

AUTOMATED SYNTHETIC FEASIBILITY ASSESSMENT: A DATA-DRIVEN DERIVATION OF
COMPUTATIONAL TOOLS FOR MEDICINAL CHEMISTRY

by

Abraham Heifets

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

© Copyright 2014 by Abraham Heifets

Abstract

Automated Synthetic Feasibility Assessment: A Data-driven Derivation of Computational Tools for Medicinal
Chemistry

Abraham Heifets

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2014

The planning of organic syntheses, a critical problem in chemistry, can be directly modeled as resource-constrained branching plans in a discrete, fully-observable state space. Despite this clear relationship, the full artillery of artificial intelligence has not been brought to bear on this problem due to its inherent complexity and multidisciplinary challenges. In this thesis, I describe a mapping between organic synthesis and heuristic search and build a planner that can solve such problems automatically at the undergraduate level. Along the way, I show the need for powerful heuristic search algorithms and build large databases of synthetic information, which I use to derive a qualitatively new kind of heuristic guidance.

Contents

Relation to published work	xi
1 Introduction	1
2 Prolegomena to any future automated synthesis planning	6
2.1 Search	6
2.2 AND/OR graph search algorithms	9
2.2.1 AO*	10
2.2.2 Learning in Depth-First Search (LDFS)	12
2.2.3 The question of cycle semantics	13
2.2.4 Proof Number Search (PNS)	15
2.2.5 PN*	19
2.2.6 Proof Disproof Search (PDS)	21
2.2.7 Depth-First Proof Number and variants (DFPN, DFPN+, DFPN(r), DFPN-TCA, and DFPN-SNDA)	21
2.3 The shapes of chemistry	25
2.4 Challenges of organic synthesis	27
2.5 Automated organic synthesis planners	29
2.5.1 LHASA and its variants (interactive synthesis planners)	29
2.5.2 Noninteractive synthesis planners	31
2.6 Heuristics	33
2.6.1 Complexity-based	36
2.6.2 Fragment-based	38
2.6.3 Machine learning and retrosynthetic analysis	40
2.7 Reaction libraries	41

2.8	The critical need for search guidance	47
3	A declarative description of chemistry	49
3.1	Definitions	49
4	Retrosynthetic search algorithms	55
4.1	Introduction	55
4.2	A Proof Number Search-based Solver	57
4.3	Constructing a Public Chemistry Benchmark	62
4.4	Results and Discussion	63
4.5	Chapter Conclusion and Future Work	65
5	Compilation of synthesis and chemical data	66
5.1	SCRIPDB	66
5.2	Discussion	71
5.2.1	Patents as a source of chemical images	71
5.2.2	Patents as biomedical literature	72
5.2.3	Patents as a reaction database	72
5.2.4	Patents as a bioisostere catalog	73
5.3	Chapter Conclusion and Future Work	73
6	Domain-specific heuristics for synthetic feasibility	75
6.1	Introduction	75
6.1.1	Humans, eh?	77
6.1.2	Objective measures of synthetic feasibility	80
6.2	Materials and Methods	81
6.2.1	Data collection	81
6.2.2	Data cleaning	87
6.2.3	Data labeling	88
6.2.4	Data modeling	90
6.3	Results	92
6.4	Discussion	95
6.5	Chapter Conclusion and Future Work	98
7	Summary & Conclusion	100

Appendices	103
A Glossary	104
B Which molecules should be built?	107
B.1 Introduction	107
B.2 System and methods	111
B.2.1 Correspondence of bound ligands	111
B.2.2 Ligand alignment	113
B.2.3 Residue cluster extraction via clique detection	115
B.3 Results and discussion	116
B.3.1 Heme	117
B.3.2 Nicotinamide adenine dinucleotide	120
B.4 Chapter Conclusion and Future Work	126
Bibliography	127

List of Figures

2.1	Cyclical AND/OR graphs. Semicircles connect paths in a single hyperedge. Double circles indicate goal states. The examples are deliberately simple; equivalent yet more-realistic examples may be generated by replacing simple arcs with larger subgraphs.	14
2.2	If a descendant node can be reached via multiple paths, then Equations 2.1 and 2.2 are no longer correct. An example schematic when a node has repeated precursors showing the (dis)proof counts will be incorrect. In this case, there are only 3 leaf nodes but the root reports a count of 4.	18
2.3	Graph history interaction problem from Kishimoto and Müller (2004). Assume node D is a loss for the player at the root. Node B is an AND node, marked by an semicircle connecting its out arcs. Nodes C, G, and H are AND nodes as well but are not marked because they have a single outgoing arc.	24
2.4	Aspirin. Vertices labeled C, H, or O denote carbon, hydrogen, and oxygen atoms, respectively. Edges drawn with single lines denote single bonds and double lines represent double bonds. Typically, bonds to hydrogen are not drawn.	26
2.5	Esterification reaction of an alcohol active site with an anhydride active site to produce an ester. Atoms in the molecular fragments are numbered to help the reader track bond changes. When atomic labels are omitted, the vertex is presumed to be a carbon atom with sufficient hydrogens to total 4 bonds. Reaction conditions have been omitted for simplicity.	26
2.6	Synthesis of aspirin (right column) from carbon dioxide, sodium hydroxide, phenol, acetic acid and ketene starting materials (left column) via the precursor molecules, salicylic acid (top mid) and acetic anhydride (bottom mid). The final aspirin-forming step is an application of the esterification reaction depicted in Fig 2.5.	26
2.7	Palytoxin, a 409-atom molecule synthesized in 1994 (Suh and Kishi, 1994). Bonds depicted as wedges indicate the bond is angled out of the plane of the paper toward the reader. Bonds depicted as dashes indicate the bond is angled away from the reader.	28

2.8	Example from Todd (2005). Structure (40) depicts the quinone Diels-Alder reaction, while (41)-(43) show natural products that had been synthesized using the quinone Diels-Alder. LHASA does not apply the quinone DA to these cases.	30
2.9	8 step synthesis from Takahashi et al. (1990). Molecular complexity proceeds nonmonotonically.	32
2.10	Figure 19 from Boda et al. (2007) depicting minimum, maximum, and average synthetic accessibility scores by five medicinal chemists. Structures are sorted by average score. Molecules of particularly wide score disagreement are labeled. Most molecules have scores of 4 ± 1 and many have ranges which overlap.	35
2.11	Figure 7 from Huang et al. (2011) depicting minimum, maximum, and average synthetic accessibility scores by five medicinal chemists. Structures are sorted by average score. Most molecules have scores of 4 ± 1 and many have ranges which overlap.	35
2.12	Example comparison of synthetic complexity measures, taken from Barone and Chanon (2001). Synthetic progress is nonmonotonic.	37
2.13	Reaction example from Pirok et al. (2006) depicting a Friedel-Crafts Acylation reaction. Additional properties specify the charge necessary at the active site for the reaction to complete. Problematic compounds are excluded with additional patterns.	42
2.14	Diels-Alder example from Wilcox and Levinson (1986). Lines 1 and 4 show two reactions. Lines 2 and 3 are the MXC and CXC, respectively, for the first reaction. Line 5 shows the maximum common substructure from the two reactions (note the activating electron-withdrawing oxygen on the dieneophile).	44
2.15	Example from Law et al. (2009). Figure (a) depicts a sample reaction, while (b) and (c) show an extracted core and extended core, respectively. Compare to the MXC and CXC from Figure 2.14. Law et al. note that the non-chemically-essential atom 2 is correctly not included in the extended core; in contrast, a bond radius approach such as (Sato and Funatsu, 1999) would have included it.	46
4.1	The benchmark target molecules. Images generated directly from the problem definition using OpenBabel O'Boyle et al. (2011a).	58
4.2	Computer-generated Atorvastatin synthesis matching the synthesis reported in Brower et al. (1992) and Roth (2002).	63

5.1	Cumulative SCRIpDB content. Although SCRIpDB includes patents from 2011, we show data through 2010, the last complete year. (a) Left panel shows the number of ChemDraw CDX structure files, and the structures described therein, available in SCRIpDB for various years. SCRIpDB contains 4,814,913 CDX files from 2001 through 2010, comprising 10,840,646 molecules. Duplicate molecules were filtered from each patent but not across patents, as described in the text. (b) Right panel shows the number of patents and details the subset containing reactions. For 2001 through 2010, SCRIpDB contains 107,560 patents, of which 25,048 contain synthetic reactions.	68
5.2	Example Markush structure from US Patent 6,268,504 (Raveendranath et al., 1999) defining a chemical class via substituent, positional, and frequency variations.	68
5.3	Structures per patent in SCRIpDB. While 34,789 patents contain ten or fewer structures, two-thirds of patents contain more than ten and 1,296 patents contain more than a thousand structures.	69
5.4	Number of CDX data files with that contain various numbers of reaction arrows.	70
5.5	Sample of search results for molecules containing an acridine substructure.	71
6.1	Chart from Boda et al. (2007) depicting minimum, maximum, and average synthetic accessibility scores by five medicinal chemists. Structures are sorted by average score. Molecules of particularly wide score disagreement are labeled. (This is Figure 2.10, duplicated here for reader convenience.)	78
6.2	Chart from Ertl and Schuffenhauer (2009) depicting the average of chemist scores for 40 test molecules, in blue. The error bars denote the standard error of the mean of scores by 9 chemists. Therefore, to translate to standard deviation the bars would need to be tripled in size. Structures are sorted by average score. Molecules of particularly wide score disagreement between the average chemist score, in blue, and their software's scores, in red, are labeled.	79
6.3	Data flow schematic depicting the processing steps in each of the collection, cleaning, labeling, and modeling stages.	82
6.4	Example synthesis from US Patent 7,238,717 (Fleming et al., 2007). "Procedure A (Synthesis of Weinreb Amides)" is described in the patent as: " A substituted anthranilic acid (24 mmol) was dissolved in acetonitrile (200 mL), 1.05 equivalents of N,O dimethylhydroxylamine hydrochloride, 1.05 equivalents of EDC, 0.05 equivalents of dimethylaminopyridine and 1.0 equivalent of triethylamine were added and the reaction was stirred at room temperature overnight. The acetonitrile was removed by rotary evaporation and the residue was partitioned between ethyl acetate and water. The organic layer was washed with brine then concentrated to a residue. The residue was chromatographed on silica gel (ethyl acetate as eluent) to give the product. Typical yields are 70-90%".	83

6.5	Available bounding boxes around geometric entities in the ChemDraw files distributed with USPTO patents.	84
6.6	Difficult visual parsing example. Which molecules are reagents and which are products, for which reaction step?	84
6.7	CDX reading example. First bounding boxes (in blue) are computed for each molecule. Then the minimum distance between fragments is measured (in green), and used to compute mean and standard deviation for inter-molecule distance. Molecules intersecting with 2 standard deviations from an arrow head or tail (dashed purple arrow) are considered possible reagents or products, and are classified as described in the text. The three bottom molecules overlap and, therefore, are grouped into an enclosing bounding box (in orange). Because this box encloses an arrow, these molecules are rejected. Similarly, because the central arrow intersects line segments (in the center black circles) we cannot assign unique direction for the arrow and it is rejected.	86
6.8	Illustrative schematic of organic synthesis. Although there are 8 reaction arrows depicted, the A→B reaction is repeated four times and it is likely that only 5 reactions were run at the bench. Assuming that A and C are starting materials, Algorithm 5 labels A and C as requiring zero steps, B requiring 1, D requiring 2, E requiring 3, F requiring 4, and G requiring 5.	88
6.9	Distribution of the labels for molecules in the training data.	93
6.10	The predictive performance on the automatically-labelled training data for the top 3 algorithms from Table 6.1 and random forest for comparison. The plots were generated with R; the linear fit, R^2 and p-values were computed using R's <code>lm</code> linear model.	95
6.11	The predictive performance on the automatically-labelled external validation data for the top 3 algorithms from Table 6.1 and random forest for comparison. The plots were generated with R; the linear fit, R^2 and p-values were computed using R's <code>lm</code> linear model.	96
6.12	The predictive performance on the manually scored data from Ertl and Schuffenhauer (2009) for the top 3 algorithms from Table 6.1 and random forest for comparison. The plots were generated with R; the linear fit, R^2 and p-values were computed using R's <code>lm</code> linear model. For comparison, Ertl and Schuffenhauer (2009) report that the r^2 among 9 chemists on these 40 molecules ranged from 0.450 to 0.892, with an average of 0.718.	97
B.1	LigAlign Flowchart. The three stages of ligand-based active site alignment, namely ligand correspondence, ligand alignment, and cluster detection, are shown along with the techniques used to accomplish each stage.	110

B.2	Topological vectors encode atom counts as a function of bond distance from a selected atom. For example, the numbers next to the atoms depict the minimal bond distance from atom A. The topology vector for B is [3, 5, 1] indicating three atoms at distance 1, five atoms at distance 2, and one atom at distance 3.	112
B.3	A sequence alignment of P450 homologs and extracted patterns, as computed by MUSCLE Edgar (2004) using default settings. The colors correspond to the default Clustal coloring scheme. Four biologically-significant regions Nebel (2006) are boxed and numbered and the conserved cysteine residue from PROSITE pattern PS00086 is marked. The first five rows comprise the input proteins; the next five rows show the subset of residues marked as conserved by LigAlign; the last two rows are the conserved residues in the results reported by Nebel.	118
B.4	A ligand-based alignment of P450 homologs showing the shared heme moiety (colored tan) surrounded by the residue clusters detected by LigAlign. The patterns labelled in Figure B.3 are distinguished by color. Pattern 1 is shown in purple; pattern 2 is dark blue; pattern 3 is green; and pattern 4 is presented in red. Clustered residues which are not members of one of these patterns are shown in light grey.	119
B.5	A rigid minimal-RMSD ligand-based alignment of the NAD ligand from the 21 NAD-binding proteins listed in Table B.1. The shared NAD ligand is colored tan. Consistent clusters of residues are shown surrounding the ligand. Hydrophobic residues (LVAGIMP) are depicted as orange lines; acidic residues (DE) are shown as pink lines. Red spheres correspond to the location of structurally conserved water molecules.	121
B.6	Fragmentation of the bound NAD ligand from 1HDR (each rigid fragment is shown in a different color), after fragmentation but before alignment to the green pivot NAD ligand in 1HDX. The NAD moieties of nicotine, nicotine ribose, nicotine phosphate, adenine phosphate, adenine ribose, and adenine are labeled.	123
B.7	A protein alignment using only the nicotine moiety of the NAD ligand (shown in a dashed circle). The shared NAD ligand is colored tan. The misalignment induced by nicotine alignment on the non-nicotine moieties is apparent. Consistent clusters of residues, as listed in Table B.2, are shown surrounding the ligand. The underlined C5 cluster was identified by flexible alignment but was not detected by rigid alignment. Hydrophobic residues (LVAGIMP) are depicted as orange lines; basic residues (RHK) are blue lines. Red spheres correspond to the location of structurally conserved water molecules.	125

Relation to published work

Chapter 4 was published as:

- “Construction of new medicines via game proof search”, Abraham Heifets and Igor Jurisica, AAAI, (2012). Heifets and Jurisica (2012a). Reprinted (adapted) with permission from Heifets and Jurisica (2012a). Copyright 2012 Association for the Advancement of Artificial Intelligence.

The material in chapter 5 was published as:

- “SCRIPDB: a portal for easy access to syntheses, chemicals and reactions in patents ”, Abraham Heifets and Igor Jurisica, Nucleic Acids Research, 40:428-433 (2012) Heifets and Jurisica (2012b). Reprinted (adapted) with permission from Heifets and Jurisica (2012b). Copyright 2012 Oxford University Press.
- “Diversity as a first-class ranking measure in automated bioisosteric replacement”, Abraham Heifets and Igor Jurisica, Abstracts of Papers, 242nd ACS National Meeting(2011) Heifets and Jurisica (2011). Adapted with permission from Heifets and Jurisica (2011). Copyright 2011 American Chemical Society.

