Review[1] of
Handbook of Chemoinformatics Algorithms
by Faulon, Bender, eds.
CRC Press, 2010
440 pages, hardcover

Review by
Aaron Sterling (sterling@iastate.edu)
Department of Computer Science, Iowa State University
(visiting EECS Department, Northwestern University)

1 Introduction

This book is written by chemists, for chemists, with the objective of producing "an overview of some of the most common chemoinformatics algorithms in a single place." A first attempt at a definition of *chemoinformatics* might be, "Algorithms, databases and code to help chemists." Unlike the field of bioinformatics, which enjoys a rich academic literature going back many years, the *Handbook of Chemoinformatics Algorithms* (HCA) is the first book of its kind. The editors (Faulon and Bender) commissioned authors to produce chapters on topics like computer-aided molecular design, open-source chemoinformatics software, and how to store chemical structures and properties in databases. Most of the algorithms presented, from the point of view of a theoretical computer scientist, are "folklore" or "trivial"; however, the application of these algorithms is complex, the problems computational chemists face are hard and not always well defined mathematically, and hundreds of millions of dollars ride on the speed and usefulness of practical implementations of these algorithms. Therefore, this book has the potential to be important to the TCS community: fundamental improvements to techniques in this volume could be a big deal.

That said, HCA is not for everyone. There are no theorems or proofs, most of the chapters are written in a laundry-list style ("This works in this case, and *this* works in *this* case"), and the language barrier is formidable. To provide a single data point, I took a year of chemistry in college, and did well, but even so, I found myself spending a lot of time reading Wikipedia articles just to understand what was being discussed. Further, the various authors do not appear to have experience framing things for a TCS reader. As one example, the phrase "non-polynomial (NP)-hard problem" appears in Chapter 9.

To aid in digestion of the material, I recommend one read the chapters out of order. HCA presents general technical details in earlier chapters, and motivating applications in later chapters. Specifically, I recommend the reader start with Chapter 13 (on sequence alignment algorithms) which is accessible with little chemistry background, then read Chapters 5-15 as desired. Refer to Chapters 1-4 to fill in gaps, or once the motivating applications are clear. (Readers with a background in bioinformatics or computational biology may wish to start by reading Chapters 13-15, which apply techniques from those areas to chemoinformatics.)

Chemoinformatics itself is defined by the editors as "the field of handling chemical information electronically"; the term was coined by F. K. Brown in 1998. (Chemoinformatics is sometimes

 This work was supported in part by NSF grant CCF-1049899. The author would like to thank Bill Gasarch, who made helpful comments on an earlier draft of this review.

spelled "cheminformatics," without the first "o"; the terms are interchangeable.) According to Rajarshi Guha, who discusses open-source software in Chapter 12:

> Although the idea of free and shared software has been in existence since the 1960s, such software has not been generally available in chemoinformatics. In contrast, the field of bioinformatics has produced a number of core algorithms (such as BLAST) as freely available software. Part of the reason for this is the fact that the majority of chemical information has been proprietary, in contrast to biological data (such as gene sequences and protein structures), which has traditionally been freely exchanged.

Both the editors and other chapter authors make similar points: because of the proprietary nature of chemical databases, etc., thoroughgoing academic (public sector) investigation into chemoinformatics is a recent phenomenon, and many functionalities are not available for free, nor inexpensively—nor, publicly, at all.

As a general warning to the reader, I was unable to determine how well the techniques in HCA work, even after reviewing several Wikipedia articles and one other computational chemistry book. Authors of most chapters in HCA make the point that the techniques they present have been used to find compounds of potential interest more inexpensively than by using wetlab experimentation alone, but the ratio of hits to misses is not something I was able to track down. It does seem, especially in the area of QSAR models that I discuss in Section 2, that there has been serious criticism leveled against *in silico* research as publications devoid of real-world applicability. Some authors in HCA address those concerns, and present techniques with safeguards they claim solve those problems, but, even so, they provide only anecdotes of success, no hit-and-miss ratio. Still, the algorithms in HCA are employed by petroleum companies, chemical companies, pharmaceutical companies, universities worldwide, and the FDA, so, for better or for worse, they are what's currently available.

I will focus this review on the two chemoinformatics applications most discussed in HCA: quantitative structure-activity relationships (QSAR), and computer-assisted molecular design (CAMD). Then I will briefly consider the contents of HCA chapter by chapter.

