraldehyde and butyric acid (Figure 4.5, Figure 4.6 and Table 4.1).

Figure 4.5: Chemical structure of butyraldehyde (left) and butyric acid (right).

As can be seen, the atoms C_1 and O_1 in the aldehyde graph and C_1, O_1, O_2 in the acid graph cannot be mapped. Although both carbon atoms appear only once in both structures, they have different connectivity patterns and thus, cannot be mapped by the algorithm. Additionally, there are several options for the mapping of oxygen atoms. The latter are redundant species, and CLCA cannot find an unambiguous mapping for these nodes.

Table 4.1: Atom mapping of butyraldehyde and butyric acid with the CLCA algorithm.

ID	Mapping	String	PN_{1*}	PNP_1	CAS_1	PN_2	PNP_2	CAS_2	PN_3
O_1		86	3	3	863	7	7	867	7
C_1	S	668	1	1	1	1	1	1	1
C_2	(C_2)	666	5	25	66625	3	3	3	3
C_3	M (C_3)	666	5	50	66650	5	5	5	5
C_4	M (C_4)	66	2	2	2	2	2	2	2
ID	Mapping	String	PN_1	PNP_1	CAS_1	PN_2	PNP_2	CAS_2	PN_3
O_1		86	3	3	863	7	7	867	7
O_2		86	3	3	863	7	7	867	7
C_1	S	6688	1	1	1	1	1	1	1
C_2	M (C_2)	666	5	25	66625	3	3	3	3
C_3	M (C_3)	666	5	50	66650	5	5	5	5
C_4	M (C_4)	66	2	2	2	2	2	2	2

*The subscripts in the table heading indicate the loop number in the CLCA algorithm.

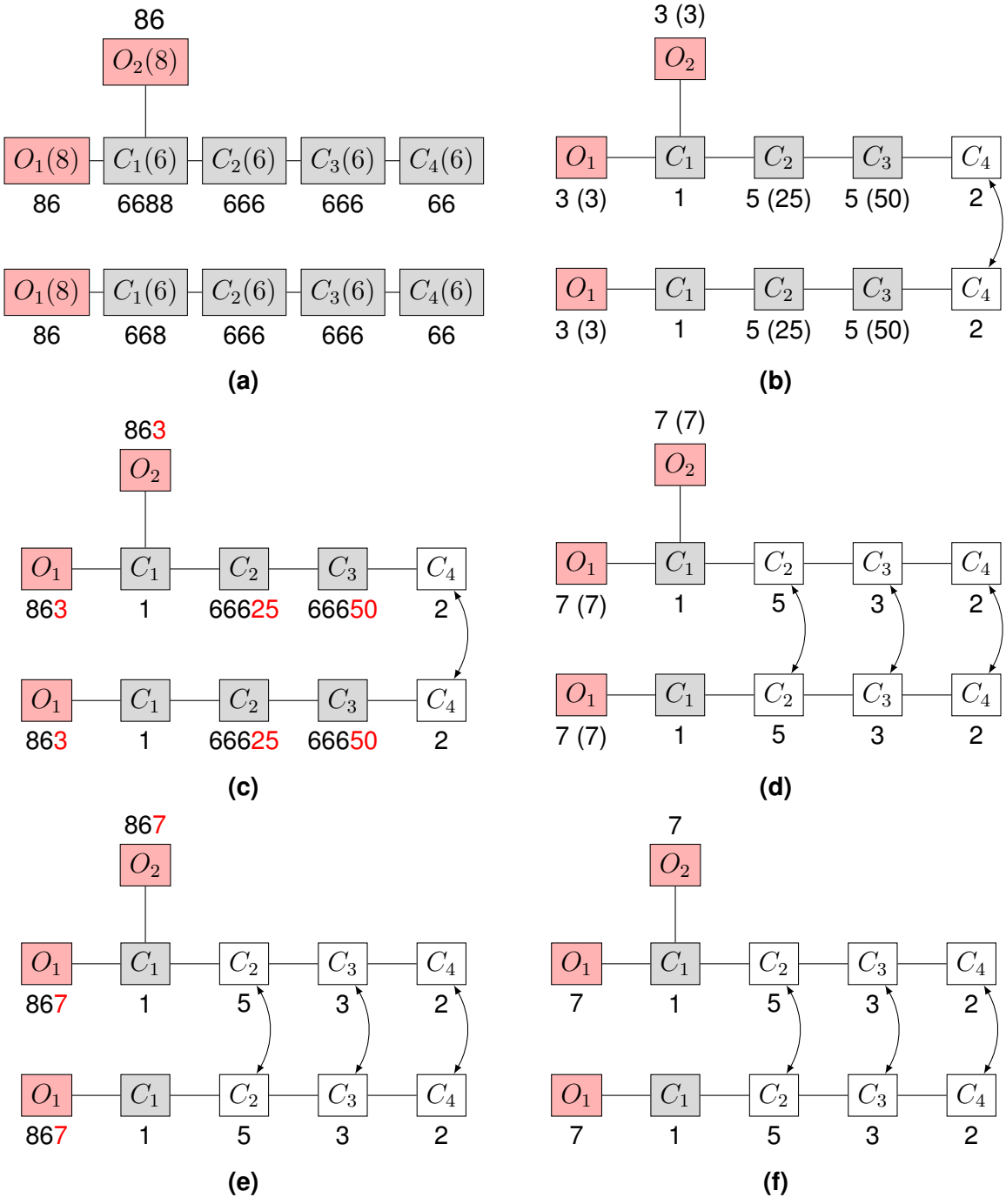


Figure 4.6: MCS search of butyraldehyde and butyric acid with help of CLCA. (a) The atom strings are created. (b) Singular atom strings are assigned the value ”1”. Atom strings which are unique in either graph, but common to both graphs are assigned the first prime number 2. C_4 is mapped to C_4. Next, prime numbers 3 and 5 are assigned to the remaining string groups. Values in parentheses are the prime number products PNP of adjacent atoms. (c) New atom strings generated by concatenation of strings from step (a) and $PNPs$ (in red) from step (b). (d) The steps from (b) are repeated. Two new mappings are identified. (e) New concatenated atom strings are generated. (f) The algorithm stops, because the new maximum prime number 7 has not changed from step (d).

4.2 Extended algorithm

CLCA is a robust algorithm for the identification of common substructures in chemical compounds. As shown in the previous examples (Figure 4.6) the algorithm cannot map atoms with different connectivity patterns, which, however, can be precisely assigned to each other, at first glance, by using chemical intuition (e.g. C_1). Figure 4.8 illustrates the mapping of disilene and its constitutional isomer, silylsilylene, with the CLCA algorithm. Both silicon atoms in disilene have three neighbours each, and the atoms Si_1 and Si_2 in silylsilylene have four and two neighbours respectively. So the connectivity patterns of silicon atoms in both structures are different, and the corresponding atoms cannot be mapped. Since one part of atoms are interchangeable (more than one mapping possible), and another part has different connectivity patterns, CLCA cannot find any matches. To overcome the connectivity restriction, Kumar et al.[46] have proposed to present chemical bonds as nodes as well. This approach changes the representation of the adjacency key. The bond strings are built in the same way as atom strings. Since bonds have no atomic number, a zero value is used instead (Figure 4.7).

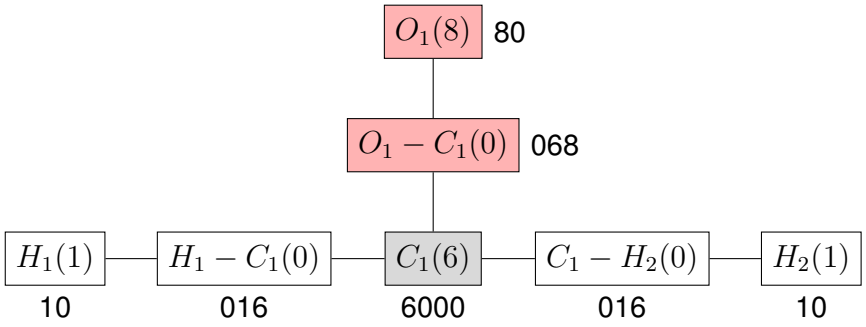


Figure 4.7: Conversion of bonds into nodes in an extended molecular graph representation of formaldehyde. The value of zero in parentheses is used to represent a bond.