Chapter 6 is in manuscript preparation as:

- “Synthetic feasibility without human judgment”, Abraham Heifets, Andrea Vargas, and Igor Jurisica.

Appendix B was published as:

- “LigAlign: Flexible ligand-based active site alignment and analysis ”, Abraham Heifets and Ryan Lilien, Journal of Molecular Graphics and Modeling, 29(1):93-101, (2010) Heifets and Lilien (2010). Reprinted (adapted) with permission from Heifets and Lilien (2010). Copyright 2010 Elsevier.

Chapter 1

Introduction

Paclitaxel, a chemotherapy used for breast and ovarian cancer, occurs in the bark of the Pacific yew tree (*Taxus brevifolia*). Pacific yew tree bark contains so little paclitaxel that ten trees must be sacrificed to produce one course of chemotherapy. Unfortunately, the Pacific yew is both ecologically threatened and too slow-growing to be cultivated. Therefore, harvesting paclitaxel is - at best - a temporary solution until the Pacific yew is wiped out and puts the lives of cancer patients directly at odds with environmental stewardship.

If we can construct paclitaxel artificially, we are freed from the need to gather naturally-occurring molecules. Artificial molecule construction, or *synthesis*, is often more cost-effective and dependable than collecting wild molecules (which may be rare, hidden, laborious to harvest, or legally protected). Additionally, certain useful molecules do not exist in nature. Indeed, many modern conveniences, such as medications, dyes, fragrances, food additives, plastics, and pesticides, are synthesized rather than harvested. At the beginning of my PhD program, the ten largest pharmaceutical and chemical companies had revenues of over USD\$434 and USD\$414 billion, respectively (Cacace, 2009), giving an indirect measure of their pervasive impact.

However, to synthesize a molecule, we require a scheme for its construction from available starting materials using known chemical reactions. Analogously to cooking a meal, syntheses begin with simple commercially-available ingredients which are combined and transformed into new products. Often, as in the motivating example of paclitaxel, we have a specific molecule which we would like to construct, and the key question is how to figure out the recipe which will cook up the molecule we want. When the molecule has already been made, the recipe can be found in a synthetic database (the chemist's equivalent of a cookbook). Unfortunately, many desirable molecules, such as new medicines, have never been made and recipes must be invented to create the molecule. Indeed, a core task in organic chemistry is the *planning of organic syntheses*: the determination of a succession of chemical reactions that constructs a desired molecule from simple commercially-available starting materials.

Over the last 44 years, computational chemists have achieved important technological advances in computer tools that spawned the entire field of cheminformatics. We have significantly advanced since Corey and Wipke's pioneering paper, which detailed the repurposing of oscilloscopes to display molecules and the use of "lightpens" to avoid entering molecules on punchcards (Corey and Wipke, 1969). Corey and Wipke's system formed the basis of ChemDraw software, a closed but common chemical interchange format. Additionally, a number of popular molecular toolkits have been developed, including JChem (ChemAxon, 2011), RDKit (Landrum, 2006), OpenBabel (O'Boyle et al., 2011a), Pybel (O'Boyle et al., 2008), the Chemistry Development Kit (Steinbeck et al., 2003), and the Small Molecule Subgraph Detector toolkit (Rahman et al., 2009). In addition to providing data structures to represent molecules and reactions, these toolkits provide standard implementations to compute properties of molecules, such as molecular weight, charge, polar surface area, compute the product of chemical reactions, or to recognize important biochemical substructures such as hydrogen bond donors or acceptors. These capabilities are used in software that helps to propose new interesting molecules, such as *de novo* design or lead elaboration software (Langdon et al., 2010, Leach et al., 2006, Meanwell, 2011, Patani and LaVoie, 1996, Segall et al., 2011, Sheridan, 2002, Stewart et al., 2006, Wagener and Lommerse, 2006). Because these toolkits read and interconvert molecular data in a variety of formats and provide code to efficiently find molecular substructures in large sets of molecules, they are often used to perform operations on data from chemical databases such as ChEMBL (Warr, 2009), PubChem (Wang et al., 2009), ChEBI (Degtyarenko et al., 2008), and the Protein Databank (Berman et al., 2000).

Even computer-aided organic synthesis planners have been extensively studied. Corey and Wipke developed the first computer-aided organic synthesis planning system demonstrating that the computer could be a useful tool for the bench chemist, although their system relied on the human to decide which reaction to apply at every step (Corey and Wipke, 1969). Krebsbach et al. note that chemical synthesis planning may be thought of as an instance of AND/OR graph search, although they did not describe the use of algorithms which search AND/OR graphs efficiently (Krebsbach et al., 1998). Chemists have, at a high level, noted the analogy between the kind of reasoning used in playing chess and in planning chemical syntheses (Todd, 2005). Specifically, in both tasks, the choice between two alternative actions requires an analysis of what states each action leads us to and our available options in that imagined future. (Here, caution should be exercised as the analogy may be misleading: chess computers can beat humans by searching to some depth cut-off, even if checkmate is not yet achieved. In contrast, a chemical synthesis must be described all the way to starting materials or else the chemist does not know how to begin making the molecule.) In Section 2.5, I give a deeper discussion of these and other previous efforts to build automated synthetic planners and, in Chapter 4, I construct an automated planner of organic syntheses using modern search algorithms.

None of these previous efforts could compete with an expert chemist, evidenced by the fact that new syntheses

are still primarily planned by hand today. Automated organic planning systems “understood” how to apply many fewer reactions than a human chemist, and could analyze syntheses of smaller sizes. Additionally, many of the systems relied on *ad hoc* strategies to guide the search, which can prevent the discovery of good syntheses and have been shown to be unreliable in other domains (Campbell, 1999). To date, there have been no examples of computer-aided organic synthesis planners that use complete and efficient search algorithms. Existing heuristics have been evaluated only on small test sets and, at best, capture the opinions of a handful of chemists.

But, before we discuss search algorithms in detail, let us cover the general design of such systems. As we will see, such automated organic synthesis planners typically take three pieces of input. First, the user specifies a goal molecule for the computer system to synthesize. Second, the computer is provided with a database of available starting materials and, third, with a database of reactions that encode transformations which can be performed on those starting materials. (In contrast, a human chemist largely carries the set of available reactions and starting materials in his or her head, and queries external databases only to verify the existence of a particular example.) The computer system outputs a set of partially-ordered reactions that produce the desired goal molecule by iteratively transforming available starting materials.

While I will formalize these descriptions in Chapter 3 as Definitions 12 and 13, I will note an important point about partial-ordering because it impacts our intuitions about how to perform synthesis planning: the output is partially-ordered because, often, there is no restriction of the order of several reactions. Consider our previous example of a recipe to cook a meal, where different courses can be made separately: the cooking of the dessert can be planned and executed independently of the main course. Indeed, this partial-ordering often holds recursively: to make a pie for the dessert course, the pie crust is built separately from the pie filing. Similarly, the synthesis of a molecule often decomposes into several independent pieces. These pieces can be performed in any order, or indeed in parallel, and the work for one piece does not impact other pieces. Unlike shortest-path problems, where the output is a single path (Russell and Norvig, 1995, Chapter 3), organic syntheses are acyclic graphs. An example is shown in Figure 2.6.

Fortunately, the automated derivation of such graphs of actions has been studied extensively in the field of artificial intelligence, under the subfields of *heuristic search*, *path planning*, *game proofs*, and *automated domain-independent planning* (Russell and Norvig, 1995, Chapters 3, 4, 5, 6, 7, 11, 12, 13, 16, and 17). In these research areas, problems are abstracted into a graph. The nodes of the graph correspond to the state of the world under consideration. For example, in chess the nodes may represent different board states, whereas for a robotic car the nodes could represent different locations. The connections between nodes in the graph correspond to actions that change the world. In chess, these would be specified by legal piece moves, whereas the car’s actions might correspond to wheel velocities. Strategies for solving the problem are articulated as paths in the graph, corresponding to sequences of actions which eventually transform the world from some initial state into some desired state. As

I will show, organic syntheses can be modeled as branching plans in a discrete, fully-observable state space.

In this thesis, I describe the design of computer-aided organic synthesis planners and draw connections to proof-number based game tree search. Since the efficiency of heuristic search depends on the strength of available heuristic guidance, in Chapter 2 I describe recent developments in molecular complexity measurements that incorporate starting material availability as well as difficulty of chemistry, which can be used to provide heuristic guidance for organic synthesis planners. Then, in Chapter 6, I discuss the critical limitation of existing heuristics for estimating the difficulty of constructing molecules: specifically, existing systems use human opinion to build their training sets. Unfortunately, in a number of fields including chemistry, human opinions have been shown to be consistent neither across experts nor across time. I address this limitation by proposing a heuristic estimator that predicts the number of synthetic steps required to produce a molecule. These synthetic step counts approximate the total difficulty of making a given molecule, and are derived objectively from syntheses in our training data.

I offer this dissertation in support of my thesis: Through the use of principled algorithms and massive datasets, it is possible to create human-caliber automatic synthetic feasibility estimators. I demonstrate this deliberately-provocative claim in two separate ways. First, I show heuristic-free retrosynthetic analysis that can solve problems from undergraduate organic chemistry exams. Second, I derive heuristics for estimating synthetic feasibility that correlate well with human-generated assessments. While the automatic tools I describe can not and should not replace expert human judgement, my hope is that they may serve as useful adjuncts - especially when the number of molecules we wish to consider are far too numerous for manual curation. In fact, the design of new syntheses is so crucial that a variety of tools to (partially) inform the design are useful; we are not obliged to create fully-automated planning systems for organic syntheses. Tools related to the planning of syntheses can be used for finding relevant starting materials, searching the literature for appropriate reactions, and rough ordering and filtering of candidate molecules via the prediction of synthetic feasibility. Therefore, even heuristics of limited scope or accuracy can have utility.

To summarize, the contributions of this thesis include

1. the first complete and principled search algorithm for synthesis planning, specifically the use of proof-number search algorithms to guide the search (Chapter 4);
2. the first public benchmark for computer-aided organic synthesis, to permit direct comparisons of alternative planners (Chapter 4);
3. the first synthetic feasibility heuristic derived from the number of steps per synthesis, rather than from subjective estimates of difficulty (Chapter 6);

4. the largest synthetic feasibility training set (Chapter 6); and
5. an analysis of synthetic feasibility estimators on substantial external validation sets (Chapter 6).

At 43,976 data points, the training data is an order of magnitude larger than the previous largest comparable data set of 3,980 molecules in Takaoka et al. (2003), and the validation set of 5,280 molecules is the largest validation set to date.

I note that many of these contributions are open-source and publicly-accessible. In contrast to computer science and bioinformatics tradition, a substantial proportion of work in cheminformatics is commercially motivated and is not publicly distributed (such as Takaoka et al. (2003), which was work done inside a company and is not available for analysis). As a consequence, science is slowed: direct comparisons are often impossible and every practitioner must build each tool from scratch. In the case that the tool is derived from empirical measures, as in this thesis, often the necessary datasets must be gathered and curated before any analysis can begin. I constructed SCRIPDB, a publicly-accessible database containing chemical structures, syntheses, bioisosteric alternatives, and biomedical literature to enable the investigations described in the preceding paragraph (Chapter 5). At the time of its deposition to the NIH's comprehensive PubChem database, SCRIPDB had over 6 million unique structures and was the 7th largest contribution of molecules to PubChem and the 2nd largest academic contribution in history. It is my hope that my contributions will provide a foundation so that future research endeavors will not begin *ex nihilo*.

Chapter 2

Prolegomena to any future automated synthesis planning

A limiting factor in the usefulness of previous synthetic planners has been the combinatorial explosion of the search space but the search algorithms previously employed have been rudimentary (Law et al., 2009). Over the last two decades, a number of important advances in search algorithms appropriate for synthetic search have been developed. For example, I will show that the derivation of syntheses can be represented as an AND/OR graph search (Nilsson, 1980, Wipke et al., 1978) since we analyze precursor molecules independently¹.

In presenting the background work for my thesis, I will begin by discussing standard heuristic search algorithms and their application to domain independent planning. Then, I will discuss algorithms that are specific to search problems with branching solutions. Finally, I will briefly cover some basic chemistry, and discuss domain-specific guidance for search algorithms and how to derive libraries of available reactions.

2.1 Search

Search is a fundamental technique in artificial intelligence, and many problems have been framed as a search for a solution in an appropriate problem formulation (Russell and Norvig, 1995, Chapters 3, 4, 6, 7, 11, 12, 13, 16, and 17). Search problems comprise a description of the world, a set of actions that can be performed to alter the world, and a preferred way for the world to be. The question that search tries to answer is what actions need to

¹For example, to make macaroni and cheese, I need to cook both the macaroni and the cheese sauce. If my scheme to cook the macaroni is totally separate from my scheme to make the sauce, then these are independent. Since a chemist synthesizes required precursor molecules in separate test tubes from abundant starting materials, the construction of the molecules proceeds independently. Contrast this with problems in other planning domains, where progress toward one goal may undo progress toward another. For example, an autonomous rover that drives to location X may lack sufficient fuel to reach location Y. Such entanglement hinders the decomposition of the problem into clearly delineated subproblems.

performed, and in which order, to convert the world from its current state to the desired world. An extraordinary number of algorithms have been devised to guide the search to find these sets of actions.

As a reminder, Russell and Norvig write:

A [search] problem is really a collection of information that the agent will use to decide what to do.

[...]

- The initial state that the agent knows itself to be in.
- The set of possible actions available to the agent. The term operator is used to denote the description of an action in terms of which state will be reached by carrying out the action in a particular state. (An alternate formulation uses a successor function S . Given a particular state x , $S(x)$ returns the set of states reachable from x by any single action.)

Together, these define the state space of the problem: the set of all states reachable from the initial state by any sequence of actions. A path in the state space is simply any sequence of actions leading from one state to another. [...]

- The goal test, which the agent can apply to a single state description to determine if it is a goal state. Sometimes there is an explicit set of possible goal states, and the test simply checks to see if we have reached one of them. Sometimes the goal is specified by an abstract property rather than an explicitly enumerated set of states. For example, in chess, the goal is to reach a state called "checkmate," where the opponent's king can be captured on the next move no matter what the opponent does.

Finally, it may be the case that one solution is preferable to another, even though they both reach the goal. For example, we might prefer paths with fewer or less costly actions.

- A path cost function is a function that assigns a cost to a path. In all cases we will consider, the cost of a path is the sum of the costs of the individual actions along the path. The path cost function is often denoted by g .

For example, in classical domain-independent planning, states and goals are expressed in the equivalent of first-order logic (Russell and Norvig, 1995, Chapter 11). Typically, for a given domain, these are compiled into conjunctions of Boolean variables. States are a set of Boolean variables which encode what is true in the state, while Goals specify a conjunction of Boolean variables which must be true. Goals may also contain existentially quantified variables. Operators have two parts: first, a precondition which is a conjunction of Boolean variables

that must be true for the operator to be applied; and second, an effect which is a conjunction of Boolean variables which describes what changes are made to the state variables when the operator is applied.

As we shall see, organic chemistry can also be mapped to state space search: molecules are the states, and the operators are the reactions that transform one molecule or set of molecules into another. The desired product is the goal molecule. Some molecules can be acquired simply by purchasing them. These may be chosen as the initial states; all other molecules will need to be constructed by some set of reactions from molecules which are purchasable. In our discussion, we will only attempt to minimize the total number of synthetic steps, so the cost of every action is 1.

A search algorithm proceeds by checking whether the initial state is a goal. If it is, the search is complete. If not, alternative states need to be considered. Other states may be found by expanding the state: that is, applying the operators to the current state to generate a new set of states. The process is repeated by choosing one of these new states to investigate, and storing the other states for later evaluation.