2 Principal motivating applications

2.1 Quantitative Structure-Activity Relationships (QSAR)

QSAR is far and away the most-discussed application in HCA; it is mentioned in every chapter. A *quantitative structure-activity relationship* is a mathematical relationship between a molecule's physical properties and its chemical properties, along the lines of, "All molecules whose molecular graphs contain subgraph G will produce reactant R when reacting with chemical C." The primary purpose of QSAR is to *make predictions* about how an as-yet-unstudied molecule will behave, based on better-known molecules that are structurally similar to it. Some QSAR approaches use algorithms that solve the Graph Isomorphism Problem; while this can work for small examples, or in situations where heuristics help, all known Graph Isomorphism algorithms are (of course) worst-case infeasible. Therefore, the main QSAR approaches collapse the information in molecules down to individual real numbers called *molecular descriptors*, which are the topic of Chapter 4. For different applications, many different molecular descriptors may be required.

The Wikipedia article on QSAR is well-written, and I recommend it. Tropsha and Golbraikh in Chapter 6 of HCA describe QSAR as follows.

> The discovery of novel bioactive chemical entities is the primary goal of computational drug discovery, and the development of validated and predictive QSAR models is critical to achieve this goal.... The main QSAR hypothesis underlying all QSAR studies is as follows: similar compounds should have similar biological activities or properties. If this condition for compounds in the dataset is not satisfied, building truly predictive QSAR models is impossible. In fact, one can define two compounds to be similar if their chemical structures are similar. In computer representation, compounds are characterized by a set of quantitative parameters called descriptors. *Similarity* between two compounds is a quantitative measure that is defined based on compounds' descriptor values. Different definitions of compound similarity exist.... Obviously, quantitative values of similarity measures between two compounds also depend on which descriptors are used. So there is no unique similarity measure.

QSAR is more complicated than "just" a massive chemical database search, because, in reality, structurally similar molecules can have very different properties. (This is sometimes termed the "SAR Paradox.") As a result, much of the literature on *in silico* QSAR provides only *conditional results*, along the lines of: assuming molecule family $\mathcal{F}$ has similar properties to its structural cousins $\mathcal{F}'$, we can deduce results X, Y and Z. Recently, prominent scientists and scientific organizations have called much of the QSAR research into question, because of the uncertainty (and probable falsity) of many starting assumptions. The European Organization for Economic Co-operation and Development now requires "appropriate measures of goodness-of-fit, robustness and predictivity" for QSAR models. To address these concerns, HCA devotes two chapters to QSAR: Chapter 6, on data preparation and modeling; and Chapter 7, on development and validation of the QSAR models themselves.

2.1.1 Chapter 6: Predictive Quantitative Structure-Activity Relationships Modeling: *Data Preparation and the General Modeling Workflow*

There are now large (millions of compounds) chemical databases publicly available online. In Chapter 6, the authors (Tropsha and Golbraikh) present their workflow for developing predictive QSAR models from such databases. The workflow breaks down to the following general steps: data treatment and preparation, calculation of molecular descriptors (i.e., construction of a similarity metric over the dataspace); preprocessing those descriptors; clustering the dataspace according to the similarity metric; and detecting and removing outliers from those clusters (i.e., compounds that do not share properties with other cluster elements, even though they are metrically close to them). A cluster thus produced can serve as a QSAR model. Finally, in addition to model validation, the authors recommend laboratory verification of the *in silico* prediction that compound X will have property Y.

The authors use a machine-learning approach. From the initial dataset, they first remove 10-20% of the compounds at random, for use as an external evaluation set, for eventual model verification. The remaining compounds are divided into multiple training and test sets. The current computational upper limit of the size of the training set is 2000 compounds. The authors recommend a minimum of 20 compounds in the training set, and a minimum of ten compounds in each of the test and external evaluation sets, to avoid chance correlation and overfitting.

Once compounds are chosen and assigned to sets, descriptors are calculated for each molecule. Molecular descriptors are real numbers that quantify a molecular property. Examples that do not require chemistry vocabulary include: how many different molecular groups (i.e., labeled subgraphs appearing in a set of subgraphs) appear in a given molecule; or, how many distinct spanning trees occur in a molecular graph. Over 2000 molecular descriptors appear in the chemoinformatics literature, and this calculation step is complicated by economics: most chemical descriptors are available only from (expensive) commercial software, although the FDA has recently produced free software that generates some descriptors.