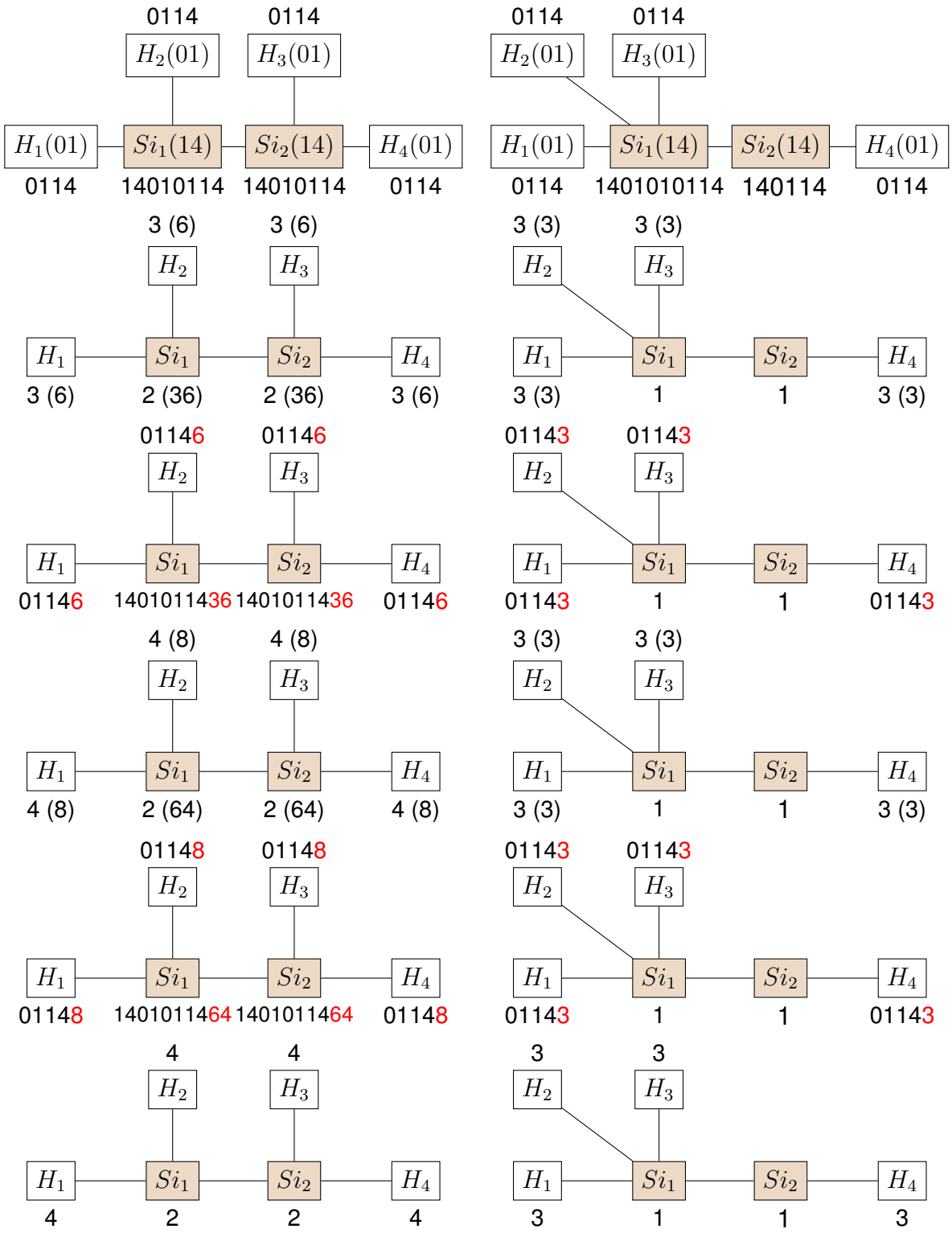


Figure 4.8: Attempted mapping of disilene (left) and silylsilylene (right) with the CLCA algorithm. Due to redundant atoms and atoms with different adjacency patterns CLCA fails to find any bijections between the atoms of the two molecules. Values in parentheses are the prime number products PNP of adjacent atoms. New atom strings generated by concatenation of original atom strings (in black) and $PNPs$ (in red).

This technique makes it possible to identify not only corresponding atoms but bonds as well. While the mapping of disilene and silylsilylene with the original CICA finds no mappings at all, extended CLCA with additional nodes recognises the bijection between the Si-Si bonds in both molecules (Figure 4.9).

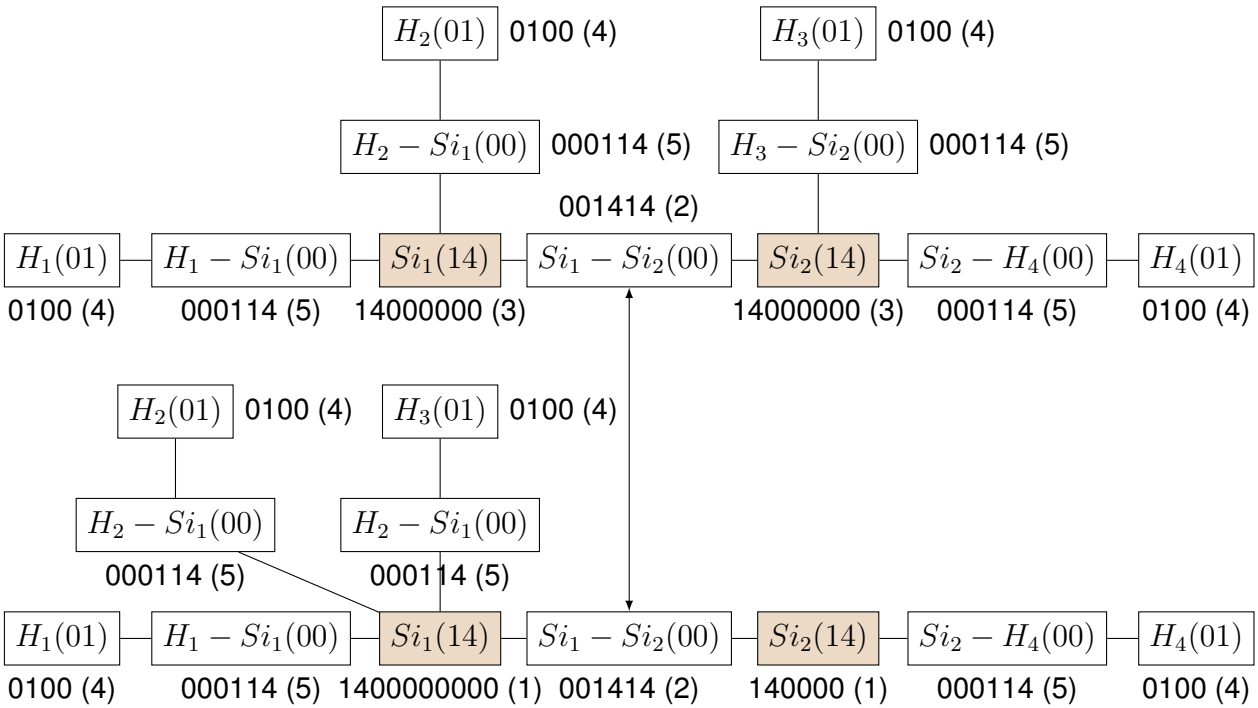


Figure 4.9: Mapping of disilene (top) and silylsilylene (bottom) with extended CLCA, where bonds are converted into nodes. In the construction of the node string, a value of zero is used to encode the bond. The extended algorithm with additional nodes, unlike the basic CLCA, is able to map Si-Si bonds of two molecules.

The atom string in the extended algorithm encodes only the information about the atom itself and the number of adjacent atoms. After the first CLCA loop, the generated string contains data about adjacent atoms. So it becomes equivalent to the atom string from the base algorithm. In this way, the string information is divided into smaller blocks, which makes the mapping more flexible.

For example, the mapping of propanol and propylamine (Figure 4.10) with original CLCA cannot identify the bijection between the carbons connected to oxygen and nitrogen atoms. Because the atom strings of the original algorithm include the information about connected atoms, both carbons become singular species in the first loop. However, in the extended CLCA both bonds connected to nitrogen and oxygen, as opposed to the atoms connected to the bonds, become unique and the corresponding C_1 atoms can be mapped.