As Russell and Norvig write:

This is the essence of search - choosing one option and putting the others aside for later, in case the first choice does not lead to a solution. [...] We continue choosing, testing, and expanding until a solution is found, or until there are no more states to be expanded. The choice of which state to expand first is determined by the search strategy. [...] Breadth-first search expands the shallowest node in the search tree first. Uniform-cost search expands the least-cost leaf node first. Depth-limited search places a limit on how deep a depth-first search can go. Iterative deepening search calls depth-limited search with increasing limits until a goal is found.

Often, we have some additional knowledge about which node to select for expansion. We can incorporate this knowledge and guide our search heuristically. If the decision of which node to expand next is the node with the best evaluation, then the expansion strategy is called *best-first search*. When the heuristic evaluation is a function that estimates the cost to reach the goal, and this heuristic function is used in a best-first search, the resulting strategy is called *greedy search*. That is, greedy search chooses to expand the state with the minimum estimated cost to reach the goal.

The basic formulation of search has been extended to more general problems. For example, nondeterministic planning extends operators to have several alternative effects (Rintanen, 2004). These capture the idea that an operator application might succeed or fail, such as an imperfect robotic arm that sometimes drops a block that it tries to lift. In such a formulation, an operator may lead from a particular state to one of several states. A complete solution for this problem should describe plans for each contingency: that is, it should specify what the robot should do if the block were lifted and what the robot should do if the block were dropped. This means that

the solution contains branches, depending on which state is encountered after an operator application. Since we land in different states, the solutions from each branch may need to be different.

Often, nondeterministic planning assumes that the various outcomes of an operator application happen with some nonzero probability, although these probabilities may not be known to the planner. In this concrete example, this means that the robotic arm can repeatedly try to pick up the block and drop it, until it eventually succeeds. In other words, the solution may contain cycles as well as branches. However, in alternative problem domains, nondeterminism may need to be modeled without probabilities: for example, in a 2-player game, the choice of move by the other player may be modeled by nondeterminism because it transitions the game to one particular game state out of a set of possible states, but the other player chooses their moves adversarially rather than randomly.

Finding branching solutions to problems such as nondeterministic planning requires algorithms that are distinct from the single path generating algorithms of traditional search. We now investigate such algorithms.

2.2 AND/OR graph search algorithms

AND/OR graph search (Nilsson, 1980, Wipke et al., 1978) has been previously applied to planning problems such as Markov Decision Problems (MDPs) (Bonet and Geffner, 2006) and the derivation of winning strategies in two-player games (Allis et al., 1994, Nagai, 1998, Schaeffer et al., 2007).

We define an AND/OR graph as

Definition 1 (AND/OR graph).

- a discrete set of states S
- a distinguished initial state $s_0 \in S$
- for each state $s \in S$, a set of actions $A(s) \subseteq A$.
- Each action a_i is a directed hyperedge, connecting state s to set of successor states $\{s_i, \dots, s_j\} \in S$. In the literature, these connections are sometimes equivalently described by a *successor function* $f(a, s) \subseteq S$ (Russell and Norvig, 1995, Chapter 3).
- A function $cost(a, s)$ describing the cost of taking an action.
- A cost function $cost(s)$ for terminal states.

A subset of states, $G \subseteq S$, where $cost(s) = 0$ are termed *goal states* and the set of states where $cost(s) = \infty$ are called *failure states*. We assume that costs are non-negative. Goals and failures are *terminal states*, while

any other state is *nonterminal*. It is often convenient to represent the hyperedges as nodes, which permits all connections in the graph to be represented with simple (start,end) arcs. For example, in two player games, the hyperedges represent the outcome of the opponent's moves and it can be useful to represent the two players symmetrically. In the literature, these hyperedges are called *AND nodes* (I may use these terms interchangeably) and the nodes from which the hyperedges originate are called *OR nodes*.

The nodes in AND/OR graphs may be labeled as solved. Goals are always solved. An internal AND node is solved if all of its children are solved, while an internal OR node is solved if any of its children are solved. A solution graph therefore specifies (at least) one action for each non-goal node, which connect the initial state to some set of goal states.

For organic synthesis, an OR node comprises a single molecule. The hyperedges from the OR node are given by the reactions that could produce the molecule. The AND node has arcs pointing to the OR nodes that represent each required reagent molecule. In other words, the OR node representing a molecule that could be produced by 5 different reactions would have 5 arcs pointing to the required reagents for each reaction. The molecule can be produced (*i.e.*, solved) if any one of the reactions is executed but each reaction is executable only if all of its required precursors are available. Molecules in the set of starting materials are labeled as solved, implying that molecules not available as starting materials are solved if they are produced by some reaction with solved reagents.

We will formalize the notion of the unbounded synthesis problem in Definition 12 of Chapter 3. However, we can intuitively describe how a synthesis problem may be represented as an AND/OR graph. Each state in S corresponds to a different possible molecule. A synthesis problem specifies a goal molecule, which corresponds to the initial state s_0 of the AND/OR graph. The set of goal states corresponds to the catalog of available starting materials for the synthesis. The library of reactions available to the chemist correspond to the set of actions A .

2.2.1 AO*

Algorithm 1 describes AO^* , one of the earliest complete and admissible algorithms for AND/OR graph search (Martelli and Montanari, 1973). AO^* may be seen as a generalization of A^* (Hart et al., 1968). A^* finds a minimal cost path from the initial state to the goal state. AO^* finds minimal cost trees from the initial state to a set of goal states. This generalization is necessary because AND nodes require solutions for every child node, not just for a single child. A^* and AO^* are equivalent when the searched graph only has OR nodes.

AO^* executes as follows. The partial solution graph is initialized to contain the start node. AO^* then proceeds in two phases that may be termed *find* and *revise* (Bonet and Geffner, 2006). First, an unexpanded node is found in the current minimal cost partial solution and expanded. This expansion may increase the costs of the current

Algorithm 1: Pseudocode AO^* listing as presented in (Hansen and Zilberstein, 2001)

```

1 procedure  $AO^*$ 
  Input: start state  $s$ 
2  $G' \leftarrow \{s\}$ 
3 while partial solution graph contains nonterminal unexpanded node  $n$  do
  # Expand best partial solution
4   foreach  $n' \in \text{children}(n)$  do
5     if  $n' \notin G'$  then
6        $G' \leftarrow G' \cup \{n'\}$ 
7       if  $\text{goal}(n')$  then  $f(n') \leftarrow 0$ 
8       else  $f(n') \leftarrow h(n')$ 
9     ]
  # Update state costs and mark best actions
10  $Z \leftarrow \{n\} \cup \text{ancestors-along-marked-arcs}(n)$ 
11 while  $Z$  not empty do
12   Remove from  $Z$  a node  $i$  such that no descendent of  $i \in G'$  occurs in  $Z$ .
13    $a \leftarrow \text{argmin}_{x \in \text{actions}(i)} [c_i(x) + \sum_j p_{ij}(x) f(j)]$ 
14   Mark  $a$  as the best action for  $i$ .
15    $f(i) \leftarrow c_i(a) + \sum_j p_{ij}(a) f(j)$ 
16 return marked solution graph

```

solution to be larger than alternatives. Therefore, AO^* iterates up the ancestors of the expanded node and revises each's best action to point to the lowest cost alternative. This process is repeated until the solution graph contains no unexpanded nodes, at which point the solution is complete and of minimal cost.

In more detail, AO^* executes in two alternating phases. The first phase is node expansion, which begins at the root of the search tree and traces a path to some nonterminal leaf. At each node, the path is extended by taking the arc which leads to the set of successors with cheapest total cost. When following a hyperedge, the path is extended arbitrarily by choosing amongst the unsolved successor states. Once a leaf is found, its successors are generated and added to the expanded AND/OR graph. The cost of leaves is defined as 0 for goal states, ∞ for failure, and the heuristic evaluation of the state otherwise.

The second phase is cost updating, which propagates changes in path cost up the tree. For each expanded state, AO^* needs to update its ancestors, whose value may have changed. To ensure correctness, updates are required to proceed from the leaves upward, since the cost of a node has to reflect the updated cost of its descendants. For efficiency, we do not attempt to update ancestors which are uninvolved in the current best solution. Since the current best solution is maintained by marking arcs on Line 14, this procedure is referred to as “ancestors-along-marked-arcs” in the listing of Algorithm 1. These nodes are precisely those where a cost revision would not change the final solution because there already exists a cheaper partial answer.

AO^* 's most serious practical problem is that, like A^* (Hart et al., 1968), it stores the entire graph in memory. Search spaces tend to grow exponentially with problem size so few real-world problems fit within modern mem-

ory, which limits A^* and AO^* 's applicability to toy problems. To address this shortcoming, several linear-space AO^* -like algorithms have been proposed (Bonet and Geffner, 2003, 2005, 2006). These can be understood as iterative best-first variants, analogous to the relationship that IDA^* (Korf, 1985) and $RBF S$ (Korf, 1992) hold to A^* .

2.2.2 Learning in Depth-First Search (LDFS)

Algorithm 2: Learning in Depth-First Search from (Bonet and Geffner, 2005)

```

1  $V$  (Global variable storing costs of nodes)
2  $\pi$  (Global variable storing policy mapping states to actions)
3 procedure LDFS-DRIVER
4   Input: start state  $s$ 
5   solved  $\leftarrow$  false
6   repeat
7     solved  $\leftarrow$  LDFS ( $s$ )
8   until solved
9   return ( $V$ ,  $\pi$ )
10 procedure LDFS
11   Input: node  $n$ 
12   if  $n$  is solved or terminal then
13     if  $n$  is terminal then  $V(n) \leftarrow$  cost ( $n$ )
14     Mark  $n$  as solved
15     return True
16   flag  $\leftarrow$  False
17   foreach  $a \in$  actions ( $n$ ) do
18     if [ $c(a, n) + \sum_{n' \in F(a, n)} V(n')$ ] >  $V(n)$  then continue
19     flag  $\leftarrow$  True
20     foreach  $s' \in F(a, n)$  do
21       flag  $\leftarrow$  LDFS ( $s'$ )  $\wedge$  [ $c(a, n) + \sum_{n' \in F(a, n)} V(n')$ ]  $\leq$   $V(n)$ 
22       if  $\neg$ flag then break
23     if flag then break
24   if flag then
25      $\pi(n) \leftarrow a$ 
26     Mark  $n$  as solved
27   else
28      $V(n) \leftarrow \min_{a \in A(n)} [c(a, n) + \sum_{n' \in F(a, n)} V(n')]$ 
29   return flag

```

Learning in Depth-First Search is presented as Algorithm 2 (Bonet and Geffner, 2005). $LDFS$ is similar to IDA^* , in that CPU time is increased to reduce memory requirements. Whereas the entire search tree is saved in AO^* , in $LDFS$, nodes are re-expanded on the path from the root to the leaves and only the score for each state is saved, to guide the search on future expansions. While this incurs a certain computational overhead, in

cases where the size of the search tree exceeds available main memory, the reduction in use of slower levels of the memory hierarchy can be a net win.

LDFS is a recursive algorithm which returns whether a node is *consistent*, namely whether the cost of a node is equal to the least expensive child cost plus the cost of the transition to the child. As a side-effect, node values are brought into consistency with their children and a choice of best-child is maintained at each node, which permits policy-generation after termination of the algorithm. The base case for the recursion is described on line 10. If the node under examination is a goal, a failure terminal, or has already been solved, the node is marked as solved which prevents the node from being re-expanded unnecessarily. Furthermore, the appropriate value (0 for goal states, ∞ for failure, or the minimum of its children's value otherwise) is recorded for the node. Terminal and solved nodes are always consistent by definition and this fact is reported up the callstack.

If the node is a nonterminal node, it is checked for consistency in the block beginning on line 16. For each hyperedge defined by an action, the set of successor states is examined. If the cost of choosing the successor set $F(a, s)$ (that is, the cost of the transition plus the sum of the costs of the states) exceeds the cost of the original interior node, we know that this arc did not determine the previous cost of the nonterminal node and therefore would not change its cost, even if the cost of this set were revised. The check on line 18 lets us skip examination of these nodes in such cases.

Alternatively, lines 19-25, check that every successor state of the current action is itself consistent and cheaper than the interior node. The interior node is consistent exactly when it has an action for which all of the successors are consistent and cost less. It may seem that the second cost check (line 23) is unnecessary as it repeats the computation on line 18 but the recursive call to *LDFS* on line 21 can have an imperative side-effect of revising the cost of the child upwards.

Two aspects of this basic algorithm should be noted. First, heuristics can be incorporated by initializing $V(s)$ to a heuristically computed value for an instantiated node. That is, the cost function on line 12 may incorporate admissible heuristic estimates for the cost that solving node n will eventually incur. The second consideration is the algorithm's behavior when the search graph contains cycles. We consider this general problem in the next section.

2.2.3 The question of cycle semantics

LDFS cannot deal with cycles as the recursive call on line 21 would never terminate. *LDFS* shares this deficiency in handling cycles with *AO**. This is a critical limitation. The AND/OR graphs that model many problem domains contain cycles when the set of operators contain inverses. These cycles must be handled with some care. For example, in Figure 2.1a, the state is not a goal state (double circle), and the futility of this cycle would need

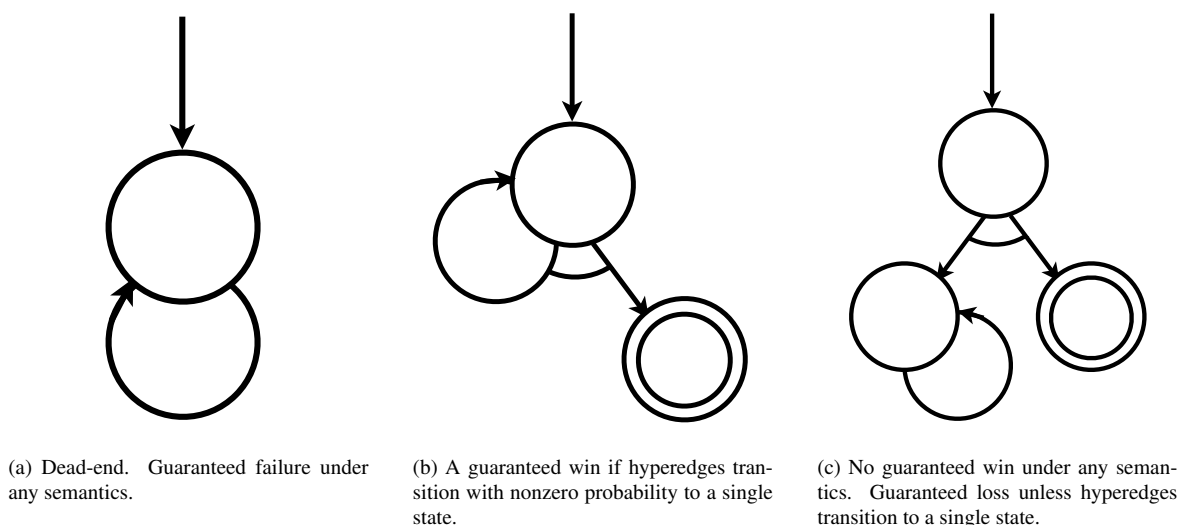


Figure 2.1: Cyclical AND/OR graphs. Semicircles connect paths in a single hyperedge. Double circles indicate goal states. The examples are deliberately simple; equivalent yet more-realistic examples may be generated by replacing simple arcs with larger subgraphs.

to be detected by algorithms searching the AND/OR graph.

Cycles occur in many domains and, in our particular domain, cycles are common and must be handled correctly. Cycles occur in the organic synthesis planning state spaces due to invertible reactions. For example, oxidations and reductions or the addition and removal of protecting groups can regenerate a molecule from itself. However, successful syntheses cannot depend on consuming a molecule in its own production.