Once the descriptors are calculated, the authors build a multidimensional descriptor space—and finally we can talk about theoretical computer science! The multidimensional descriptor space is a metric space with each axis defined by a different descriptor. The most common metrics used on this space are the Euclidean distance, the Mahalanobis distance (which weights correlation between descriptors), and the Tanimoto coefficient, which I will present here because it is "the most popular similarity measure used in chemoinformatics," according to Clark and Roe, the authors of Chapter 5. Given two vectors $\overrightarrow{x_1} = \langle x_{10} x_{11} \cdots x_{1n} \rangle$ and $\overrightarrow{x_2} = \langle x_{20} x_{21} \cdots x_{2n} \rangle$, the Tanimoto similarity measure (coefficient) S_{TAN} is defined as:

$$S_{TAN} = \frac{\sum x_{1j} x_{2j}}{\sum x_{1j}^2 + \sum x_{2j}^2 - \sum x_{1j} x_{2j}}.$$

Recall that the dot product of two vectors divided by their geometric mean is the cosine of the angle between those vectors, which provides a measure of how far the vectors are from each other. The Tanimoto coefficient is the dot product of two vectors divided by their arithmetic mean with a correction for the size of the dot product. So it is like the cosine measure, but it provides more resolution when measuring similarity between vectors that are nearly maximum length in the descriptor space.

Tropsha and Golbraikh devote several pages to normalization of descriptors, which I will skip. Once the metric space is defined and the axes normalized, it is time to analyze clusters of data points. Computation of the distance betweeen each point may be infeasible, so the authors present the following randomized algorithm. Decide upon a minimum similarity threshold. Then choose an initial compound at random and put it into the cluster. Choose another compound at random and put it into the cluster if its similarity with every cluster element exceeds the threshold. (Otherwise, discard it.) Continue until all compounds have been considered. (This concludes the algorithm.) The authors suggest running this algorithm repeatedly on the same compounds, with significantly different thresholds, until finding a threshold that selects a cluster with the number of elements desired.

Once the cluster space is mapped, the next step is to remove outliers, so all points in a given cluster do indeed share common molecular properties. A "leverage outlier" is a point distant from all other points, even though it may lie at the geographic center of the distribution. The authors present two methods to remove leverage outliers. Once the leverage outliers are removed, one must remove "activity outliers," which are points that may be close to other points in the descriptor space, but whose chemical activity is nonetheless quite different. This step seems to rely more on chemical intuition than CS theory.

A cluster thus prepared can function as a QSAR model, for, e.g., *in silico* pharmaceutical or solvent design. To establish and validate the predictive power of such a model, the authors advocate a machine-learning procedure, "during which the model is 'trained' (i.e., model parameters are

tuned to provide the highest predictivity in terms of some statistical criterion used as a target function which is optimized during the procedure)." This is the topic of Chapter 7, which I will now briefly consider.

2.1.2 Chapter 7: Predictive Quantitative Structure-Activity Relationships Modeling: *Development and Validation of QSAR Models*

The authors of Chapter 7 (Tropsha and Golbraikh again) measure the correctness of a predictive QSAR model by running statistical tests on target functions the model is trained for. One such target function is Correct Classification Rate (CCR), defined by $CCR = N(\text{correct})/N(\text{total})$, where $N(\text{correct})$ is the number of compounds the model has classified correctly, and $N(\text{total})$ is the total number of compounds considered. Most of the measures of the target function are well-known statistical tests, like leave-one-out cross-validation.

Rather than just test a particular descriptor space that is optimized in a particular way, the authors recommend the combinatorial QSAR approach (combi-QSAR). In combi-QSAR, one starts with a single dataset and produces several different descriptor spaces, with different collections of descriptors. Then take each descriptor space, and consider its behavior under different optimization methods. Then choose the best models out of all these, and take the consensus predictions of those best models. The authors claim this approach adequately address the criticisms leveled against the QSAR literature, and will allow QSAR to be a significant tool for drug exploration and discovery in the 21st century.