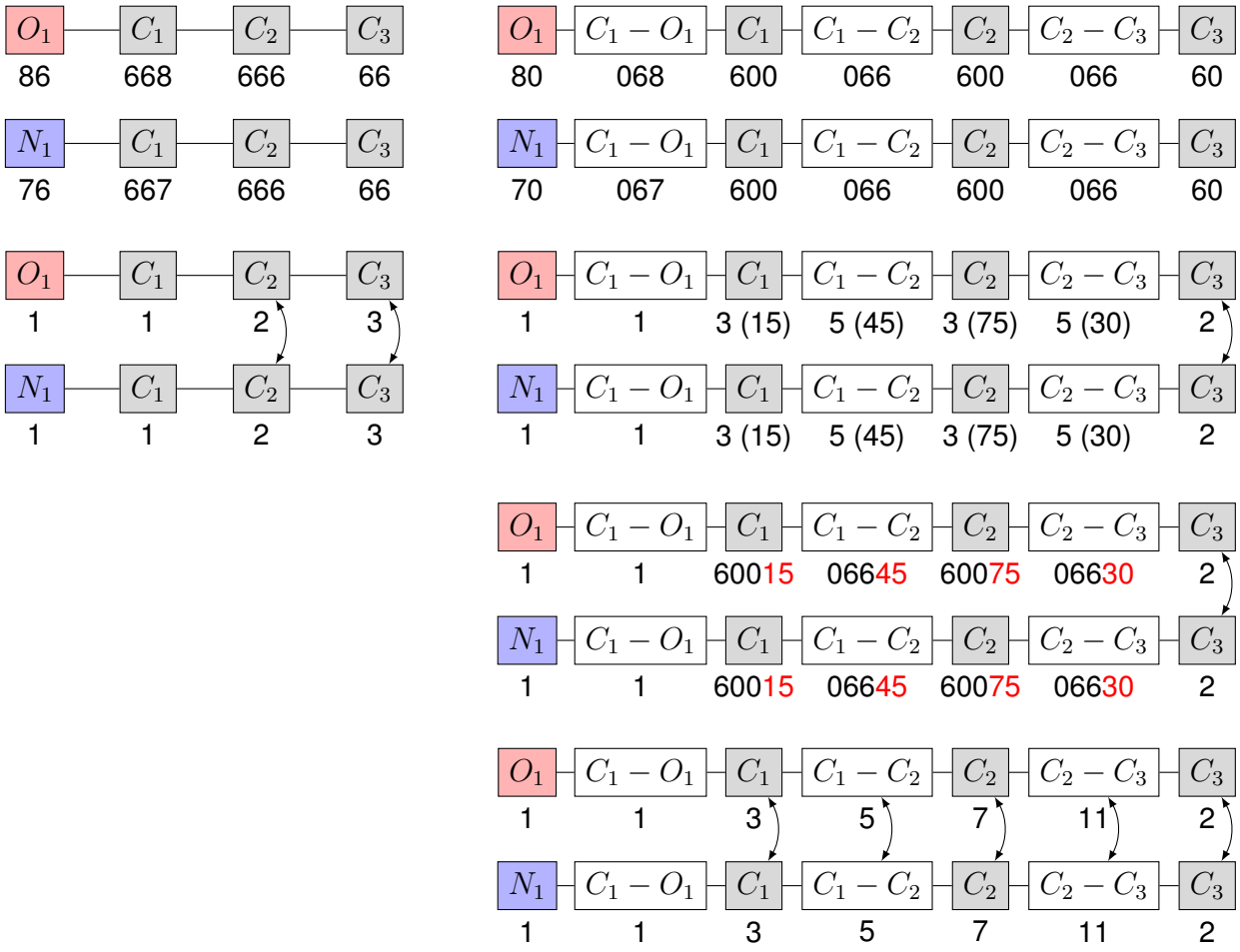
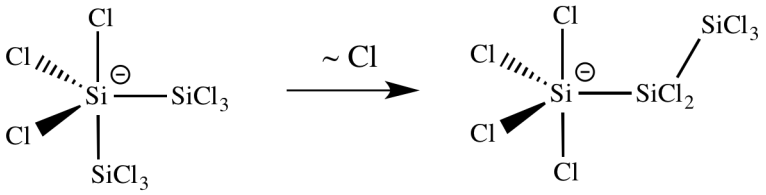


Figure 4.10: Mapping of propanol and propylamine with the basic CLCA algorithm (left) versus the extended CLCA algorithm (right).

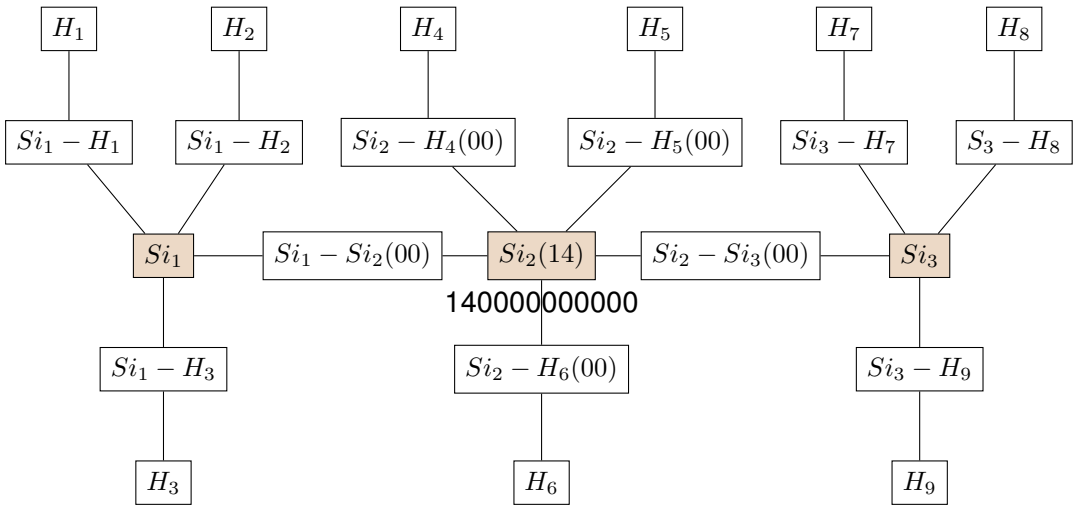
The information about a bond order (if available) can be encoded in a bond string and used for the mapping as well. It increases the probability of the correct mapping and makes it possible to identify bijections between molecules with different connectivity patterns.

The weakness of the extended method is that reduced information about the topological environment can lead to the incorrect mappings at the start, e.g. if a chlorine-shift reaction in perchlorinated silane is considered (Figure 4.11). Despite the apparent topological difference between silicon atoms connected to four chlorides, both atoms get the same string and CLCA maps them to each other. To avoid such accidental matches, only connected groups of mapped nodes[2] are considered as final mapping. Such groups must contain at least two nodes, e.g. an atom node and bond node. Another disadvantage of the extended method is the growing computational cost. After converting bonds to nodes, the total number of vertices in the algorithm rises from n to 2n.

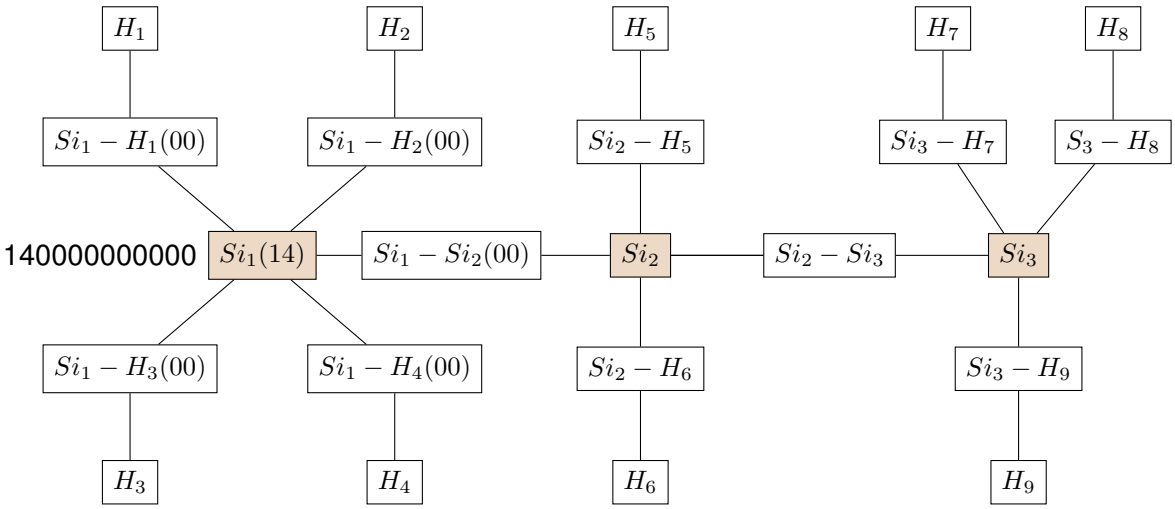
[2] As opposed to individual, unconnected atoms that may have been accidentally mapped.