Not all algorithmic responses are appropriate for the domain of organic synthesis planning because the semantics of what a cycle denotes can differ across problem domains. For example, *LAO** returns cyclical solutions for Markov Decision Problems by running a cyclical value revision step until the change in node values converges below some ϵ -bound or the change in node values does not alter the best action policy (Hansen and Zilberstein, 2001). This solution is not applicable to organic synthesis planning because the semantics of cycles are different. While both search spaces contain cycles, successful policies for MDPs can contain certain types of cycles. For example, consider Figure 2.1b. In an MDP, following the arc out of the start node will transition to the goal with probability p while, with probability $1 - p$, it will have no effect and leave the system at the start node. Because this transition can be taken an unbounded number of times, a policy which takes it will be successful as long as $p > 0$. In fact, the graph in Figure 2.1b is a guaranteed win because every path could be extended to reach a goal.

In an adversarial context where an opponent chooses the destination state of a hyperedge, such as in two-player games, the opponent can force the transition to always return to the start state. Therefore, there is no successful policy under an adversarial interpretation. Similarly, for organic synthesis planning, where we are obliged to provide a synthesis for every precursor molecule (that is, each state pointed to by the hyperedge) rather than just

one, a loop such as in Figure 2.1b denotes a failed synthesis.

Figure 2.1c gives an example where a probabilistic problem interpretation should return a policy with a non-zero probability of success (namely, taking the arc out of the start state). In contrast, if the graph represents an adversarial problem, a problem where we need a guarantee of success, or an organic synthesis problem, then our planning algorithm must detect that this case is a certain failure. Although Figure 2.1c is a simple example, we could replace the left and right child nodes with arbitrarily large subgraphs (either containing or lacking cycles) without changing the results of the analysis.

2.2.4 Proof Number Search (PNS)

Due to these subtleties in analyzing the domain of applicability of graph search algorithms, we now turn to algorithms designed to yield plans in an adversarial context. Game proofs, in games of first-player loss (Kishimoto and Müller, 2004), are a domain with a semantics that closely matches the semantics we require in organic synthesis. A game state is a guaranteed win for the first player if the game state is already a win or if there is a move they can play such that, for every response by the second player, the resulting game state is a guaranteed win. We can cast organic synthesis as a game in the following way: player 1 moves by picking a reaction that synthesizes the goal molecule. Then player 2 demands one of that reaction's required reagents be synthesized. The players continue to take turns until the molecule to be synthesized is in the library of starting materials. The game is a forced win for player 1 iff player 1 can construct all required reagents.

Proof number search, or *PNS*, analyzes two-player zero-sum games and determines whether a player has a forced win (Allis et al., 1994). As a side-effect, it returns the strategy that produces the victory. Unlike classical planning, this strategy is not representable as a single sequence of moves since the proper response can vary depending on the opponent's move. Instead, such a strategy can be given as a mapping from game states to moves. Allis et al. used *PNS* to give game theoretic values for Connect-Four, Qubic, and Go-Moku, demonstrating the algorithms applicability to real-world problems.

In the context of proof number search, a node is *proven* if it is a guaranteed win for player 1 and *disproven* if player 1 cannot prevent a loss. *PNS* maintains a proof number for each node, which correspond to the number of leaves in the subtree rooted at the node that would need to be examined to conclude that the node is proven as well. Similarly, each node has a disproof number that tracks the smallest number of leaves needed to be disproved for the node to be disproved as well.

Proof numbers are defined as follows:

$$\text{proof}(n) = \begin{cases} 0 & \text{if } n \text{ is terminal win} \\ \infty & \text{if } n \text{ is terminal loss} \\ 1 & \text{if } n \text{ is unevaluated leaf} \\ \sum_{x \in \text{children}(n)} \text{proof}(x) & \text{if } n \text{ is internal AND node} \\ \min_{x \in \text{children}(n)} \text{proof}(x) & \text{if } n \text{ is internal OR node} \end{cases} \quad (2.1)$$

If a node's particular game position is already won, then zero additional nodes need to be proven to prove the node, so the proof number is 0. If a position is lost, then no amount of additional nodes will suffice to prove it, so the proof number is ∞ . In the best case, an unexamined leaf node could turn out to be a won position. The proof number in this case is 1 because the leaf is the only node that had to be examined. If a node is an AND node, all of its children must be guaranteed wins for it to be proven. Therefore, the proof number of an AND node is the sum of the proof numbers of its children. Finally, it is sufficient for an OR node to be proven by a single proven child. Therefore, the minimum number of leaves to prove an OR node is the minimum of proof numbers of its children.

Disproof numbers are defined symmetrically:

$$\text{disproof}(n) = \begin{cases} \infty & \text{if } n \text{ is terminal win} \\ 0 & \text{if } n \text{ is terminal loss} \\ 1 & \text{if } n \text{ is unevaluated leaf} \\ \min_{x \in \text{children}(n)} \text{disproof}(x) & \text{if } n \text{ is internal AND node} \\ \sum_{x \in \text{children}(n)} \text{disproof}(x) & \text{if } n \text{ is internal OR node} \end{cases} \quad (2.2)$$

No matter how many additional nodes are evaluated, a won node cannot be disproven; an AND node is disproven if any of its children are disproven; and an OR node is disproven only if all of its children are disproven.

Proof number search, as given in Algorithm 3, begins with an unevaluated start leaf node. The algorithm selects a leaf node, expands it, and assigns it proof and disproof numbers in accordance with Definitions 2.1 and 2.2. The selected leaf is found by descending from the root, at each point choosing the child node that needs the least work to (dis)prove it, until the leaf is found. The choice at an interior OR node is not governed by its children's disproof numbers; since every child needs to be disproven, the order in which they are examined is irrelevant. However, it is possible that a proof will be discovered for a child during exploration, which will prove the OR

Algorithm 3: Proof Number Search from (Allis et al., 1994)

```

1 procedure proof-number-search
  Input: root node root
2 while proof(root)  $\neq$  0  $\wedge$  disproof(root)  $\neq$  0 do
3   most_proving_node  $\leftarrow$  select-most-proving-node(root)
   expand-node(most_proving_node)
   update-proof-numbers(most_proving_node)
4 return proof(root) = 0

5 procedure select-most-proving-node
  Input: Node n
6 if n is a leaf then return n
7
8 else if n is OR node then
9   return select-most-proving-node( $\operatorname{argmin}_{i \in \text{children}(n)} \text{proof}(i)$ )
10 else
11   return select-most-proving-node( $\operatorname{argmin}_{i \in \text{children}(n)} \text{disproof}(i)$ )

12 procedure expand-node
  Input: Node n
13 foreach  $c \in \text{children}(n)$  do
14   c.parent  $\leftarrow$  n
15   if c is a win then
16     proof(c)  $\leftarrow$  0
17     disproof(c)  $\leftarrow$   $\infty$ 
18   else if c is a loss then
19     proof(c)  $\leftarrow$   $\infty$ 
20     disproof(c)  $\leftarrow$  0
21   else
22     proof(c)  $\leftarrow$  1
23     disproof(c)  $\leftarrow$  1

24 procedure update-proof-numbers
  Input: Node n
25 if n is OR node then
26   proof(n)  $\leftarrow$   $\min_{i \in \text{children}(n)} \text{proof}(i)$ 
   disproof(n)  $\leftarrow$   $\sum_{i \in \text{children}(n)} \text{disproof}(i)$ 
27 else
28   proof(n)  $\leftarrow$   $\sum_{i \in \text{children}(n)} \text{proof}(i)$ 
   disproof(n)  $\leftarrow$   $\min_{i \in \text{children}(n)} \text{disproof}(i)$ 
29 if n.parent  $\neq$  NULL then update-proof-numbers(n.parent)

```

node and terminate exploration of the rest of the OR-node-rooted subtree. Therefore, child selection at interior OR nodes is the child with minimal proof number. By construction, this child will have a proof number equal to the OR node. The case for interior AND nodes is symmetric.

Once the node is expanded, its (dis)proof numbers are likely to have changed, so the proof numbers of this node and its ancestors must be updated. *PNS* walks node parent pointers, recalculating the proof and disproof numbers on the way up. This process is repeated until this root node is either proven (that is, the computed proof number equals 0) or disproven (disproof number = 0).

Allis et al. provide three optimizations for *PNS*. First, they note that the update can be terminated as soon as the (dis)proof numbers for a node do not change. This can occur whenever a node has children with tied minimal

node counts, e.g., when a node has a number of unexplored leaves. Second, if the update is terminated early in such a manner, the node where the termination occurs will remain on the path to the selected leaf node. This node's subtree contained the best node in the previous iteration, this node and all of its ancestors' counts were unaffected, so it would be selected again if we were to select a node all the way from the root. Therefore, a second optimization is possible where, when the update is terminated early, the next expansion starts immediately from the unchanged node. Finally, Allis et al. observe that the most-proving child can be determined during count updating and that caching the index of the best child during the updating makes selection a constant time operation.

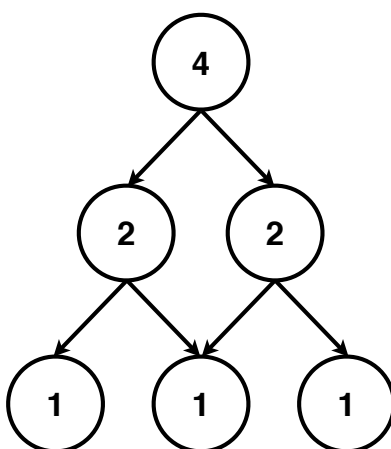


Figure 2.2: If a descendant node can be reached via multiple paths, then Equations 2.1 and 2.2 are no longer correct. An example schematic when a node has repeated precursors showing the (dis)proof counts will be incorrect. In this case, there are only 3 leaf nodes but the root reports a count of 4.

Proof number search is not without its limitations. First, Allis et al. discuss the game search space as if it were a tree. That is, Allis et al.'s implementation lacks transposition tables, rendering it unable to detect previously encountered game states (such as in loops or in transpositions), and necessitating the generation of a new node for every state. Although this state duplication is not incorrect, it is needlessly inefficient.

In many cases, games are general graphs which can contain cycles of board states when moves are reversible (*vide supra*) and where nodes can have multiple parents. When nodes have multiple parents, the update procedure of Algorithm 3 must be extended to update the node counts for every parent. Not only does the additional bookkeeping computations increase runtime overhead and storing all parent pointers increase memory consumption but the existence of multiple parents introduces subtle constraints into the computation of node counts. For example, Figure 2.2 shows a situation where the counts computed for the root node do not reflect the minimal number of leaves that must be explored to (dis)prove the node. Because the middle layer of nodes each count the shared leaf and Equations 2.1 and 2.2 are defined only in terms of the values of immediate children, the root

double counts the proof burden of the leaf. Müller (2003), Schijf (1993) proposed set-based solutions that back up from the leaves the sets of nodes to be proven. Unfortunately, the additional memory for storage of the sets and increased computation for merging and checking set membership rendered this technique impractical even for Tic-Tac-Toe.

Like AO^* , PNS keeps the entire expanded graph in memory. Allis et al. describe two methods to reduce memory usage, named *DeleteSolvedSubtrees* and *DeleteLeastProving*. The *DeleteSolvedSubtrees* technique removes all nodes in a subtree which has been (dis)proven. This removal is safe because a (dis)proven internal node will never be chosen to be expanded, so no node contained in the subtree will ever be referenced. As with the problem of maintaining correct node counts, transitioning from a tree to a graph complicates the deletion of solved nodes, since nodes may simultaneously be children of unsolved parents elsewhere in the search space. An analogy can be drawn to garbage collection in language runtimes, where cycles pose problems for reference-counting garbage collectors.

The *DeleteLeastProving* technique is not safe, even in the restricted context of tree search, and is described as a “last resort when all memory has been used”. That is, it may delete nodes which are still needed for the proof of the root node. When the search cannot continue because all of memory is in use, *DeleteLeastProving* frees the nodes least likely to be needed in the rest of the search. It walks the tree, selecting children with maximal (dis)proof number; these are the nodes that would be selected last for proving. It marks these nodes as un(dis)provable and removes them with the *DeleteSolvedSubtrees* method. Either this frees memory, and the search can continue as normal, or the root is removed and the problem is declared unsolvable within the given resource constraints.

2.2.5 PN^*

Again we see that memory consumption is a critical limiting factor in application of AND/OR search algorithms. To address this constraint, Seo et al. (2001) proposed a depth-first version of PNS , called PN^* . A depth-first traversal need store only the nodes along the current solution path, reducing the memory requirements from exponential in search depth to linear. This increases computational overhead, due to reexpansions of interior nodes, but Seo et al. report a relatively modest reexpansion ratio of 20%. Like most depth-first variants of best-first algorithms, PN^* uses a bound to limit the amount of search permitted in any subtree. Between iterations, this budget is increased, causing successively deeper explorations of the tree. For PN^* , the bound used is the proof number.

Seo et al. incorporate a number of extensions in their solver, namely *dynamic evaluation*, *dominance relations*, *killer moves*, and *recursive iterative deepening*. Dynamic evaluation carries information about proof numbers between iterations using a hashtable. If, on the current iteration, a leaf node is generated that had been previously

expanded on a previous iteration and unsolved within a proof number threshold K , then the leaf's proof number is initialized to K . Even at a depth of 1, for example, dynamic evaluation will set an AND node's proof number to the number of moves that are possible in that game state. This number is typically much higher than the default value of 1. If the leaf had been previously solved, then no benefit can be derived from solving it again, so the leaf's proof number is initialized to 0. Similarly, if the leaf had been disproved, it is initialized to ∞ . Finally, if the leaf is truly novel, it receives the default value of 1.

Dominance relations exploit the fact that many states are not independent, so solving one state provides clues on how to solve others. Seo et al. describe a domain-specific shogi dominance relation between game states A and B which describes when a solution for A guarantees a solution for B . One can make an analogy to resource constrained planning: if we plan a successful trip that uses half a tank of gas, we've also generated a plan for when we have a full tank. If such a dominance relation holds, it is clearly better to try to solve A before B and, if the proof is successful, to solve B without expanding it.

Because such a clear dominance relationship seldom holds, the authors describe a secondary heuristic technique to handle game states which are different but which differ only by the placement of irrelevant pieces. In such a situation, the move that served in a proof of one child of an AND node is the first move tried for proving the sibling nodes. If the differing pieces were truly irrelevant, the same sequence of proving moves would generate a proof for the new board state. In these cases, move generation can be skipped, which can be an important computational savings for domains where invoking the move generator is expensive. This idea had been developed earlier under the name *simulation* (Kawano, 1996) as an application of killer moves and the history heuristic (Schaeffer, 1989) to proof trees.

Traditional iterative deepening increases the search budget at the root and expands the tree until the budget is spent. PN^* uses recursive iterative deepening, which is iterative deepening at every OR node. The justification is that, since an AND node splits its search budget across all of its children, the thresholds to properly explore an AND node can be much higher than needed for its child OR nodes. Recursive iterative deepening prevents PN^* from searching too deeply the subtrees rooted at the OR node children. Although recursive iterative deepening causes increased node expansions, even relative to regular iterative deepening, the search tree reductions due to child re-ordering between iterations can reduce the total number of node expansions (Plaat et al., 1996). Combined with dynamic evaluation and killer move heuristics, recursive iterative deepening provides an informative and effective ordering on child node expansion.

As a final important optimization, Seo et al.'s implementation of PN^* uses transposition tables. Seo et al. note that the previously-discussed difficulties of generating proofs in game graphs rather than game trees occur in practice. For example, "A Big Labyrinth", problem 49 from the Edo era tsume-shogi (Japanese-chess mating problems) problem benchmark "Zuku-Tsumuya-Tsumazaruya", was not initially solved by their system due to

proof-number double counting, as discussed above for Figure 2.2. Despite these algorithmic limitations, PN^* dramatically outperformed human solvers, solving 194 out of 195 test problems and even solved a notoriously difficult problem with a solution depth of 1525 ply.