2.2 Computer-Aided Molecular Design

Other than QSAR, the principal application considered in HCA is Computer-Aided Molecular Design (CAMD). Visco, the author of Chapter 9, explains CAMD as follows.

> CAMD is the application of computer-implemented algorithms that are utilized to design a molecule for a particular application.... [W]hen the desired molecules are for biological systems, the solution space is estimated to be at least 10^{60}, which is a relatively tiny fraction of the space as a whole.... Accordingly, efforts are made *a priori* to limit the search space using techniques such as full-fledged templating or requiring the presence of certain features in a candidate molecule.
>
> The two most visible industries using CAMD are the chemical process industry and the pharmaceutical industry. While both industries are solving CAMD problems, they differ in substantial ways. For example, the chemical process industry regularly uses CAMD in the area of solvent design. Solvents are designed to have certain properties for applications in a particular area and outputs of CAMD algorithms are scored based on predicted properties.... In the pharmaceutical industry, CAMD is often used in a *de novo* approach.

Chapter 9 focuses on molecules designed using QSAR. Chapter 10 focuses on molecules designed *de novo* (Latin for "from the beginning"), i.e., building compounds from scratch to fit a target receptor.

2.2.1 Chapter 9: Computer-Aided Molecular Design: *Inverse Design*

Historically, the first CAMD algorithms combined functional groups (predefined molecular subgraphs) to generate a larger structure. This suffered from combinatorial explosion, as the algorithms produced many nonfeasible structures. A more feasible approach, *inverse QSAR* (iQSAR) began to be studied in the early 1990s. In iQSAR, "rather than using a QSAR to score potential molecules created from combining groups, one fixes a desired value (or range of values) and attempts to solve for the set of independent variables (descriptors) that satisfy the QSAR." The first step in this general approach is to determine the pool of molecular fragments that will be used to build the lager structures. Then one builds candidate structures and evaluates their fitness.

Visco, the author of Chapter 9, presents several algorithms used in Hybrid-CAMD, a popular approach which "combines the generate and test paradigm of previous techniques with inclusion of higher-order information (such as molecular connectivity through topological indices)."

The author then considers CAMD as an optimization problem. This approach minimizes the differences between the desired property and the prediction of the candidate structure's possession of that property. Visco presents a Mixed Integer Linear Programming (MILP) algorithm for CAMD, and discusses the application of MILP to designing soybean oil products and a new pharmaceutical with a penicillin backbone. Finally, Visco considers *CAMD using signature*, a method that structures molecular generation by providing a database of atomic "signatures" and a molecular "signature" that is the sum of the atomic signatures that appear in that molecule. Producing a molecule of interest can then be done by solving a set of Diophantine constraint equations that provide a minimum and a maximum for each atomic signature allowed to appear in the molecule one wishes to design.

2.2.2 Chapter 10: Computer-Aided Molecular Design: *de novo design*

Roe, the author of Chapter 10, begins with an high-level explanation of *de novo* design, and its advantages and drawbacks with respect to iQSAR.

> PubChem, the largest database of known chemicals, has a little more than 19 million compounds to date.... Even combinatorial libraries, which can range up in size to billions of compounds, do not begin to fully sample the range of all possible compounds. As the full compound space is too vast to search comprehensively, strategies have to be employed to search this space efficiently for the discovery of novel lead drug compounds.
>
> *De novo* design handles this challenge by building compounds from scratch to complement the target receptor. The guiding principle in this approach is that small molecules that are complementary to the target receptor, both in shape and chemical properties, will have the most specific binding.... To reduce the search of chemical space to a manageable problem, strong physical constraints must be taken into account at each step during the generation of the lead drug molecule, limiting the chemical space explored to those regions specifically complementary to the target receptor. The advantage of this approach over a virtual screening strategy [i.e., QSAR-style strategy] to identify these compounds is that the search space is directed to the relevant regions in chemical space with a far greater range and diversity of potential lead compounds that can be evaluated.... The main drawback is that the resulting compounds need to be experimentally synthesized and tested, rather than taken from an in-house inventory or ordered

commercially.

Roe presents three CAMD algorithms—Grow, Fragment-Link, and Evolutionary Strategies based on sampling—and discusses programs that currently run each. Grow has performed well when all features of a receptor are near one another, while Fragment-Link has performed better if the receptor consists of two or more subpockets separated by a large gap. Sampling strategies do not direct molecular generation explicitly toward linking with interaction sites, but rather they maximize a general fitness evaluation, so they tend to produce structures that are more stable (have lower molecular energy). However, sampling strategies have to investigate much larger search spaces than do either Grow or Fragment-Link.