(a)



(b)



(c)

Figure 4.11: An isomerisation reaction in a perchlorinated silane (a). The reactants and products converted into reactants graph (b) and products graph (c) respectively. Despite the topological difference between Si_2 in (b) and Si_1 in (c) both atoms are mapped together.

5. The A* algorithm

5.1 A* outline

A* (pronounced: ”a-star”) search is a popular and efficient searching technique used in artificial intelligence.[19] The A* search algorithm was developed in 1986 by Hart et al. and has found numerous applications in computer science since that time.[48] A* is a cornerstone of many popular algorithms because it can be used to solve many kinds of problems. The algorithm uses a heuristic function to find the shortest (or lowest cost) path through a search space to the goal state:

$$f = g + h,$$

where
g: the actual or accumulated cost path from the start to the current state,
h: the estimated future cost, a lower bound for the cost that will be still accumulated to get to the goal state,
f: the total path cost.

To illustrate the function consider a simple example of building a house. An empty area without a house is the start state, and the finished house is the goal state. The accumulated cost g at the start point equals zero and the estimated future cost h represents the estimated minimum amount of money needed to finish the house. So the total path cost f at the start will be equal to h. As the house will grow, the accumulated cost g will increase and the future cost f will decrease until it reaches zero, when the house is finished. Thus the total path cost f at the end will be equal to g. The goal of the A* algorithm in this example is to find the optimal path through the construction process that keeps the building cost as low as possible.

5.2 The A* atom mapping algorithm

The A* algorithm of Heinonen et al.[49] is an atom mapping algorithm based on the A* search technique. A* calculates the "cost" in terms of graph editing operations that transform the input graph into the the target graph.[50] The algorithm uses the principle of minimal chemical distance (PMCD)[51] to identify optimal mappings, i.e. those which require fewer graph editing operations. The PMCD states that most chemical reactions follow the shortest path for converting reactants into products, i.e. the path involving the smallest possible number of bond transformation. Consider the following simple example of a chemical reaction:

$$X - A + Y - B \longleftrightarrow X - B + Y - A$$

where X, Y, A, B are chemical species. In order to map atoms on the left-hand side to atoms on the right-hand side, the graph of reactants R has to be converted into the graph of products P. In this case at least four editing operations are required. Two edges (between X and A, and between Y and B) in R must be deleted, and two edges (between X and B, and between Y and A) in P have to be inserted. So the minimal chemical distance for converting the reactants to products is equal to four. At the beginning of the editing process, the accumulated cost g is equal to zero and the estimated future cost f is equal to four. Accordingly, the total cost f for this mapping equals four as well. In a partial mapping, where two bonds between X and A, and between Y and B in R are deleted and only one bond between X and B in P is created, the accumulated cost g will be three and the future cost h will be one. At the end of the mapping, g and h will be equal to four and zero respectively.[1] This simple example has only one possible edit sequence, e.g. one optimal mapping. The reason for this is the fact that there is only one atom of each type: X, Y, A and B. However, if we consider the following example:

$$A - B + X - X \longleftrightarrow A - X + B - X$$

there are two possible edit paths. In the first path, the node A can be connected to the left X and the node B to the right X. And in the second option, vice versa, A can be attached to the right X and B to the left X. So there are two optimal mappings which can be identified with the help of the A* Algorithm.

[1] The cost calculation is more complicated in complex cases (see below).

5.2.1 Editing costs

Within the A* atom mapping algorithm the editing costs are classified into bond editing costs and atom editing costs, which can be calculated with the help of bond and atom strings respectively. These are created employing the same technique that is used in the CLCA algorithm. This is illustrated in Figure 5.1.

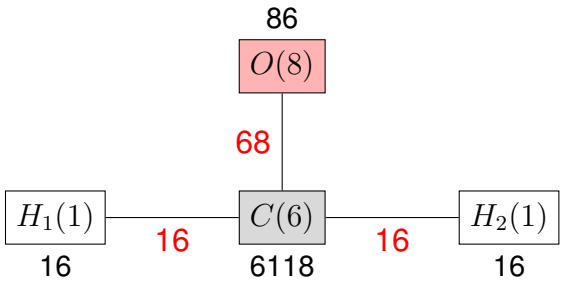


Figure 5.1: Assignment of atom strings (black) and bond strings (red) from atomic numbers (in parentheses) in the molecular graph of formaldehyde.

For example, the carbon atom C is assigned the atom string ”6118”. The first number, ”6”, represents the atomic number of carbon. The next two digits ”11” represent two hydrogen atoms and the last digit ”8” represents the atom oxygen. Atomic numbers of neighbouring atoms are sorted in ascending order. The bond strings are created by concatenation of the atomic numbers of the two connected atoms in ascending order. For example, a bond between a carbon and an oxygen atom would receive the bond string ”68”.

The editing costs by the mapping of two graphs can be calculated via the symmetric difference $\triangle$ of two string lists, i.e. the lists of bond (atom) strings of reactants and of products. For example, the symmetric difference of the two lists $[1, 2, 3]$ and $[2, 3, 3, 4]$ is $[1, 3, 4]$, because deletion of equal elements of the first list from the second list yields $[1, \cancel{2}, \cancel{3}] \triangle [\cancel{2}, \cancel{3}, 3, 4] = [1, 3, 4]$. Furthermore, the symmetric difference of two equal lists $[1]$ and $[1]$ is an empty list $[\,]$, that is, the difference between both is zero. The accumulated cost g is then calculated through the bond editing cost (using bond strings) of the mapped part of the molecule. The unmapped part is considered by estimation of the future cost h which consists of bond editing cost h_b (using bond strings) and atom editing cost h_a (using atom strings).

A simple example of the editing cost calculation for a partial mapping is shown in the Figure 5.2. Attention should be drawn to the way how atom and bond strings in partial mappings are assigned. The molecular graph of the partially mapped molecule is divided into two induced subgraphs (cf. Chapter 2.1). The first subgraph is induced by the mapped nodes and the second one by the unmapped nodes. By the string assignment, the two subgraphs must be considered as different graphs which are not connected to each other. For example, the node of nitrogen in both graphs belongs to the unmapped part (red) of the molecule, and there is no bond between carbon in the green subgraph and nitrogen in the red subgraph. So the nitrogen is assigned an atom string ”7”.

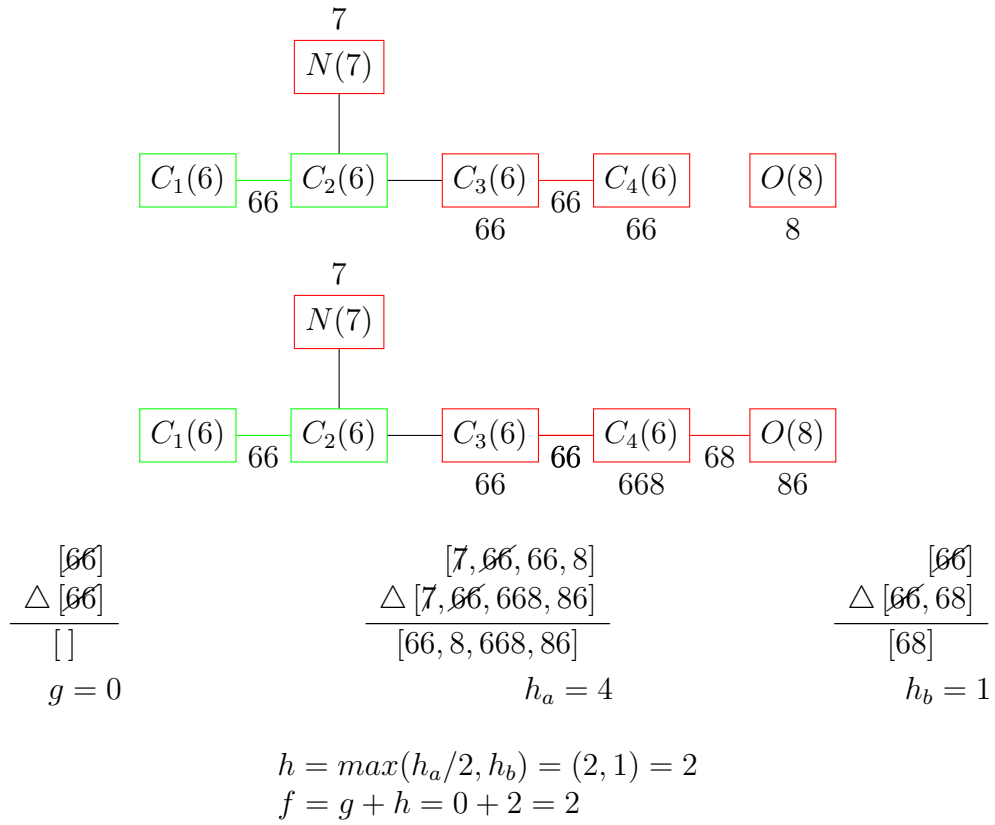


Figure 5.2: Calculation of the editing costs of two molecular graphs. Hydrogen atoms are omitted for simplicity. Green and red nodes represent mapped and unmapped parts of the graph respectively. The accumulated cost g is calculated through the bond cost of the mapped part of the molecule (green), i.e. the subgraph induced by mapped nodes. The future bond cost h_b and the future atom cost h_a are estimated using the unmapped part of the molecule (red), i.e. the subgraph induced by unmapped nodes.