2.2.6 Proof Disproof Search (PDS)

Because PN^* only uses proof numbers, it can devote significant effort trying to generate a proof for nodes which are impossible to win and have easy disproofs. Nagai (1998) presented the first algorithm, Proof-number and Disproof-number Search or PDS , to use both proof and disproof numbers while requiring a linear amount of memory.

PDS maintains two bounds on the amount of search effort left to spend in proving or disproving a subgraph. It continues to search in a subgraph while either the spent proof effort or disproof effort is below the applicable bound. If the search fails to prove or disprove the root, a bound is increased and the search is retried. Although there are two bounds, PDS will only increase one. If the proof number of the root is less than the disproof number, PDS interprets this as evidence that a proof is easier to reach than a disproof and increments the proof bound. Symmetrically, if the node looks easier to disprove, the search will be rerun with an incremented disproof bound.

Nagai notes two optimizations. First, the initialization of (dis)proof numbers can be done heuristically. For example, an AND node will have a proof number equal to at least the number of moves available in the board state. Secondly, he notes that an OR node's proof number and an AND node's disproof numbers are essentially equivalent. Similarly, an OR node's disproof number and an AND node's proof number are treated equivalently. Therefore, he specifies a negamax implementation of PDS by specifying generalized data representations $n.\phi$, denoting the proof number if n is an AND node and the disproof number otherwise, and $n.\delta$, which is the disproof number iff n is an AND node and the proof number otherwise. This version is specialized to handle only two-player games where players alternate turns but yields better cache performance by shrinking the amount of code that must stay resident.

2.2.7 Depth-First Proof Number and variants (DFPN, DFPN+, DFPN(r), DFPN-TCA, and DFPN-SNDA)

Although PNS and PDS each consider both proof and disproof numbers in their searches, the order of node expansion by the two algorithms can differ. It is not the case that the nodes chosen for expansion by PDS correspond to the most-proving node as defined in PNS . Therefore, PDS can waste time expanding nodes that do not lie on the minimal proof tree. Nagai developed Depth-First Proof Number Search, or $DFPN$, and proved

Algorithm 4: Pseudocode for Depth-First Proof Number Search from (Nagai, 2002)

```

1 procedure Df-pn
  Input: root node  $root$ 
2  $root.th_\phi \leftarrow \infty$ 
3  $root.th_\delta \leftarrow \infty$ 
4  $search(root)$ 
5 return  $root.\phi = 0$ 

6 procedure search
  Input: Node  $n$ 
7 if  $n$  is a leaf then
8    $expand\_node(n)$ 
9 else
10  while  $\neg(n.\phi \geq n.th_\phi \vee n.\delta \geq n.th_\delta)$  do
11     $n_c \leftarrow \operatorname{argmin}_{i \in \text{children}(n)} i.\delta$ 
12     $n_2 \leftarrow \operatorname{argmin}_{i \in (\text{children}(n) \setminus \{n_c\})} i.\delta$ 
13     $n_c.th_\phi \leftarrow n.th_\delta - \sum_{i \in (\text{children}(n) \setminus \{n_c\})} i.\phi$ 
14     $n_c.th_\delta \leftarrow \min(n.th_\phi, n_2.\delta + 1)$ 
15     $search(n_c)$ 
16     $n.\phi \leftarrow \min_{i \in \text{children}(n)} (i.\delta)$ 
17     $n.\delta \leftarrow \sum_{i \in \text{children}(n)} i.\phi$ 

18 procedure expand-node
  Input: Node  $n$ 
19 if  $n$  is a win then
20    $n.\phi \leftarrow 0$ 
21    $n.\delta \leftarrow \infty$ 
22 else if  $n$  is a loss then
23    $n.\phi \leftarrow \infty$ 
24    $n.\delta \leftarrow 0$ 
25 else
26   foreach  $c \in \text{children}(n)$  do
27      $c.\phi \leftarrow 1$ 
28      $c.\delta \leftarrow 1$ 

```

that the sequence of node expansions always matches that of Proof-Number Search (Nagai, 2002). *DFPN* (with the improvements discussed in this section) has been shown to exceed human capability in a number of real-world domains: Nagai reported that *DFPN* could solve all standard Tsume-Shogi problems; Kishimoto and Müller applied it to solve more than 95% of problems in a Go liveness analysis test suite; and Schaeffer et al. (2007) proved that perfect play by both sides leads to a draw in checkers. Pseudocode for *DFPN* is given in Listing 4.

DFPN uses a negamax formulation. As we saw in *PDS*, the desired fact to be shown alternates with each level: at AND nodes, the algorithm is trying to show the node can be proven, while at OR nodes the algorithm is trying to show the node can be disproven. For either node, *DFPN* maintains a threshold ϕ for the budget left to spend showing the proof or disproof is possible. The ϕ threshold for a child node is set on line 13 to be the disproof budget of the parent node, minus the sum of proof effort already allocated to the child’s siblings. This is the maximum before the parent node exceeds its own thresholds. The fact that the ϕ bound of the child is based

on the δ bound of the parent reflects the alternation of AND and OR nodes in the game tree.

Similarly, the threshold δ is the computational budget left to show that success for the node (i.e., a proof for AND nodes and a disproof for OR nodes) is impossible. The expansion budget of the child depends on the value of the second-best node (Line 28). This prevents over-exploration of nodes when competitive alternatives exist.

$$n_c.\delta < n_c.th_\delta \leq n_2.\delta + 1 \quad (2.3)$$

$$\therefore n_c.\delta \leq n_2.\delta \quad (2.4)$$

The essence of Nagai’s proof of equivalence to *PNS* is that, since the selected child’s δ is always less than its δ threshold and the δ threshold is less than the next-best-node’s δ value (as in Equation 2.3), then transitively the selected node has minimal δ and is therefore the most-proving node (as in Equation 2.4), in the sense of *PNS*.

DFPN can accommodate the optimizations previously discussed for *PN** and *PDS*. In addition, Nagai discusses an algorithmic extension called *DFPN+* (Nagai, 2002). Usually, in these game proof algorithms, the proof and disproof numbers are initialized to 1 and playing a move is considered to be free. This can lead to very large proof trees. In *DFPN+*, the proof and disproof numbers are heuristically initialized and the costs of playing a move can be set arbitrarily. For total flexibility, these settings can be different for AND and OR nodes. By varying these settings, the shape of the search tree can be modified. For example, large move costs favor shallow searches, possibly at the cost of wider trees.

Although Nagai produced very good results (solving all standard problems in the domain of Tsume-Shogi), Kishimoto and Müller demonstrated a number of theoretical and practical issues with *DFPN* search related to our previous theme of search in directed cyclical graphs. Kishimoto and Müller observed difficulties in solving Go problems due to double counting of proof numbers in problems with cycles. When a cycle occurs, naive solutions will sum the (dis)proof numbers of leaves multiple times along overlapping paths. This counting anomaly can cause *DFPN* search to be incomplete on DCGs (Kishimoto, 2005, Kishimoto and Müller, 2008). Kishimoto and Müller construct an example where the counts are incremented in such a way as to always select a node that is not on the solution path, inducing an infinite loop. The authors propose a modification to *DFPN* search named *DFPN(r)* to avoid such double-counting and the resulting non-termination (Kishimoto, 2005, Kishimoto and Müller, 2003).

DFPN(r) partitions a node’s children into a set of *normal* children which are further from the graph root and a set of *old* children which are reachable from the root on paths shorter than the paths to the parent. Intuitively, the normal nodes are “below” the current node and the old nodes are “above” the current node in the graph. So long as a node has unproven normal children, the old children are ignored and ϕ and δ are computed based only on the normal children. If a node only has unproven old children, the node is relabeled old because its old children

must be explored for it to be solved. Therefore, the depth of the node is set to the minimal depth of its old children. This updating of depth can change the (dis)proof counts of its ancestors if the node is moved out of the set of nodes considered for computation of ϕ and δ .

This method of ignoring (dis)proof counts of certain child nodes prevents the overcounting problem but risk creating an undercounting problem when the ignored nodes actually lie on the proof tree. Completely satisfactory solutions do not exist. *DFPN – TCA* breaks the infinite loop by increasing the (dis)proof thresholds, rather than decreasing the (dis)proof numbers as in *DFPN(r)* (Kishimoto, 2010). Undercounting does not occur since the (dis)proof numbers of all nodes are considered, but the increased thresholds mean that the algorithm can spend significant time exploring regions of the search space which do not contribute to the solution of the root. *DFPN – SNDA* detects alternative paths to the same node via parent pointer chasing. The (dis)proof counts of the common ancestor are restricted to sum only one of the paths to the repeated node. In practice, however, *DFPN – SNDA* incurs a high computational overhead and still suffers from undercounting (Kishimoto, 2010).

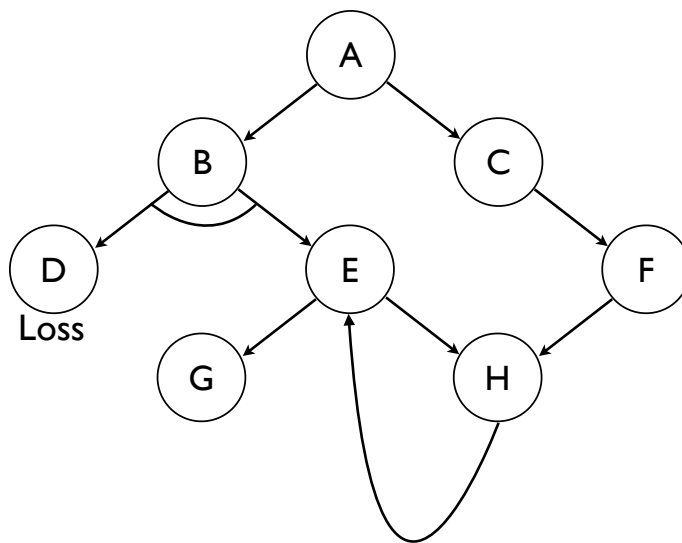


Figure 2.3: Graph history interaction problem from Kishimoto and Müller (2004). Assume node D is a loss for the player at the root. Node B is an AND node, marked by a semicircle connecting its out arcs. Nodes C, G, and H are AND nodes as well but are not marked because they have a single outgoing arc.

A separate issue concerns the issue of game proofs in the presence of repetitions. Game proofs are not commutative when cycles determine success or failure. Consider Figure 2.3 under the semantics that repetition implies a loss for the player moving at the root, who we will call Player 1. This is similar to organic synthesis where the chemist must avoid consumption of a molecule in its own creation. Also, assume that node G is a win for Player 1. If we search the sequence $A \rightarrow B \rightarrow E \rightarrow H \rightarrow E$, we find that Player 1 loses because a game

state was repeated. Now if we search $A \rightarrow B \rightarrow D$, we conclude that B is a loss because it is an AND node and a loss at any child implies a loss at B . Finally, we search $A \rightarrow C \rightarrow F \rightarrow H$, find that H has been previously determined to be lost, and conclude that C and F are lost as well. Since all of A 's children (B and C) are losses, we conclude that A is lost, despite the winning strategy $A \rightarrow C \rightarrow F \rightarrow H \rightarrow E \rightarrow G$.

Kishimoto and Müller provide a solution for the path-dependency of proofs, called the Graph History Interaction problem (Kishimoto and Müller, 2004). The authors extend Zobrist hashing encode the sequence of moves used to generate a board state. Given a maximum search depth, $MaxDepth$, and the number of possible moves in the game, $MaxMoves$, the authors create a table of size $MaxDepth \times MaxMoves$ that is initialized with random values. The fingerprint for a sequences of moves is computed as the XOR, in sequence, of the values in the table. When a new proof is generated for a board position, the proof and fingerprint are appended to the entry in the transposition table. If the board position is reached with a different path, a proof for the new path is attempted to be generated by simulating the stored proof (Kawano, 1996). Without this path-dependent technique, incorrect proof trees could be reported in cases similar to Figure 2.3.

2.3 The shapes of chemistry

We now turn to a brief description of chemistry and will discuss how to map the concepts to the state space search discussed earlier. For the problem of organic synthesis planning, the search states are molecules. The operators are reactions that are applied over sets of states. Some molecules are initially available, and every other molecule must be synthesized by finding a set of reactions which reach from initially available molecules. The synthesis problem is finding such a set of reactions for a given goal molecule.

Figures 2.4, 2.5, and 2.6 show typical depictions of a molecule, a reaction, and their use in a multistep synthesis. These representations can be seen as generalizations of domain-independent planning problems, except where planning states are typically represented using sets of Boolean variables, molecules are represented as labeled graphs. The states of organic syntheses are molecules, which are represented by graphs where vertices correspond to atoms (of elements such as carbon, oxygen, hydrogen, etc.) and edges correspond to bonds of different types (single, double, aromatic, etc.). The operators in chemistry are reactions, which describe the change in bonds that can be enacted on molecules containing appropriate reactive centers. Reaction specifications include required activating substructures, which must be present to enable a reaction, and interfering substructures, which forbid them. These required and forbidden substructures correspond to the positive and negative preconditions of planning operators.

Chemists plan syntheses using regression. In domain-independent planning, regression determines the precursor states that lead to a particular target state through the application of a given operator. Iteratively regressing

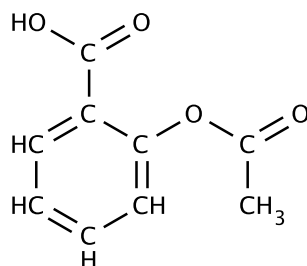


Figure 2.4: Aspirin. Vertices labeled C, H, or O denote carbon, hydrogen, and oxygen atoms, respectively. Edges drawn with single lines denote single bonds and double lines represent double bonds. Typically, bonds to hydrogen are not drawn.

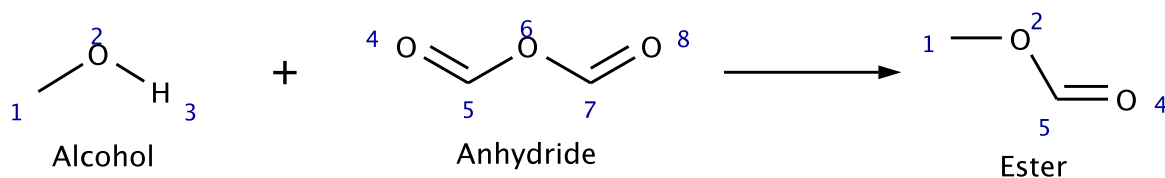


Figure 2.5: Esterification reaction of an alcohol active site with an anhydride active site to produce an ester. Atoms in the molecular fragments are numbered to help the reader track bond changes. When atomic labels are omitted, the vertex is presumed to be a carbon atom with sufficient hydrogens to total 4 bonds. Reaction conditions have been omitted for simplicity.

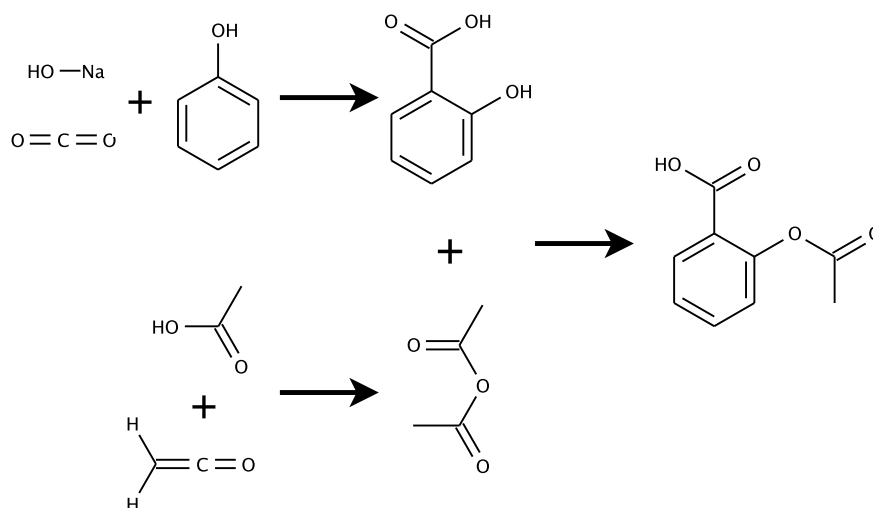


Figure 2.6: Synthesis of aspirin (right column) from carbon dioxide, sodium hydroxide, phenol, acetic acid and ketene starting materials (left column) via the precursor molecules, salicylic acid (top mid) and acetic anhydride (bottom mid). The final aspirin-forming step is an application of the esterification reaction depicted in Fig 2.5.

a goal state through a sequence of operators establishes the set of states from which the goal is reachable. When the initial state is a member of the set of precursor states, the operator sequence is a valid plan for reaching the goal.