Grow is a Metropolis Monte Carlo algorithm. At a given stage, a candidate to add to the structure is chosen at random and its probable binding affinity is calculated. Then a random number is generated, and if the binding affinity is greater than that random number, the candidate is added to the structure; otherwise, it is discarded and the process repeats. Fragment-Link requires more user interaction. The algorithm places binding structures in all the "hot spots" of the receptor, and then a user has to clean up the molecule through visual inspection. Evolutionary strategies are often programmed as genetic algorithms; Roe presents pseudocode for one such strategy.

3 Summary of Chapters

Chapter 1 (by Ivanciuc) discusses molecular graphs. It begins by introducing notions familiar to many SIGACT News readers (walks, adjacency matrix, Laplacian matrix, etc.), and then presents many different ways to represent two-dimensional chemical structures as graphs. For example, one can weight both vertices and edges, as well as labeling vertices with chemical symbols. Vertices then represent atoms (or, more generally, chemical groups); and edges, the bonds between those atoms. When constructing a graph for a biological (i.e., carbon-based) molecule, the vertices are weighted according to the atomic number of the atom the vertex represents, normalized so the weight of a carbon atom is 0. An edge representing a single bond is weighted 1; one representing a double bond, 2; etc.

Chapter 2 (by Misra and Faulon) considers **Algorithms to Store and Retrieve Two-Dimensional (2D) Chemical Structures**. One of the most-used approaches to build chemical databases is SMILES (Simplified Molecular Input Line Entry Specification), which produces a linear string representation of a molecule. The SMILES representation can then be parsed using string processing algorithms. (For example, the SMILES string for acetaminophen is "C(=O)[Nc1ccc(cc1)O]C".) The authors present the canonical SMILES algorithm: it has a worst-case running time of $\mathcal{O}(n!)$ if one wants to find the minimum-length SMILES string that correctly represents a molecule. The authors discuss more recently proposed notations that do not have this problem. Also, as one can see from the string for acetaminophen, SMILES only records names of molecules and their qualitative relationships. It is useful to include more information in a database, like the physical distance between pairs of atoms. To do this, more complex languages have recently come into use, like CML (Chemical Markup Language), an XML-based molecular file format hosted on Sourceforge.

Chapter 3 (by Willighagen) is titled **Three-Dimensional (3D) Molecular Representations**. The author's main concern is "how to represent 3D molecular geometries such that they are useful for analysis and computation," because a coordinate representation useful for visualization may be cumbersome tool for data analysis or pattern recognition. Therefore, much of the chapter

examines how to convert between different coordinate systems, and how to produce a numerical, fixed-length vector representation (i.e., a molecular descriptor) from a molecule's coordinate representation. Willighagen presents an algorithm to calculate the molecular center-of-mass from a Cartesian representation of the molecule; from a TCS point of view, this calculation proceeds in the "expected" way. He defines a notion of "internal coordinates," which provide a 3D description of a molecule without an external frame of reference, by describing "the molecular geometry in terms of distances between atoms and angles and torsions between bonds." He presents an algorithm to convert internal coordinates to Cartesian coordinates. Willighagen also discusses how to compare two 3D geometries: one method is to find the maximum common substructure (MCSS) shared by two molecules, and to orient the molecules by minimizing the root mean square deviation of the coordinates of their shared MCSS.

Several pages of Chapter 3 are devoted to a practical example of the use of Radial Distribution Functions (RDFs) to classify different types of crystal packing, so I'll make the following advisement to the reader. The Wikipedia article on RDFs states that RDFs are a measure of how atomic density is distributed through a molecule. However, RDFs are used in HCA in a more general way: to measure the geographic distribution of some feature in the molecule, or to evaluate how influential an atom is on its neighbors and neighbors' neighbors, etc.