The key feature of A* Algorithm is the heuristic function which represents the future cost. This is further broken down into the atom cost h_a and the bond cost h_b.[2] The future cost h which the algorithm uses, is the maximum value of h_a and h_b.

[2]To make both values comparable, the atom cost is divided by two because one edge-edit operation changes precisely two atom neighbourhoods.

5.2.2 Algorithm workflow

The most important steps of the A* atom mapping algorithm are presented in Figure 5.3.
(1) First, the algorithm takes two molecular graphs as input.
(2) Second, A* needs a starting point in the form of two mapped atoms r and p in both graphs, e.g. node r of graph R which can be mapped to the node p of graph P. This input can be entered manually or taken from another algorithm, e.g. CLCA.[49]
(3) Third, the graph with the largest number of edges is used as the reference graph. The nodes of this graph are sorted in breadth-first search order, that is, the neighbor nodes of a starting node are added first, before moving to the next level neighbours.[52]. This order determines which atom of the reference graph will be considered for mapping next.
(4) Next, the first mapping pair with the total path cost $f = 0$ is added to the priority queue. The priority queue is a type of list that sorts items by their costs and always returns the item with the lowest cost.
(5) In the next step, the upper bound for the total path cost ub_f is initialised to infinity. As soon as the algorithm finds the first mapping, ub_f will be set to its editing cost (see below). ub_f is used to stop the algorithm when the editing cost of a new mapping exceeds the cost of a previous one.[3]
(6) In the next step, the future cost of complete mapping (both graphs are unmapped) is calculated and the upper bound for the accumulated cost ub_g is set to this value. In this new implementation of the A* algorithm, ub_g is responsible for keeping the priority queue small, i.e. partial mappings whose accumulated cost already exceeds the estimated cost of the complete mapping are removed.
(7) The algorithm stops if the priority queue is empty, otherwise the mapping with the smallest total cost f is taken from the priority queue and considered.
(8) In the following step, the algorithm determines if f of the current mapping is smaller than the upper bound for the total cost ub_f. If this is the case, the algorithm stops, because only worse solutions (with higher costs) are left in the priority queue.
(9) If f is smaller or equal to ub_f, A* determines if the current mapping is complete.
(10) The complete mapping is added to the set of optimal mappings and the upper bound for total cost ub_f is set to the total cost f of the current mapping.

[3]Which would indicate that the new mapping is worse, not better, than the previous one.

(11) If the mapping is incomplete, it will be expanded in the next step. To expand the mapping the next atom from the breadth-first order of the reference graph is combined with all atoms of another graph which have the same chemical element.
(12) Next, for each possible mapping g, h and f are calculated.
(13) If the accumulated cost g of the current mapping does not exceed the upper bound for accumulated cost ub_g the mapping and its corresponding f value are added to the priority queue. Otherwise, the mapping is pruned from the mapping process.

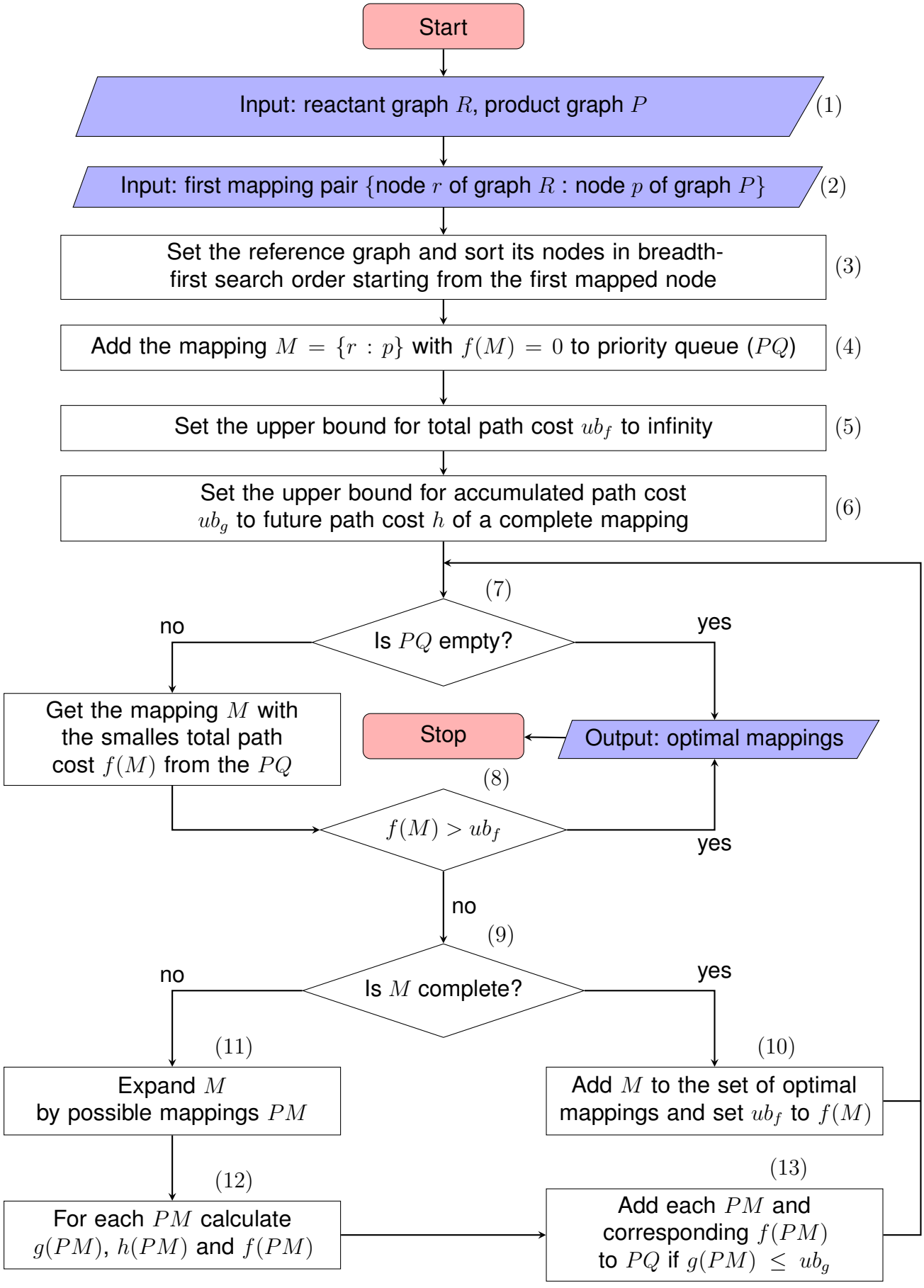


Figure 5.3: A flowchart diagram for the A* algorithm. Numbers in parentheses refer to indexes in the text.