As with operators in domain-independent planning, chemical reaction application can be considered in the forward or reverse directions. A reversed reaction application asks whether it is possible to create a given product via a particular reaction and, if so, what reagents are sufficient. These reagents may be recursively analyzed in turn until all proposed starting materials are commercially available. This recursive reversed reaction application, called *retrosynthetic analysis* (Corey and Cheng, 1995), provides an algorithm to generate syntheses. Corey's development of this analytic technique was honored with the 1990 Nobel Prize in Chemistry (Corey, 1991). Retrosynthetic analysis is conceptually similar to regression (Waldinger, 1977, Weld, 1994) except regression typically begins with a set of goal states, whereas a synthesis has a single desired product that is fully specified.

A reaction can combine multiple reagents into one product. Because each required reagent is acquired or constructed separately from the others, chemists plan the synthesis of each reagent independently. Retrosynthesis can therefore be considered a branched planning domain such as nondeterministic planning (Rintanen, 2004). Unlike Rintanen's model, the branches in retrosynthesis do not admit a probabilistic interpretation as every reagent must be created. Retrosynthetic solutions must also be acyclic. Loops in an organic synthesis denote that a molecule is consumed in its own production, and such plans are not feasible.

Branched solutions also occur in the context of game tree proofs, which establish a game-theoretic value for game positions. For example, to demonstrate a particular board configuration is a win for one player, a game tree proof provides, for every opponent move, a response that forces a win (although the responses can vary depending on the opponent's move). Game tree proofs must also be correct when repetition is possible but cycles are forbidden (Kishimoto and Müller, 2004). Proof number search and its memory-efficient variants (Allis et al., 1994, Kishimoto, 2010, Nagai, 1998) have been used to solve large real-world problems such as checkers (Schaeffer et al., 2007).

2.4 Challenges of organic synthesis

Organic synthesis presents an interesting and demanding planning domain due to the complexity of the state description, the cost of checking whether a state satisfies the goal conditions, and the number of operators and their interaction. We now discuss each of these factors in turn.

A particular challenge for computational chemistry is that molecules can be considerably larger than the states generally considered in computer game playing or automatic reasoning. To represent a molecule, the atoms, bonds, and their 3-dimensional relationships are all critical pieces of information which must be encoded in the

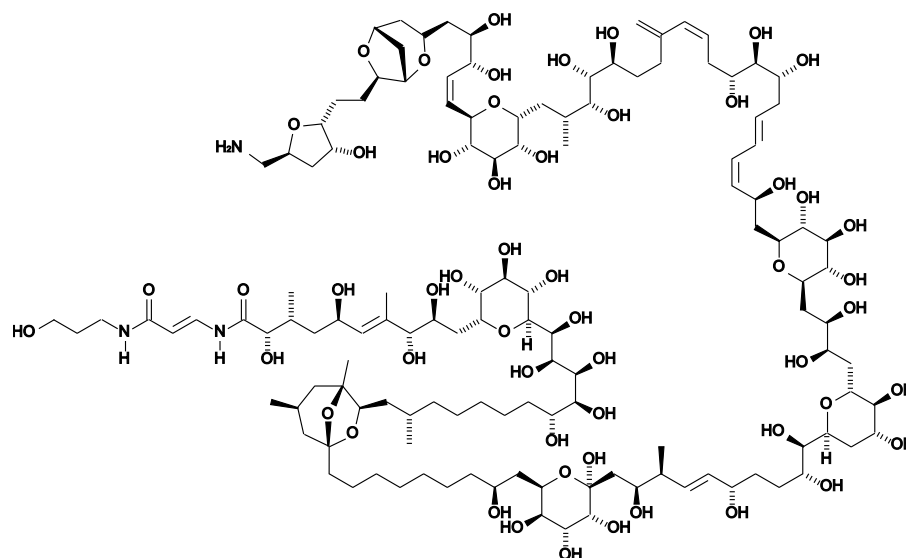


Figure 2.7: Palytoxin, a 409-atom molecule synthesized in 1994 (Suh and Kishi, 1994). Bonds depicted as wedges indicate the bond is angled out of the plane of the paper toward the reader. Bonds depicted as dashes indicate the bond is angled away from the reader.

state description. Atoms vary by element type, and bonds by bond order (*e.g.*, single, double, aromatic, etc.). Distinct configurations interact differently with their environment despite having the same internal connectivity². Therefore, varying the atomic type, bond type, or atomic configuration encodes a different molecule. Considered large among human-synthesized molecules, palytoxin (shown in Figure 2.7) has 409 atoms of which 71 can establish non-superimposable 3-dimensional configurations (Hudlicky and Reed, 2007). The state size required to encode this information is significantly larger than for other computer reasoning domains, such as checkers (Schaeffer et al., 2007), chess (Campbell, 1999), or domain-independent planning (Helmert et al., 2008).

Many different molecules have been characterized in practice. Any combination of constituent elements, graph connectivity, and 3-dimensional molecular configuration can define distinct molecules. In fact, over 30 million molecules have been deposited in the PubChem compound database. Of these, at least 14 million come from chemical vendors (Wang et al., 2009) and, therefore, are efficiently and economically synthesizable. To determine whether planning can be halted because the molecule under analysis is commercially available, an organic synthesis planner must efficiently check the molecule for membership in this set.

Chemical syntheses tend to have a length which is comparable to the search depth of modern chess engines. The length of syntheses is restricted to be relatively short because a single reaction's yield is never 100%, so long syntheses produce very little product. The average industrial pharmaceutical synthesis uses 8.1 steps, albeit

²An analogy to stereoisomers is provided by human bilateral symmetry: right and left hands interact differently with right-handed gloves, despite identical constituent fingers and thumb connected to one palm.

with a heavy tail of syntheses with 16 or more reactions (Carey et al., 2006). Complex academic targets or natural products may require more involved chemistry. For example, synthesis of vitamin B₁₂ required over 100 steps (Nicolaou and Sorensen, 1996).

Additionally, the branching factor in the search for a synthesis can be large. (ChemAxon, 2011) provides a free-for-academic-use reaction engine containing a library of 145 manually-curated commonly-used reactions (ChemAxon, 2011). This is more than the average number of moves in a game of chess or checkers, for example. Additionally, the same molecule can be created by the same reaction from different precursor molecules; similarly, in the forward causal direction, different molecules can be created by one reaction applied to one set of starting materials. This happens when the reaction operates on a substructure which shows up several distinct times in the molecules. Therefore, unlike moves in chess, one action applied to one state can produce multiple successor states. Empirically, dozens of distinct sets of precursor reagents can be proposed for each reaction (Agarwal et al., 1978). Furthermore, the reaction library can be considerably enlarged. A statistical analysis of synthesis databases estimated the existence of 92,781 unique known transformations (Law et al., 2009).

2.5 Automated organic synthesis planners

Despite these technical challenges, over a dozen synthesis systems have been constructed (Todd, 2005). These planners include interactive and automated systems, using hand-constructed, statistically-derived, or theoretically-motivated transform databases (Law et al., 2009). I focus on automated systems using transforms extracted from the chemical literature. In addition to planning organic syntheses for particular molecules (and in contrast to interactive systems), automated systems can be used to generate assessments of synthetic feasibility (Baber and Feher, 2004), search starting material catalogues (Johnson et al., 1992) or the chemical reaction literature (Tseng, 1997) for relevant related work.

2.5.1 LHASA and its variants (interactive synthesis planners)

Although it is an interactive system using a manually-constructed reaction database, the OCSS (later renamed LHASA) synthesis program is notable for founding the field of computer-assisted organic synthesis (Corey and Wipke, 1969). LHASA pioneered the algorithmic recognition of synthetically-significant molecular features, such as strategic bonds and real versus pseudo rings, as well as individual functional groups and stereochemistry. Retrosynthesis is the process of planning an organic synthesis by working backwards from the goal molecule. (For a molecule, we ask “Which precursors would we need to generate this molecule in one step, using some reaction we can perform?”) In LHASA, the retrosynthetic analysis proceeded by simplification of molecular complexity via disconnection of chains, removal of reactive functional groups and chiral centers, and a cutting of rings. Strategic

bonds are bonds which, when eliminated in the retrosynthetic direction, dramatically simplify the molecule. These were detailed manually in LHASA's transformation database, analogously although with different syntax to the transformation in Figure 2.13. Pseudorings are cycles in the molecule which a naive ring-finding algorithm might return but which are not constructed directly by known chemistry (for example, because they contain smaller rings due to "bridging" connections, as in Figures 2.7 and 2.8), and should be eliminated from further consideration. These are useful heuristics unless advanced starting materials are available and justified because LHASA lacked a starting material database.

LHASA would suggest possible alternative reactions, along with their required precursors, to the chemist. These would be prioritized according to an extensive expert-system rule base. For example, functional groups would be ranked according to reactivity and the most reactive would be removed first. Seven- to ten- member rings would be converted to five- or six- membered rings by rearrangement or elimination. Bonds to nucleophilic heteroatoms would be cleaved (because these are substructures that could be built by many different reactions, and therefore often show up in syntheses as the place of joining two simpler molecules).

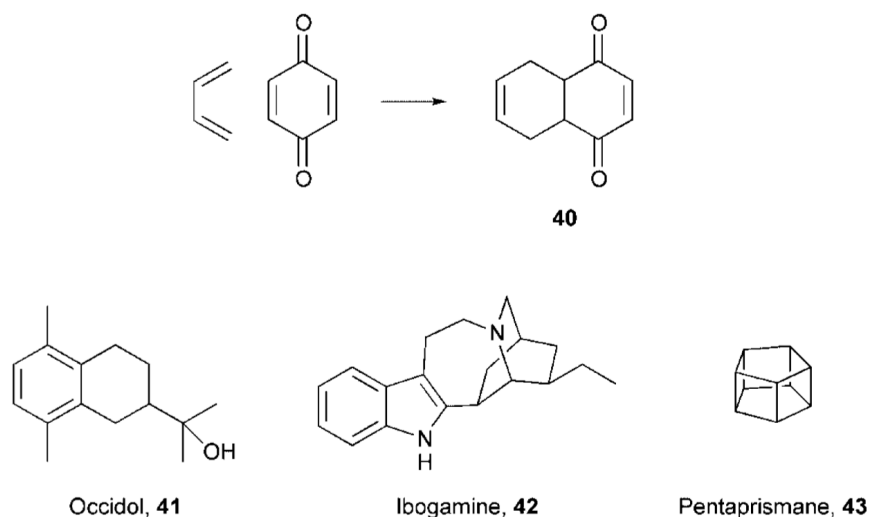


Figure 2.8: Example from Todd (2005). Structure (40) depicts the quinone Diels-Alder reaction, while (41)-(43) show natural products that had been synthesized using the quinone Diels-Alder. LHASA does not apply the quinone DA to these cases.

If a reaction was judged to powerfully simplify the molecule, but could not be immediately applied, LHASA would add subgoals to create the functional relationships necessary to permit the application of the transform. These subgoals would then be recursively analyzed, similar to Hierarchical Task Network (HTN) planning (Nau et al., 1999). In HTN planning, the plan to achieve a difficult goal is generated by recursively breaking the goal down into a combination of simpler goals. This top-down strategy also causes difficulties when the conditions under which a goal can be broken into subgoals are masked (Ott and Noordik, 1997). For example, the system may not recognize that a given transformation would advance the synthesis when the application of the transformation

requires several preparatory steps. The setup steps “hide” the fact that the transformation could be applied, and the system may never try to setup the transformation. The limiting factor becomes the skill of the transform writer to detect opportunities for appropriate and advantageous application of the transform. For example, Figure 2.8 shows a quinone Diels-Alder (QDA) transform, along with several molecules which were synthesized with it. However, LHASA was unable to recognize that the QDA retron in these molecules. It did not select it as a transform, nor explore the subgoals necessary to permit its application.

Johnson et al. present a starting-material-aware extension to the LHASA system (Johnson et al., 1992). The principle feature is automatically calculating mappings between the target and starting molecules. The system can compute different mappings of atoms in target material to atoms in the user-chosen starting material. Also, the search system can utilize the operator-chosen starting material goal to prune the search tree by only extending the search of the node most similar to the starting material. Unsurprisingly, clever unintuitive mappings can lead to dramatically cleaner syntheses.

2.5.2 Noninteractive synthesis planners

SYNCHEM was the first noninteractive synthetic planner (Agarwal et al., 1978, Gelernter et al., 1977). The molecular representation and functionality detection followed the design of LHASA and, while SYNCHEM could not detect stereochemistry, its successor, SYNCHEM2, could. Both systems were quite sophisticated in their chemical understanding; they could perceive differences in relative functional group reactivity, disfavored intermediates due to structural violations, and the relative arrangements of rings. Whereas (due to limitations of the storage systems of the time) LHASA did not represent available starting materials, one of SYNCHEM’s important advantages was the ability to discover useful starting materials in databases too large to be investigated by humans. At the time, the starting materials database was a 3000-item subset of Aldrich Chemical Company’s catalog, which was selected because it was available on punched cards (Gelernter et al., 1977).

SYNCHEM was the first description of heuristic greedy search for organic synthesis planning. SYNCHEM used heuristic estimates of both the probability of reaction completion and the complexity of precursor molecules to guide its search. Each reaction is augmented with a set of tests on the goal molecule, the results of which modify the estimated merit of the reaction. The paper gives the examples of detecting the presence of activating groups, which would raise the estimate of the probability of reaction completion, whereas a cramped molecular configuration which hindered access to the reactive site would lower the reaction probability. The frontier of the search was stored and, if the leading synthetic route encountered difficulties from dodgy reactions or complicated precursors, exploration of the line would be temporarily paused and alternatives explored. In this way, the system would avoid local minima in the synthetic route space.

SYNCHEM was later rewritten to exploit the parallelism afforded by a cluster of nine Sparc20 workstations, each with four 50MHz processors and 64MB RAM (Krebsbach et al., 1998). The parallel speedup was sublinear, but often 3-6x faster. Although effective branching factors of several dozen were observed, a significant amount of repetition was found in the synthetic routes explored. Approximately 35% of nodes in a synthetic tree were duplicates, implying that the search space is often a graph, in practice. If duplicates were very rare, then algorithms that are appropriate for trees would yield reasonable answers in many cases. Empirically, duplicates appear often enough that we would expect that treating the graph as a tree would be incorrect or, at least, inefficient. The master node had to ensure that worker nodes did not waste time synthesizing previously explored molecules and this communication effort quickly became the process bottleneck.