Chapter 4, **Molecular Descriptors** (by Fechner, Hinselmann and Wegner), is the largest chapter in HCA, at over fifty pages. It presents multiple ways to measure structural properties with real numbers, from graph-theoretic topological indices, to hash functions on molecular graphs, to notions of edit distance, to graph kernels and kernel functions. It also presents pseudocode for several algorithms, but many of these are "introductory" from a TCS perspective, such as Dijkstra's shortest path, or DFS/BFS traversal. More interesting is the algorithm for shapelet extraction, which approximates the shape of a molecule by local approximations to its surface using hyperbolic paraboloids. (The definition of a molecule's surface is not something I will get into here.) Speaking philosophically, the authors note:

> Graph complexity can be defined in various ways, but still there is no standard definition. In [*Complexity and Chemistry: Introduction and Fundamentals* by Bonchev and Rouvray] various criteria are compiled from different sources, which describe the requirements for a "good" molecular complexity descriptor. For example, a complexity index should
>
> - Increase with the number of vertices and edges
> - Reflect the degree of connecteness
> - Be independent from the nature of the system
> - Differentiate nonisomorphic systems
> - Increase with the size of the graph, branching, cyclicity, and number of multiple edges.
>
> Still, this is an ongoing discussion, with even conflicting positions.

Rather than attempt to summarize the breadth of material in Chapter 4, I will offer an opinion. The material is presented in a style of, "This descriptor has worked for some applications, now let's move on to the next descriptor." The approach is pragmatic, and even in the more specialized *Complexity and Chemistry*, there is no construction of a mathematical theory, only informal

arguments from induction on small examples. Also, it seems, from a literature search, and two questions I asked on cstheory.stackexchange.com, that these molecular descriptors have not been studied much in TCS or in graph theory (though more general notions certainly have). Therefore, I believe there is an interdisciplinary research opportunity here.

Chapter 5 (by Clark and Roe) is on **Ligand- and Structure-Based Virtual Screening**. The authors define virtual screening as "the selection of molecules likely to show desired bioactivity from a large database." In other words, once one has assigned descriptors to molecules, the objective is to select molecules that are similar to one another. Clark contributes a section on similarity searching, in which he discusses similarity metrics, including the Tanimoto coefficient.

Roe begins by considering the *docking problem*: given a target receptor, find the molecule that docks best in that receptor (i.e., find the best-fitting peg for a hole). As Roe puts it:

> The docking problem can be broken down into three components: (1) orientational search, or the search for the 3D orientation of a molecule with respect to another; (2) conformational search, or the search through rotatable torsions; and (3) scoring, or evaluation "pose" or orientation/conformation by some measure of predicted binding.

Docking algorithms include the search for all cliques of a limited size (this search problem is tractable), in order to find sections of the receptor that match up correctly to sections of the molecule (this is an example of orientational search); and the Lamarckian Genetic Algorithm (LGA), which combines orientational and conformational search. In LGA, the genetic algorithm's chromosome is created to represent the orientation/conformation of the molecule with respect to the target receptor. This chromosome then evolves according to a fitness function. LGA is nondeterministic, and it terminates based on user parameters.

Chapters 6 and 7 are about QSAR modeling, and I have considered them already in Section 2.

Chapter 8 (by Meringer) is on **Structure Enumeration and Sampling**. The motivating questions are:

> Given a molecular formula plus, optionally, a list of structural constraints, the typical questions are: (1) How many isomers [nonisomorphic graphs satisfying these constraints] exist? (2) Which are they? And, especially if (2) cannot be answered completely: (3) How to get a sample?

The mathematical foundation used to answer these questions is Pólya's theory of counting. Meringer gives an example of Pólya's theory by counting the 22 permutational isomers of dioxin by calculating the cycle index of the binding sites of the molecular graph and then inserting a generating function into the cycle index. Meringer generalizes this approach to an algorithm and provides more examples. The author then considers the construction problem of producing all structurally distinct molecules that have the same chemical formula (i.e., all distinct isomers). He begins with the DENDRAL system, which was perhaps the first expert system of any sort, and was "one of the roots of chemoinformatics." DENDRAL (short for DENDRIC ALgorithm) was generating isomers as early as the 1960s.