5.2.3 A worked example of reaction mapping

The following example demonstrates a simplified reaction mapping workflow of two structures ($X-C_1-C_2-C_3$ and $X+C_1-C_2-C_3$) with the A* algorithm (Figure 5.4). First, the initial mapping A with two mapped atoms is returned from a priority queue.[4] There are three possibilities for the matching of the next atom $C1^\star$ in the reference molecule $X - C_1^\star - C_2 - C_3$. After expanding the partial mapping A, three new combinations (B, C, D) are created and added to the priority queue. In the next step, the two mappings B and D with the least path cost $f = 1$ are retrieved from the priority queue.[5] There are only two carbon atoms left (C_2 and C_3) and consequently, only two possibilities to map the atom $C_2^\star$. After expanding the structure B, two new structures E and F with path costs two and one respectively are created. The structure D is expanded to two new mappings G and H as well. Their f values are equal to two and one respectively. All four new combinations are added to the queue, which now contains five partial mappings with corresponding f values in parentheses: $F(1), H(1), E(2), G(2)$ and $C(3)$. The combinations F and H with the highest priority[6] are returned first from the queue. There are only two unmapped atoms in both structures left, so they are mapped to each other. Consequently, two new mappings I and J, which are complete, are added to the queue. After the last step, the priority queue contains the following mappings: $I(1), J(1), E(2), G(2)$ and $C(3)$. When one of both complete mappings I and G are returned from the priority queue, the algorithm adds this mapping to the set of optimal mappings and sets the upper bound for total cost ub_f to f of the complete mapping. After the second complete mapping is added to the set of optimal mappings the algorithm takes the next mapping E or G (not shown) and determines that their path costs exceeds the upper bound for f. This means that no better solutions are left in the priority queue, and the A* algorithm stops the search. The partial mappings $C(3)$ and $E(2)$ or $G(2)$ (depending which one was returned first) stay in the priority queue without expanding.

[4] The step is not shown due to space considerations. The first mapping pair X-X is choosen manually
[5] The order in which the mappings with equal costs are returned from the priority queue is insignificant.
[6] The highest priority have the mappings with the smallest editing costs.

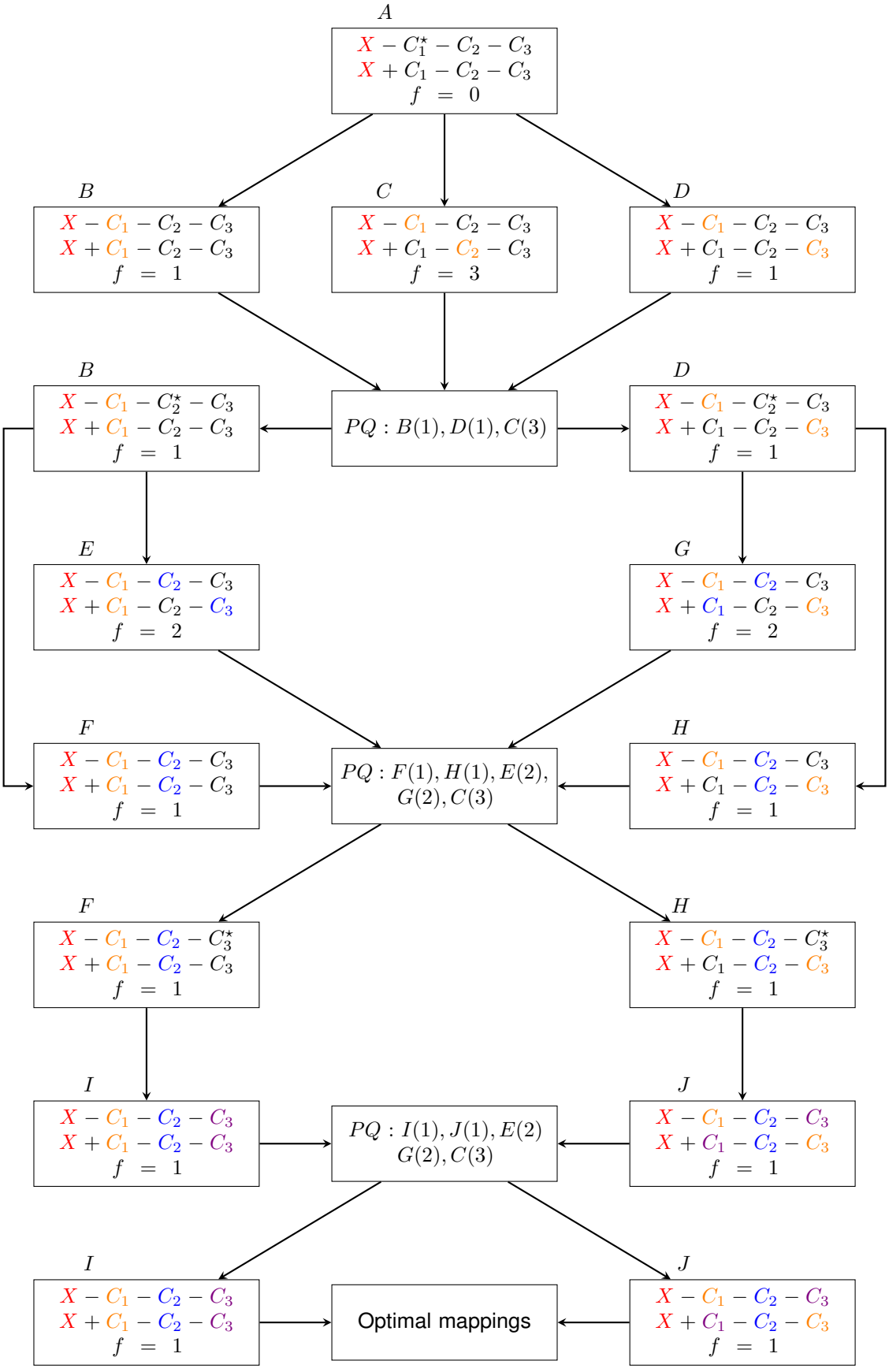


Figure 5.4: Reaction mapping with A* algorithm. The under consideration atom is marked with a star. All pairs of mapped atoms are colour-encoded (unmapped atoms are black). A priority queue (PQ) is repeated for a better presentation.

5.2.4 Path tracing

The accumulated cost g for the mapped part of the molecule is recalculated each time the algorithm expands a mapping by identifying a new possible bijection between two nodes. However, this approach fails for symmetric structures (or substructures). As far as the mapping of the graphs in Figure 5.5 is considered two possible matches are easily identified by visual inspection.

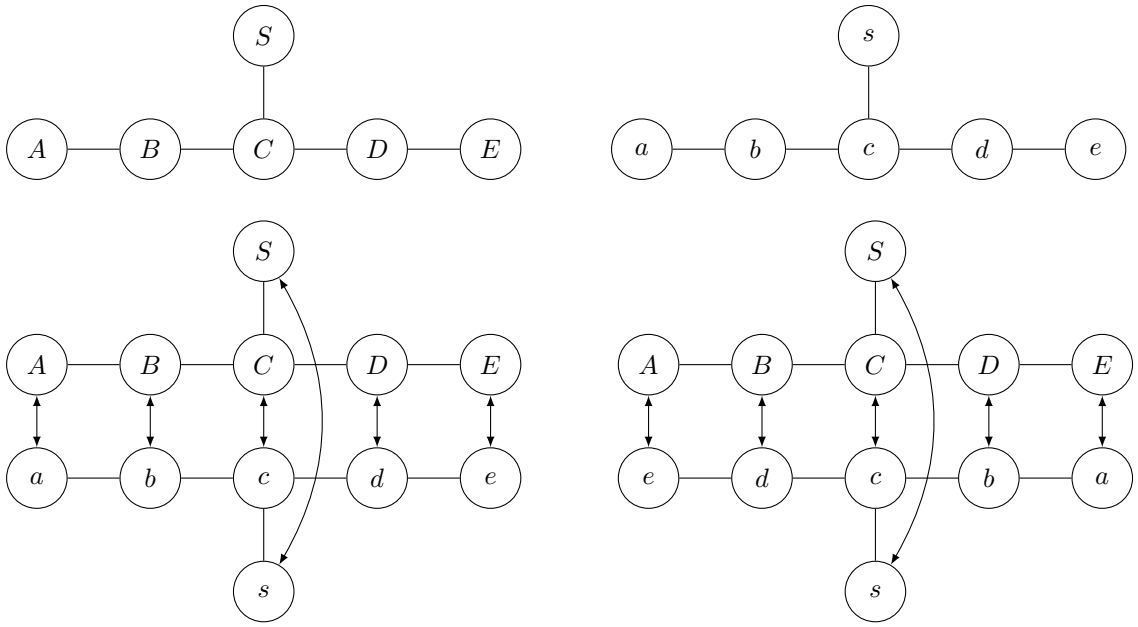


Figure 5.5: Symmetric graphs (top) and their possible mappings (bottom).

However, if the A* algorithm maps two structures starting from the nodes S and s in addition to correct mappings it determines two wrong ones (Figure 5.6). The last nodes (black) are incorrectly mapped to the nodes on opposing branches. The reason for this is that after expanding of the partial mapping (green) with a new combination A and a, the new partial mapping has the same accumulated cost (which is equal to zero) as the mapping where the node A is mapped to the node d. The same is true for partial mappings with combinations D - a and D - d.