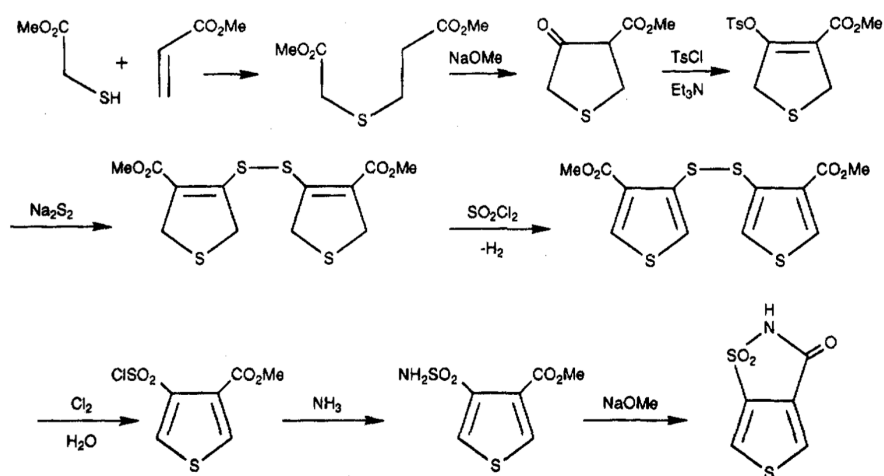


Figure 2.9: 8 step synthesis from Takahashi et al. (1990). Molecular complexity proceeds nonmonotonically.

Takahashi et al. (1990) encoded a number of chemical pruning rules to help combat the combinatorial explosion inherent in state-space search. Generated substructures were checked for significant instability or limiting reactivity, and reactions could specify required unsaturation, atomic number, quantity of attached hydrogen and kind of heteroatoms in the neighborhood around the reaction center. Such rules are sufficient to filter out, for example, Bredt's rule violations (Appendix A). The user could request transforms to be pruned if they do not functionalize a saturated atom, make carbon-carbon bonds or heterocyclic rings or if they only add or remove a simple group such as a methyl, ethyl, or tert-butyl. With this setting, the user specified the number of protection and deprotection steps permitted in a synthesis. The user could also limit the number of non-stereo-selective or regiospecific reactions in the entire synthesis and the number of facilitating reactions permitted at one site. The user optionally specified bonds or atoms which should not be affected by the synthesis. Similarly, the acceptable starting materials or intermediates could be filtered by limiting the molecular size, elements, function groups, rings, chiral atoms, and so on. Using these controls, most chemical nonsense can be avoided and relatively long

syntheses were discovered noninteractively, for example as in Figure 2.9.

In Tseng (1997), Tseng describes his use of MDL's RXL Browser to design a synthesis for a compound of agrochemical interest. If we are to take Tseng's walkthrough as a canonical description of a chemist's workflow, we would conclude that chemists operate by performing a database search for a substructure of their target, finding a specific previous reaction which generates it, and trawling the literature for a published reaction which is similar enough in mechanism to successfully generate their desired compound when run on appropriate starting materials. In such a scenario, they do not design a synthesis from scratch and, surprisingly, the key mental tools are not retrosynthetic disconnections (an assumption made by Corey when he established computer-aided organic synthesis) but rather judgements about the independence of functional groups and the cross-applicability of published reactions.

RouteDesigner is a recently developed organic synthesis planner (Law et al., 2009). Like the planners that preceded it, RouteDesigner accepts a molecule as input, detects molecular properties such as aromaticity or protonation, and adds the annotated molecule to the search queue. The first molecule is popped from the search queue and all valid retrosynthetic transforms are applied. The generated precursors are subjected to pruning rules like in Takahashi et al. (1990). Precursors are checked for duplication to molecules already in the search queue and in the starting material library and, if they are novel, added to the search queue. Unless the queue empties, the search proceeds exhaustively until a depth limit is exceeded.

The focus of RouteDesigner is in the automated extraction of transforms from the chemical literature (*vide infra*). Briefly, reactions are clustered by their "graph edits", that is, the changes in atoms and bonds enacted onto the reagents to generate the products. The reactions within each cluster are then analyzed for consistency of neighborhood around the modifications. These consistent substructures are putatively considered to be activating substructures necessary for the reaction to proceed. Unfortunately, unlike the manually constructed rules which can be refined to respect interfering groups and stereo- and regio- selectivity, the automatic extraction does not consider these reaction aspects. Furthermore, automated reaction extraction assumes access to large corpora of reported chemical reactions. Unfortunately, commercial vendors sell limited search access to reaction databases; reaction databases which are free and permit analysis over the entire dataset, rather than returning results per query, do not exist.

2.6 Heuristics

The effectiveness of heuristic search greatly depends on the quality of heuristics used. For example, the development of efficient relaxed plan heuristics (Hoffmann and Nebel, 2001) led heuristic search to become the predominant technique for automated planning (Bryce and Kambhampati, 2007) and the success of Schaeffer

et al. in solving checkers was predicated on access to accurate board analysis by Chinook, a checkers playing program stronger than all living human players (Schaeffer, 2009).

Corey and Cheng (1995) describe a number of fundamental criteria to determine which transformation to explore in a retrosynthetic analysis. Although the measures are described as pertaining to which reaction to choose, the selection is based on the simplifying effect the reaction achieves on the molecule being analyzed. Furthermore, different strategies, *e.g.*, the transform-based, structure-goal, or topological, can be used to simplify a molecule along one measure of complexity, such as functional group reactivity, without modifying others, such as cyclic connectivity (Corey, 1991).

Because Corey and Cheng write for human chemists, the assessment of molecular complexity is often left implicit; unfortunately, a computational synthesis does not have this luxury. An automated planner of organic synthesis could search for syntheses blindly but, as in the examples from classical search above, the search frontier grows so quickly that only simple problems could be solved. Alternatively, an automated organic synthesis planner could guide its search using appropriate heuristics. Such heuristics inform the planner how much effort is left to build any particular molecule in its search frontier. These heuristic estimates are used to guide the search by favoring the investigation of molecules which should be easy to synthesize and delaying the investigation of molecules which are expected to be hard to synthesize.

A number of conceptually-different approaches have been proposed for the algorithmic assessment of a chemical structure's synthetic accessibility, which Baber and Feher (2004) categorize into *complexity-*, *chemistry-*, *starting material-*, *retrosynthetic-*, and *neural network-*based estimation. Briefly, complexity-based assessment calculates the number of synthetically-difficult structural features present, such as fused rings or stereocenters, and computes a score for the molecule based on the structural intricateness. Chemistry-based assessment functions as a preprocessing filter for complexity-based measures. They identify portions of the molecule for which there exist known synthetic routes and reduce or eliminate the fragment's contribution to the overall synthetic burden of the molecule. Starting material-based estimation relates the portion of the desired molecule that can be covered by commercially-available molecules. Straightforward implementations require exact substructure matches whereas more sophisticated variants accept near-equivalences (defined either structurally or chemically). Retrosynthetic measures produce a score based on the length or yield of a generated synthetic plan to produce the molecule under analysis. Retrosynthetic approaches tend to be more computationally expensive and imply a bootstrapping problem if they are to be used as heuristic guidance in a retrosynthetic planner.

Finally, what Baber and Feher term "neural network-based" assessment covers any machine learning based prediction of synthetic accessibility from molecular features. The authors postulate that, given a training set of pairs of molecules and scores of synthetic difficulty, predictors could be trained to score novel molecules. Baber and Feher provide no examples of this category, the absence of which they ascribe to a lack of training data.

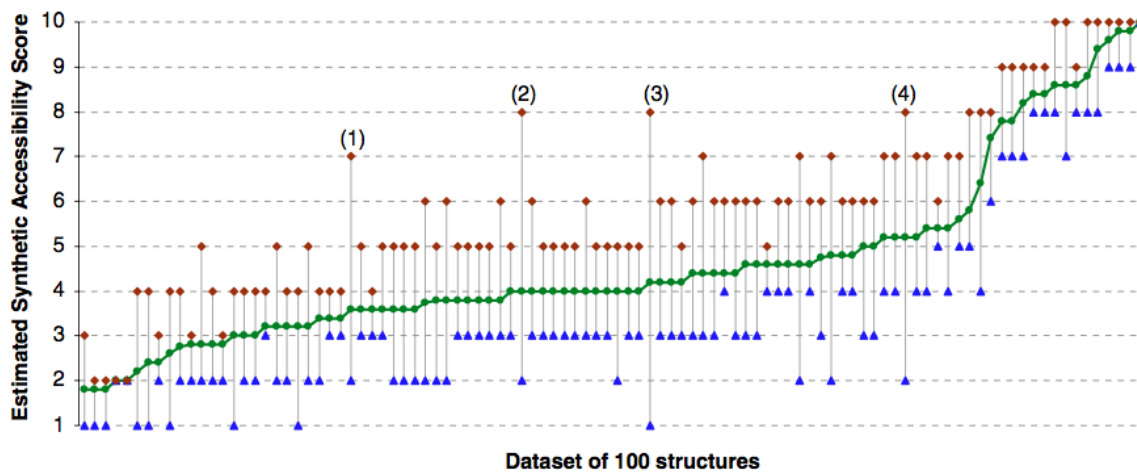


Figure 2.10: Figure 19 from Boda et al. (2007) depicting minimum, maximum, and average synthetic accessibility scores by five medicinal chemists. Structures are sorted by average score. Molecules of particularly wide score disagreement are labeled. Most molecules have scores of 4 ± 1 and many have ranges which overlap.

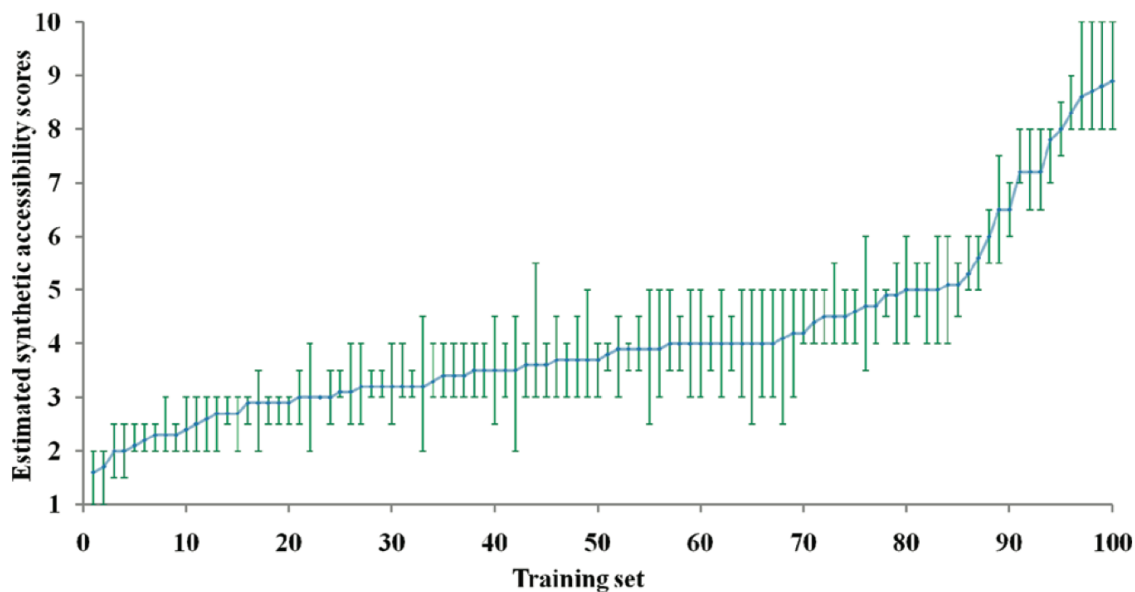


Figure 2.11: Figure 7 from Huang et al. (2011) depicting minimum, maximum, and average synthetic accessibility scores by five medicinal chemists. Structures are sorted by average score. Most molecules have scores of 4 ± 1 and many have ranges which overlap.

Not only is manual annotation prohibitively laborious, but individual chemists vary greatly in their assessments. An accurate annotation would require assessment by multiple independent chemists, as demonstrated in the large variance in Figure 2.10 and Figure 2.11. The molecular difficulty of most molecules overlap and cannot be distinguished from each other by chemists.

2.6.1 Complexity-based

Bertz compared a number of structural measures of molecular complexity, including C_r , the complexity due to rings, C_p , the complexity due to the number of atoms, and C_l , the complexity due to the number of bonds, and found that these do not exhibit fidelity to the chemist's manual determination of molecular complexity (Bertz, 1981, 1983). Bertz also defined a new measure of molecular structural complexity inspired by information theory

$$C(\eta) = 2\eta \log_2 \eta - \sum_i \eta_i \log_2 \eta_i \quad (2.5)$$

where η is the number of edges in the graphical representation of the molecule, and η_i is the number of ways a subgraph of size i can be found in the molecule. For example, for $i = 2$, it is the number of adjacent edges in the graphical representation of the molecule. That is, the number of ways a graph of three nodes connected by two edges (like a letter 'V') can be subgraph isomorphically mapped onto the molecule.

Randić and Plavić (2002) proposed a related measure of molecule complexity which is a summation of the complexity of each nonsymmetric node in the molecular graph. The complexity of each node is initialized to be the node degree. This complexity is updated by computing (for example) $1/2^d$ times the complexity of every other node in the graph, with d equal to the distance in the graph between the nodes. Randić and Plavić experiment with a number of linear and exponential decay rates for the penalty contributed to an atom's neighbor. The authors discuss how the complexity updates can be performed multiple times. Nodes which are in denser areas of the graph, which have high degree or are in a region of nodes with high degree will accumulate higher scores. In chemistry, sterically hindered atoms are difficult to construct, as are dense regions of hindered atoms. The high scores for the nodes representing these atoms reflect this construction difficult.

The preceding abstract structural measures do not directly account for chirality, symmetry, or variance in atomic element. Barone and Chanon (2001) discuss empirically derived measures intended to be more intuitive to the practicing chemist.

$$W(\cdot) = 4 * \text{number of rings} + 2 * \text{number of nonaromatic unsaturated bonds} + \\ 1 * \text{number of heteroatoms} + 2 * \text{number of chiral centers} \quad (2.6)$$

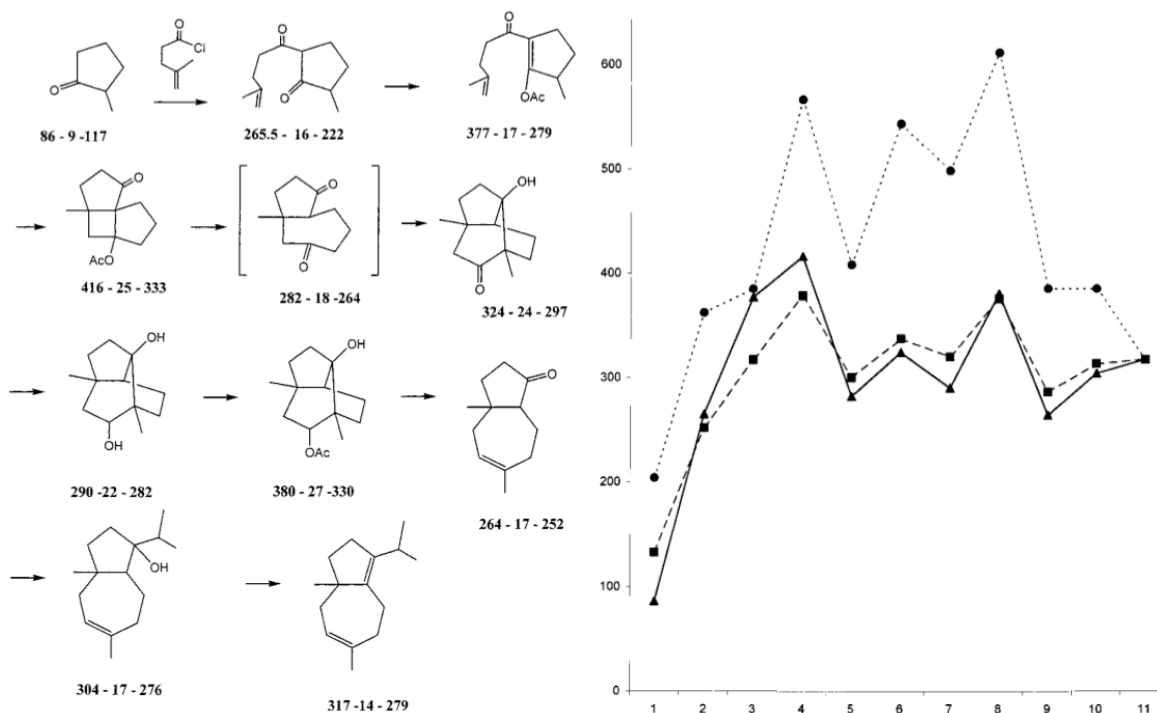


Figure 2.12: Example comparison of synthetic complexity measures, taken from Barone and Chanon (2001). Synthetic progress is nonmonotonic.

$$\begin{aligned} EW(\cdot) = & 6 * \text{number of rings} + 3 * \text{atoms with 1 heavy atom neighbor} + \\ & 6 * \text{atoms with 2 heavy neighbors} + 12 * \text{atoms with 3 heavy neighbors} + \\ & 24 * \text{atoms with 4 heavy neighbors} + 3 * \text{heteroatoms} + 3 * \text{number of atoms} \end{aligned} \quad (2.7)$$

These included the Whitlock measure given in Equation 2.6 and an extended Whitlock measure given in Equation 2.7, which was also empirically derived and manually fitted to the authors' estimation of molecular complexity. In contrast to Bertz, Barone and Chanon do not attempt to provide a formal motivation for their complexity measure.