Until the 1970s, only acyclic structures could be constructed. Eventually, Masinter described a cyclic structure generator, which Meringer sketches. The weakness of Masinter's generator is an inability to process structural constraints effectively, so different methods were introduced that, for example, put an ordering on the set of graphs to be generated, so it is not necessary to do

pairwise isomorphism testing. One can accelerate structure enumeration by placing constraints (like planarity, or minimum cycle size) on the graphs to be generated. The author mentions several other generation methods. Nevertheless, "it is often impossible to generate all molecular graphs belonging to a given molecular formula" due to combinatorial explosion. As a result, sampling techniques are essential, in order to produce a representative subset of the total isomer space. Monte Carlo methods, simulated annealing and genetic algorithms are most often used for this purpose. Meringer presents pseudocode for a simulated annealing algorithm and a genetic algorithm for isomer sampling. The chapter concludes with the application of constructing *combinatorial libraries*, i.e., chemical databases where either the compounds possess high structural diversity, or, alternatively, all share some common relation.

Chapters 9 and 10 are about computer-aided molecular design, and we have considered them already in Section 2.

Chapter 11 (by Faulon and Carbonell) shifts gears, and instead of focusing on molecules, it focuses on **Reaction Network Generation**. The authors explain:

> Designing new drugs or chemicals, understanding the kinetics of combustion and petroleum refining, studying the dynamics of metabolic networks, and using metabolism to synthesize compounds in microorganisms are applications that require us to generate reaction networks. As the networks may be quite large, there is a need to automate the procedure....
>
> While formal techniques are robust, their computational scaling limits their applicability to real reacting systems....[T]he number of reactions and intermediate species that are created generally scales exponentially with the number of atoms of the reactants.... With all these systems, not only the time taken to generate networks may scakle exponentially, but simulating the dynamics of large networks may be computationally prohibitive.

Note that the simulation cost is not exponential, but "only" N^3 where N (the number of chemical species) can easily be larger than 10^4 or 10^5.

The authors present a network generation/sampling algorithm, which simultaneously generates a reaction network, and also "reduces" it, i.e., removes reactions not critical to the particular task at hand. The method of applying a reduction strategy is a standard one to mitigate the otherwise exponential blowup of the reaction network, and the definition of a reduction strategy will depend on what products of a reaction are considered important. The authors first present pseudocode for a deterministic network generator, and illustrate its application to ethane thermal cracking. There is a comprehensive database of 324 elementary transformations that carbon atoms can take in organic reactions. The input of the deterministic reaction generator is "a list of reactant species, a list of constraints, and a list of elementary transformations." The authors then present pseudocode for three distinct sampling algorithms that are stochastic, not deterministic, and "samples" the total set of reactions generated at each stage, preserving only the ones considered most important. One way of measuring importance, for example, is to keep only the reactions whose reactants have the highest concentration in the system.

Chapter 12 (by Guha) covers **Open Source Chemoinformatics Software and Database Technologies**. These open-source efforts are quite recent (all starting in the 2000s). For example, the SMILES language (which as we saw above produces strings that represent chemicals in databases) has suffered from ambiguities in its original specification, and proprietary implementations of SMILES have extended it in different ways. Therefore, there is now an open-source

OpenSMILES project which intends "to explicitly define the SMILES language in a public manner. The end result is expected to be a formal grammar for the language along with reference implementations" that is completely free-of-charge. Guha compares three open source chemoinformatics toolkits (CDK, OpenBabel, and RDKit), reviewing which functionalities they provide and their development activity. All three toolkits are hosted on SourceForge. A computer scientist could read this chapter with little chemistry background.

As I mentioned in Section 1, Chapters 13-15 are applications of bioinformatics techniques to chemoinformatics. Indeed, **Chapter 13** (by Akutsu) on **Sequence Alignment Algorithms: *Applications to Glycans and Trees and Tree-Like Structures*** could be considered pure bioinformatics. Glycans are carbohydrate sugar chains that are usually found on the surface of cells. According to Akutsu, "The importance of glycans has been recognized in the field of bioinformatics since the beginning of the twenty-first century and then various studies have been carried out." There is now a publicly available glycan database called KEGG Glycan, and a search tool with alignment algorithms for that database, called KCaM (KEGG Carbohydrate Matcher). Like RNA or DNA, the alignment problem for glycans is: given two glycans, find the way to align them so there is maximum possible matching of locations. Glycans are usually represented as rooted trees, so the main theoretical tool used is tree edit distance (i.e., how many steps it takes to turn one tree into another by using only node deletions, insertions and substitutions). Akutsu presents one tree-edit distance algorithm and discusses others. The author then considers glycans as unrooted trees. Tree-edit distance becomes NP-hard in that case, so KCam algorithms have been developed that combine finding the Maximum Common Subtree (MCST, which can be done in polynomial time) with the standard bioinformatics tools of score matrices and local alignment. Akutsu discusses an MCST algorithm and presents pseudocode for both global and local glycan alignment algorithms.