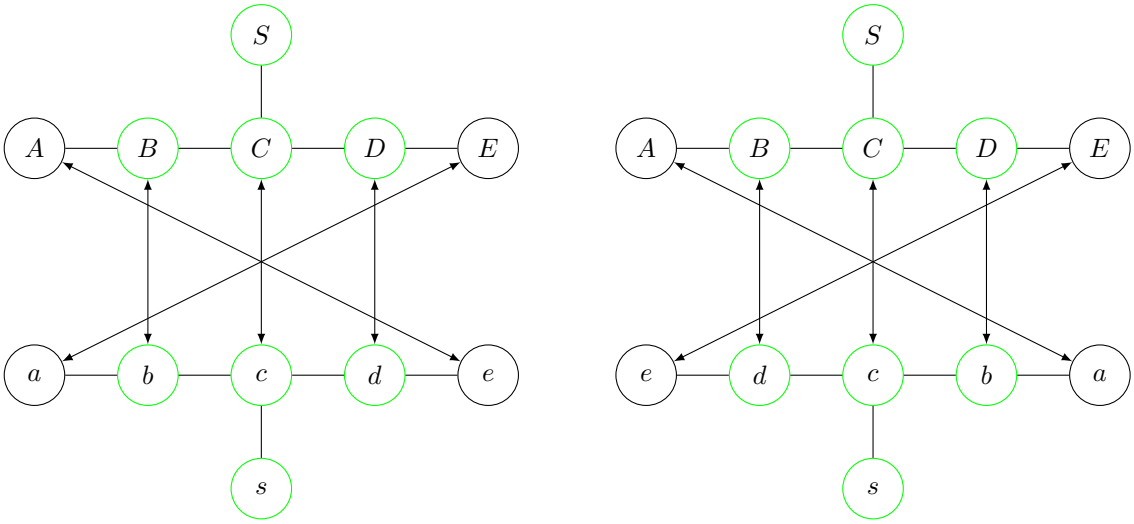


Figure 5.6: Incorrect mappings of symmetric graph identified with the A* algorithm.

To overcome this problem a path tracing extension of the A* algorithm was implemented. The basic idea behind this approach is that the accumulated cost g is being accumulated instead of being recalculated in each step. Each mapping is assigned an individual variable g and accumulated cost calculated in each step g_{step} is added to g. After a new pair of atoms is mapped the extended algorithm operates as follows. First, it determines the neighbours of the newly mapped atoms, which are themselves already mapped. Second, A* identifies if the neighbours' list of another graph includes the nodes to which the nodes from the neighbour's list of a reference graph are mapped. If it is the case these nodes together with the newly mapped node of another graph are used to induce the subgraph for the calculation of the accumulated cost for the mapping step. In the reference graph, the subgraph for the accumulated cost calculation is induced by the newly mapped node and all nodes from its neighbour's list. Using the new method, the A* algorithm can eliminate the two incorrect mappings from the final result.

For example, if the node A is being mapped to e, while the node B has been already mapped to b, the accumulated cost for this mapping is equal to one (Figure 5.7).

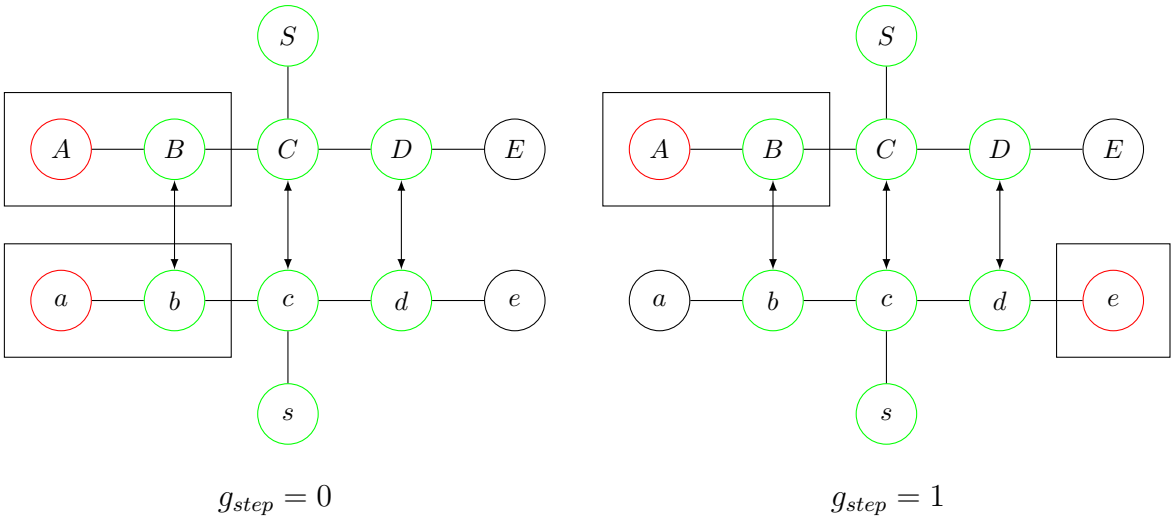


Figure 5.7: Calculation of accumulate cost of the mapping step for correct (left) and incorrect (right) mappings. The atoms under consideration are red. Rectangles indicate the induced subgraphs used for the calculation of the accumulated cost of the current step.

The reason for this is that a mapped neighbour of e is the node d, which has no corresponding nodes in the mapped neighbours' list of A. So the accumulated cost of a current mapping is calculated using two subgraphs. A first subgraph is induced by nodes A and B in the reference graph. The second subgraph is induced by only one node e in another graph. The $A - B$ subgraph has one bond, and the e subgraph has no bonds, so the difference is equal to one.

6. Summary and Outlook

The aim of this work is to identify, evaluate and implement suitable algorithms for atom mapping of chemical reactions. The main requirements for the atom mapping algorithms are the ability to map all atoms, including hydrogen atoms, in inorganic structures which can build complex networks, the capability to deal with stereochemistry and reasonable computational time for moderately large molecules (up to 100 atoms). The input data for algorithms are two sets of chemical elements and their cartesian coordinates. Both sets of atoms are converted into nodes of the molecular graph. The bonds of the molecules are represented by edges. To identify edges in the molecular graphs, the distance matrix between all possible atom combinations is computed, from which edges are generated by and interatomic distance criterion based on tabulated atomic covalent radii[20].

Two existing algorithms, the Canonical Labeling for Clique Approximation (CLCA) algorithm and the A* atom mapping algorithm, were identified as most suitable to fulfil the requirements. The CLCA algorithm is based on an extended connectivity method. It uses prime factorisation to encode the information about atom location in the full topological space of the molecule. After evaluation of recently published algorithms, Gonzalez et al.[13] showed that CLCA is comparable to commercial atom mapping tools like MWED[53], ICMAP[54] and AtoMapper and highlighted its ability to map imbalanced reactions with explicit hydrogen atoms. With features like the identification of equivalent atoms, the CLCA algorithm is ahead of competing strategies. The extended CLCA makes it possible to overcome the connectivity restriction by mapping two atoms which have different neighbours. It can identify not only corresponding atoms but corresponding bonds as well. Also, CLCA has a polynomial time complexity and can calculate the bijection between two molecules faster than other algorithms. The main shortcoming of CLCA is inability to map atoms with different connectivity patterns.

The A* Atom Mapping algorithm is based on the popular A* search technique which is widely used in artificial intelligence research. A* uses a smart heuristic to guide itself through the space of atom mappings to find the optimal results. The main shortcoming of the original A* algorithm is its inability to distinguish between different branches in symmetric structures. This weakness results in additional incorrect mappings. To overcome this problem, the path tracing extension was implemented. Also, a method for

pruning of incorrect mappings from the search was added to the algorithm. The great advantage of the A* algorithm is the straightforward possibility to include extensions and fine-tuning. In some cases, the principle of minimal chemical distance, which the algorithm uses for its search, may fail. For example, if in the real reaction the bond editing cost is equal to four and the algorithm finds a way to map the reactants and products by changing only two bonds. In this case, the introduction of additional parameters, like different costs for different bond types may be useful, e.g. bond costs in correlation with bond strengths.[49] The CLCA and A* algorithms are complementary and can be used together in one programming script. The main advantage of CLCA is the fast identification of the MCS of two molecules. The partial mapping calculated with CLCA can be used to decrease computational time as well to provide the start point for the A* algorithm. The main problem of both algorithms is their inability to handle stereochemistry. As consequence, the multiple possible mappings produced by the A* algorithm should be inspected manually to identify these issues. Kuwabata et al.[55] proposed a method for three-dimensional flexible alignment of two molecules using a two-dimensional maximum common substructure.This approach could be implemented as a next step in the atom mapping script to reduce the number of mappings produced by the A* algorithm using stereochemical information.