Chapter 14 (by Martin) is titled **Machine Learning-Based Bionformatics Algorithms: *Applications to Chemicals***. The author first considers applications of clustering to bioinformatics and chemoinformatics, and then considers applications of classification and regression to both fields. Martin presents pseudocode for an agglomerative hierarchical clustering algorithm , and for the Jarvis-Patrick clustering algorithm. (Hierarchical clustering is $\mathcal{O}(n^2)$ time and $\mathcal{O}(n^2)$ space, while Jarvis-Patrick is $\mathcal{O}(n^2)$ time but only $\mathcal{O}(n)$ space, so Jarvis-Patrick is the preferred method used in chemoinformatics.) This chapter is readily accessible to most TCS readers, and, for purposes of this review, I will focus on the author's description of the differences between the two fields.

> [B]ioinformatics uses a much wider variety of algorithms, but...chemoinformatics uses a much broader set of data types. Bioinformatics applications use a huge variety of algorithms but are generally restricted to microarray (numerical) and sequence (string) data. Chemoinformatics applications generally use two to three clustering algorithms (Jarvis-Patrick, Ward clustering, and k-means) but use very large datasets and a huge number of chemical descriptors as their input....
>
> In [the QSAR] approach, the...protein is usually fixed and all attention is focused on the drug. In bioinformatics, it is more common to consider a network of interactions, as in the case of a protein-protein interaction network. If we combine both types of approaches, we arrive at the prospect of predicting a protein-chemical interaction network. (Although the general area of large-scale (informatics-based) protein-chemical interaction prediction is relatively unexplored, both approaches mentioned here use SVMs.)

HCA concludes with **Chapter 15** (by Mu, Bauer, Faeder and Hlavacek) on **Using Systems Biology Techniques to Determine Metabolic Fluxes and Metabolite Pool Sizes**. This chapter begins with a discussion of Flux Balance Analysis (FBA). Intuitively, suppose you know the individual reactions that take place between a cell and its exterior, and you have measured experimentally how the mass of the cell changes over time. Then, by plugging all the reactions into a "stoichiometric matrix" and setting the objective function as the (already known) mass of the cell, it is possible to determine how the reactions affect one another—to understand the relations among the cell's different biological pathways. FBA is a computational technique. An experimental technique related to FBA is to mark certain molecules with radioactive isotopes (often ^{13}C). Then, by measuring the labeling patterns created by allowing reactions to progress, one can determine the fluxes (i.e., types of interrelations) of the reactions using the isotopes. Most of this chapter is devoted to ^{13}C Metabolic Flux Analysis (^{13}C MFA). In the words of the authors, "^{13}C MFA has reached a state of relative maturity. The experiments themselves have become a routine procedure, the measurement techniques are well-established, and sophisticated mathematical evaluation algorithms are available. However, ^{13}C MFA is still a time-consuming and low-throughput process." Therefore, Chapter 15 focuses on *in silico* ^{13}C MFA. By using a database of "carbon fate maps" that simulate how carbon isotopes move within individual reactions, it is possible to program a complex system in the BioNetGen software package, and graph the results using MATLAB. The authors devote several figures to BioNetGen code for a specific example.

4 Conclusion

While *Handbook of Chemoinformatics Algorithms* is probably a must-have for computational chemists, it will only interest computer scientists who are actively looking for an interdisciplinary project, and who are willing to put in the effort required to learn the vocabulary and perspective of a different field. This review has been heavy on application-explanation but light on math, and HCA has the same flavor. I wrote a much longer review than "necessary," in an attempt to build a bridge between chemoinformatics and TCS, and I hope this bridge encourages more computer scientists to investigate this area. Chemists could use help from TCS to build a mathematical and algorithmic theory for their work, and I believe chemoinformatics, like bioinformatics, will prove to be an important source of problems for computer science. My suspicion is that a "small" increase in the mathematical sophistication of one or more of the algorithms in HCA could yield a big payoff.