Two algorithms, the CLCA algorithm and the A* algorithm, were implemented in two separate Python 3.6 scripts (Appendix). Both working scripts accept two .xyz files with chemical elements and their cartesian coordinates as input. After construction, molecular graphs are visualised in a three-dimensional plot. The first mapping pair in the A* algorithm must be entered manually. The complete main script, which is not finished, must have the following features: (i) the partial mapping calculated with CLCA must be used for decreasing of computational time as well for providing the start point for the A* algorithm, (ii) a user must be able to map visually distinguishable unmapped atoms manually, (iii) the A* and manual mappings must run parallel to reduce computation time.

Bibliography

[1] J. W. Raymond, P. Willett, *Journal of computer-aided molecular design* **2002**, *16*(7), 521–533.

[2] J. M. Barnard, *Journal of Chemical Information and Computer Sciences* **1993**, *33*(4), 532–538.

[3] M. S. Lajiness, *Perspectives in drug discovery and design* **1996**, *7*(1), 65–84.

[4] J. A. Doudna, *Nature Structural and Molecular Biology* **2000**, *7*(11s), 954.

[5] A. Bender, R. C. Glen, *Organic & biomolecular chemistry* **2004**, *2*(22), 3204–3218.

[6] P. Willett, *Journal of Medicinal Chemistry* **2005**, *48*(13), 4183–4199.

[7] M. Wagener, J. Sadowski, J. Gasteiger, *Journal of the American Chemical Society* **1995**, *117*(29), 7769–7775.

[8] R. Körner, J. Apostolakis, *Journal of chemical information and modeling* **2008**, *48*(6), 1181–1189.

[9] A. L. Teixeira, A. O. Falcao, *Journal of chemical information and modeling* **2013**, *53*(10), 2511–2524.

[10] P. Willett, J. M. Barnard, G. M. Downs, *Journal of chemical information and computer sciences* **1998**, *38*(6), 983–996.

[11] D. Conte, P. Foggia, C. Sansone, M. Vento, *International journal of pattern recognition and artificial intelligence* **2004**, *18*(03), 265–298.

[12] W. L. Chen, D. Z. Chen, K. T. Taylor, *Wiley Interdisciplinary Reviews: Computational Molecular Science* **2013**, *3*(6), 560–593.

[13] G. A. P. Gonzalez, L. R. El Assal, A. Noronha, I. Thiele, H. S. Haraldsdóttir, R. M. Fleming, *Journal of cheminformatics* **2017**, *9*(1), 39.

[14] N. Biggs, E. K. Lloyd, R. J. Wilson, *Graph Theory, 1736-1936*, Oxford University Press, **1976**.

[15] V. I. Voloshin, *Introduction to graph theory*, Nova Science Publ., **2009**.

[16] A. R. Leach, V. J. Gillet, *An introduction to chemoinformatics*, Springer Science & Business Media, **2007**.

[17] E. Duesbury, J. D. Holliday, P. Willett, *MATCH Communications in Mathematical and in Computer Chemistry* **2017**, *77*(2), 213–232.

[18] H.-C. Ehrlich, M. Rarey, *Wiley Interdisciplinary Reviews: Computational Molecular Science* **2011**, *1*(1), 68–79.

[19] T. Akutsu, H. Nagamochi, *Computational and structural biotechnology journal* **2013**, *5*(6), e201302004.

[20] P. Pyykkö, M. Atsumi, *Chemistry–A European Journal* **2009**, *15*(1), 186–197.

[21] H. Morgan, *Journal of Chemical Documentation* **1965**, *5*(2), 107–113.

[22] M. F. Lynch, P. Willett, *Journal of Chemical Information and Computer Sciences* **1978**, *18*(3), 154–159.

[23] G. Vleduts, *British Library Research and Development Department Report* **1977**, (5399), 7668–7671.

[24] J. J. McGregor, P. Willett, *Journal of Chemical Information and Computer Sciences* **1981**, *21*(3), 137–140.

[25] K. Funatsu, T. Endo, N. Kotera, S.-I. Sasaki, *Tetrahedron Computer Methodology* **1988**, *1*(1), 53–69.

[26] K. Mehlhorn, *Data structures and algorithms 2: graph algorithms and NP-completeness*, Bd. 2, Springer Science & Business Media, **2012**.

[27] J. J. McGregor, *Software: Practice and Experience* **1982**, *12*(1), 23–34.

[28] Y. Cao, T. Jiang, T. Girke, *Bioinformatics* **2008**, *24*(13), i366–i374.

[29] R. J. Van Berlo, W. Winterbach, M. J. De Groot, A. Bender, P. J. Verheijen, M. J. Reinders, D. De Ridder, *International journal of bioinformatics research and applications* **2013**, *9*(4), 407–432.

[30] L. Schietgat, J. Ramon, M. Bruynooghe, H. Blockeel, in *International Conference on Discovery Science*, Springer, **2008** S. 197–209.

[31] T. Horváth, J. Ramon, S. Wrobel, *Data Mining and Knowledge Discovery* **2010**, *21*(3), 472–508.

[32] T. Akutsu, *Journal of Computational Biology* **2004**, *11*(2-3), 449–462.

[33] J. D. Crabtree, D. P. Mehta, *Journal of Experimental Algorithmics (JEA)* **2009**, *13*, 15.

[34] E. L. First, C. E. Gounaris, C. A. Floudas, *Journal of chemical information and modeling* **2011**, *52*(1), 84–92.

[35] J. Harrison, M. Lynch, *Journal of the Chemical Society C: Organic* **1970**, (15), 2082–2087.

[36] M. F. Lynch, P. Willett, *Journal of Chemical Information and Computer Sciences* **1978**, *18*(3), 149–154.

[37] N. Osório, P. Vilaça, M. Rocha, in *International Conference on Practical Applications of Computational Biology & Bioinformatics*, Springer, **2017** S. 257–264.

[38] C. Bron, *Communications of ACM*, *16*(9).

[39] R. Carraghan, P. M. Pardalos, *Operations Research Letters* **1990**, *9*(6), 375–382.

[40] P. Jauffret, C. Tonnelier, T. Hanser, G. Kaufmann, R. Wolff, *Tetrahedron Computer Methodology* **1990**, *3*(6), 335–349.

[41] I. Koch, *Theoretical Computer Science* **2001**, *250*(1-2), 1–30.

[42] J. R. Ullmann, *Journal of the ACM (JACM)* **1976**, *23*(1), 31–42.

[43] A. T. Brint, P. Willett, *Journal of Chemical Information and Computer Sciences* **1987**, *27*(4), 152–158.

[44] G. M. Downs, M. F. Lynch, P. Willett, G. A. Manson, G. A. Wilson, *Tetrahedron Computer Methodology* **1988**, *1*(3), 207–217.

[45] A. Wong, F. Akinniyi, in *Proc. 1983 Int. Conf. Syst., Man, and Cybern*, **1983** S. 197–201.

[46] A. Kumar, C. D. Maranas, *Journal of chemical information and modeling* **2014**, *54*(12), 3417–3438.

[47] V. V. Lozin, *Information Processing Letters* **2002**, *81*(1), 7–11.

[48] L. H. O. Rios, L. Chaimowicz, in *Brazilian Symposium on Artificial Intelligence*, Springer, **2010** S. 253–262.

[49] M. Heinonen, S. Lappalainen, T. Mielikäinen, J. Rousu, *Journal of Computational Biology* **2011**, *18*(1), 43–58.

[50] K. Riesen, H. Bunke, *Image and Vision computing* **2009**, *27*(7), 950–959.

[51] C. Jochum, J. Gasteiger, I. Ugi, *Angewandte Chemie International Edition in English* **1980**, *19*(7), 495–505.

[52] R. Stephens, *Essential Algorithms: A Practical Approach to Computer Algorithms*, 1. Aufl., Wiley Publishing, **2013**.

[53] M. Latendresse, J. P. Malerich, M. Travers, P. D. Karp, *Journal of chemical information and modeling* **2012**, *52*(11), 2970–2982.

[54] H. Kraut, J. Eiblmaier, G. Grethe, P. Löw, H. Matuszczyk, H. Saller, *Journal of chemical information and modeling* **2013**, *53*(11), 2884–2895.

[55] T. Kawabata, H. Nakamura, *Journal of chemical information and modeling* **2014**, *54*(7), 1850–1863.