Barone and Chanon empirically compare the complexity measures of Eq. 2.5, Eq. 2.6, and Eq. 2.7 by manually computing the complexity for each step in a number of syntheses. An example is given in Figure 2.12. As would be expected, these measures, which are all based on structural complexity, correlate with each other. Also, it is noteworthy that the complexity assessment does not rise smoothly as the synthesis progresses to the goal, pointing to limits in the accuracy of these measures. This inconsistency suggests certain structural motifs are synthetically easier (or more difficult) to achieve than is implied in their structure. Large parts of a molecule may be altered *en masse* with relatively simple reactions. For example, tosyl protecting groups comprise many atoms and contain rings and heteroatoms, yet are relatively easy to add and remove from a molecule.

Allu and Oprea attempt to correct for the fact that different molecular substructures require different synthetic effort (Allu and Oprea, 2005). Specifically, their scoring function assigns points to atoms or molecular substructures based on a number of chemical features, such as electronegativity, bond order or hybridization state, vicinal chiral centers, and ring size. However, their results show a very strong correlation between various complexity-based scores (including their score) and the molecular weight of the molecule. Therefore, it is not clear that any of the discussed measures correspond to complexity assessment as performed by the practicing synthetic chemist, who is likely to be more concerned with starting material availability, chirality of the product, and the presence of interfering functional groups that constrain the synthetic route (Tseng, 1997).

2.6.2 Fragment-based

A recently-popular technique combines aspects of chemistry- and starting material-based synthetic assessment (Boda and Johnson, 2006, Boda et al., 2007, Ertl and Schuffenhauer, 2009). The central concept is that structural rarity correlates with synthetic difficulty. In other words, structures which have been made more frequently are easy to synthesize, whereas structures that have been made rarely are difficult. The synthetic accessibility of a molecule may be a self-fulfilling prophecy rather than due to its inherent structure (as accounted for by complexity-based measures). A molecule that is frequently made will have well-developed synthetic chemistry as well as com-

mercially available precursors, both of which will reduce the difficulty of follow-on syntheses and increase the population of related molecules.

Boda and Johnson analyzed the World Drugs Index and MDL Drug Data Report databases to find frequently occurring fragments. The fragments of interest were defined *a priori* to be 1, 2, 3, or 4 adjacent nonterminal chain atoms and their neighbor heavy atoms and simple, fused, spiro, and bridged rings with and without their immediate neighbors. These fragments were arranged in an overlapping hierarchy based on atom substitutions. For example, a pyrimidine would be subsumed in a pyridine class and, transitively, in a benzene class.

$$BJ(\cdot) = 1 - w \cdot \frac{\ln(\# \text{ occurrences matched fragments})}{\ln(\# \text{ occurrences most common fragment})} \quad (2.8)$$

Equation 2.8 is computed for each fragment class and atom substitution pattern. The weight w , a penalty for not matching a particular fragment, and penalty terms for the presence of stereocenters and number of rotatable bonds may be set by the user. These factors are linearly combined into one summary score.

The atom substitution classes in Boda and Johnson (2006) provide an approximation to the fact that similar-but-distinct molecules could have similar syntheses. However, it is not the case that a pyridine could be transformed into a benzene ring. Boda et al. extend to reactions the idea that the frequency of reported structures function as a proxy for the synthetic feasibility of molecular fragments. The authors analyze a reaction database to find substructures which are frequently created in reactions. The synthetic chemist has great freedom to operate if a molecule has a large number of constructible pieces with many alternative applicable reactions. These reaction centers are matched against the molecule and a score is generated analogously to Equation 2.8.

The choice of fragments in Boda and Johnson (2006) was not explicitly motivated. Furthermore, the analyzed databases only contained drug molecules, so the derived scores could have been correlated with drug-likeness rather than synthesizability alone. Ertl and Schuffenhauer address these deficiencies by starting with a dataset of 1 million molecules from PubChem (Wang et al., 2009) and producing all fragments generated by taking a 1, 2, or 3 bond radius around any central atom. This produced 605,864 fragments, of which 51% were singletons. The score of each fragment was computed as the logarithm of the number of occurrences of the fragment divided by 484691.2. This is the ratio between the fragment count and the number of fragments forming 80% of the database, which implies frequent fragments have a positive score while rare fragments have a negative score. The score of the molecule, described in Equation 2.9, was computed as the average of the fragment scores minus a molecular complexity penalty.

$$\begin{aligned}
 ES(\cdot) = & \frac{\sum \text{fragments} \log\left(\frac{\text{fragment occurrences}}{484691.2}\right)}{\# \text{ fragments in molecule}} - \log(\text{ring bridge atoms} + 1) + \log(\text{spiro atoms} + 1) - \\
 & \log((\# \text{ rings of size} \geq 8) + 1) + \log(\text{stereocenters} + 1) - (\text{number of rings}^{1.005} - \text{number of rings})
 \end{aligned}
 \tag{2.9}$$

As is clear from Equations 2.6-2.9, all of the complexity measures described here have been heavily hand-tuned to produce results in line with the idiosyncratic synthetic assessment of their creators and testers. This is not a criticism of the measure’s usefulness but it is likely that these formulae have been overfitted to the datasets from which they derive. The small quantity of data prohibits a partition into useful independent test suites. A freely available database of syntheses could serve as the basis of such a test set, since the number of steps in the synthesis of a molecule is a direct measure of the effort to construct it.

2.6.3 Machine learning and retrosynthetic analysis

Even though Baber and Feher provide no examples of machine learning approaches to synthetic feasibility prediction, some systems have been built in the interim since publication.

The limiting factor for machine learning approaches was the lack of labelled data from which to learn model parameters. Both Podolyan et al. and Huang et al. use retrosynthetic engines to automatically construct labelled datasets, which then serve as the basis for synthetic feasibility estimating functions. The utility of estimators, given that the authors had already built retrosynthetic planners, was explained by decreasing the time required to analyze each molecule and improving the interpretability of the results by collapsing the score to a single scalar value, respectively.

Podolyan et al. (2010) create a model called RS_{SVM} . The features used to describe the molecules are the set of unique molecular fragments of width 4-6, making this approach similar to computing appropriate weights in a fragment-based synthetic feasibility estimator. RS_{SVM} casts synthetic feasibility prediction as a binary classification problem, and employs a support vector machine using a Tanimoto coefficient-based kernel to differentiate molecules which require many steps from molecules which require few. Specifically, molecules which require synthesis trees of height 1-3 are labelled ‘easy’ while molecules which were not synthesized in 5 steps of decomposition were labelled ‘hard’. Considering that the average industrial pharmaceutical synthesis is 8.1 steps (Carey et al., 2006), many pharmaceutically-relevant molecules are labelled hard.

These labels are also conditioned on the quality of the retrosynthetic analysis engine. These results were created using a 23 reaction library. This is significantly smaller than 92,781, the estimated number of unique

known transformations (Law et al., 2009), and one might be concerned that the labels are would significantly change with a different set of available reactions. To their credit, Podolyan et al. investigate the effect of removing each reaction on the labels of the molecules. For 20 out of 23 reactions, removal increased the depth of the decomposition tree and, in 13 out of 23 cases, a majority molecules that used the specified reaction become impossible to make with the remaining 22 reactions.

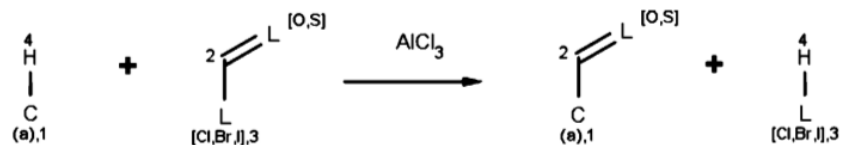
The RASA system is similar, in that the scores it assigns are computed by a function that was automatically tuned to fit a training dataset and based on output from a retrosynthetic engine (Huang et al., 2011). RASA uses a linear combination of scores measuring the variety of available synthetic routes, the difficulty of executing the reactions in each synthesis, and the difficulty of purifying the desired products. The weights on the subscores were fit, using linear regression, to the training set of 100 manually scored compounds. Like RS_{SVM} , the output of RASA depends on the quality of the retrosynthetic analysis. The RASA system uses 143 retrosynthetic transforms, each of which was manually annotated with electronic and steric penalties which are used to prune the retrosynthetic search. Additionally, RASA has a database of forbidden or unstable molecular subgraphs and it prunes search paths that require such precursors. Finally, RASA requires the complexity of molecules in its explorations to monotonically decrease, even though real syntheses are decidedly nonmonotonic as depicted in Figure 2.12.

2.7 Reaction libraries

The utility of a synthesis planner is limited by the fidelity of its reaction library to reality. The speed with which a synthesis is found is irrelevant if the proposed synthesis depends on unrealizable chemistry. Unfortunately, the effects on a molecule of a particular reaction condition are usually determined experimentally and are difficult to predict beforehand. It is often the case that a reaction predicted to enact some modification fails to do so in the lab (Nicolaou and Sorensen, 1996). Therefore all reaction libraries are necessarily approximations to reality.

Corey introduced the concept of *retrons*, or a molecular substructure generated by a particular reaction (Corey, 1967, using the earlier name *synthon*), which serve as a motivating principle for computer description and application of reactions. The presence of a retron in a molecule implies that the reaction can be used to construct the molecule from appropriate precursors. Corey's implementation in LHASA permitted the specification of additional property constraints on the product or reagents (Corey and Wipke, 1969). A modern example of this approach is presented in Figure 2.13. These labelled graph presence and absence requirements motivate Definition 10.

The majority of early computer synthesis planners, *e.g.*, OCSS/LHASA (Corey and Wipke, 1969), SYNCHEM (Agarwal et al., 1978, Krebsbach et al., 1998), SECS (Wipke et al., 1978), and SYNSUP-MB (Takahashi et al., 1990),



REACTIVITY: `charge(ratom(1), "aromaticsystem") < -0.2`

SELECTIVITY: `-energyE(ratom(1))`

TOLERANCE: `0.02`

EXCLUDE: `match(reactant(1), "[Cl,Br,I]C(=[O,S])C=C") ||`
`match(reactant(0), "[H][O,S]C=[O,S]") ||`
`match(reactant(0), "[P][H]") ||`
`(max(pkareactant(0), filter(reactant(0),`
`"match('[O,S;H1]')"), "acidic")) > 14.5) ||`
`(max(pkareactant(0), filter(reactant(0),`
`"match('[#7:1][H]', 1)"), "basic")) > 0)`

Figure 2.13: Reaction example from Pirok et al. (2006) depicting a Friedel-Crafts Acylation reaction. Additional properties specify the charge necessary at the active site for the reaction to complete. Problematic compounds are excluded with additional patterns.

used manually constructed reaction libraries. These libraries typically contained several hundred or low thousands of reactions. The extensive manual effort required to build them has been implicated as a significant contributory factor to the abandonment of the systems (Law et al., 2009). Of practical note to the PhD-candidate designer of a new synthesis planner, ChemAxon’s Reactor software is a recent free-for-academics manually-constructed library containing approximately 150 reactions (ChemAxon, 2011, Pirok et al., 2006).

Nakayama (1991) described a system to “crowdsource” the construction of reaction databases by reducing the effort of manual definition and curation of reaction patterns. SPEK represented transforms with a unique number, transform name, literature citation, matching structural pattern, reaction confidence, interfering groups, reaction conditions, and editor name for assigning blame when a reaction was described poorly. Despite this, no large repository of transformations is currently available for bulk extraction and processing.

Instead, a number of attempts have been made to extract transformations from synthesis databases. Commercial databases, such as SciFinder, Beilstein, and SPRESI, compile syntheses that have been published in the chemistry literature. Generalized transforms may be deduced by tracking the modifications enacted at each step of a synthesis. The hope is that the abundant training examples in the databases contain sufficient information from which to extract the retrons. The difficulty with this approach is that synthetic databases typically lack negative examples as syntheses that fail likely do not get published. Another challenge for automated transformation

extraction systems is that reactions act via different mechanisms may produce the same modifications in certain circumstances but not in others.

Most such systems are based on the work of Wilcox and Levinson (1986). Wilcox and Levinson define two graph-based generalizations for every synthesis step. The first, called *minimum reaction concept* of a reaction step or MXC(R), is the smallest connected subgraph of a reaction R which contains all the changed bonds in that reaction. When we define “changed bonds” to include bonds with different bond orders or the bonds internal to functional groups which had been added to the molecule, the MXC of a reaction captures the changes enacted by the reaction. However, it does not necessarily capture all of the activating groups near the reaction center that are necessary for the reaction to proceed. The second generalization, called the *complete reaction concept* or CXC(R), expands the MXC with the transitive closure of all neighboring bonds that are not carbon-carbon single bonds. While the MXC probably does not include all structures necessary for a reaction, the CXC probably includes (at least) all needed molecular structures.

Given a large corpus of syntheses, we can generate MXCs and CXCs for every step in every synthesis. The reactions are then grouped into clusters sharing identical MXCs. That is, the reactions are partitioned into groups, each of which enacts the same edit to a molecule. Unnecessary structures in the CXCs can then be pruned by taking the maximum common substructures between pairs of reactions. These substructures can then be refined further by iteratively intersecting them with other members of the original cluster. This process produces a continuum of progressively less constrained transforms; at some point the keying substructure becomes so broad that the reaction will probably no longer succeed.

Blurock (1990) avoids the problem of determining which level of generalization adequately describes a real reaction by building an interactive planner, RETROSYN, and requiring the chemist to select which reaction is appropriate from the hierarchy of reaction classes. The chemist marks where in the molecule functionality should be changed or the bond to be made or broken, and the system matches the surrounding substructure against the MXCs in its database. It then displays these options to the chemist. RETROSYN will also attempt to automatically determine the need for protecting groups by matching the reaction center against other places in the molecule. If too many alternative sites are matched, the system will search down the hierarchy to find more precise keying substructures that match fewer sites and propose modifications to the molecule that will distinguish the desired reaction location from side-reactions loci.

Satoh and Funatsu (1999) also created an interactive planner, AIPHOS, based on similar principles. AIPHOS defines an extensive list of structural fragments and stereochemical characteristics that it uses to define reaction sites. These fragments were, presumably, chosen because they proved useful in discriminating reaction types and are therefore a reasonable starting basis for pattern extraction. Unfortunately, any such finite list is necessarily incomplete and it seems somewhat arbitrary as the authors provide neither evidence to support the particular choice

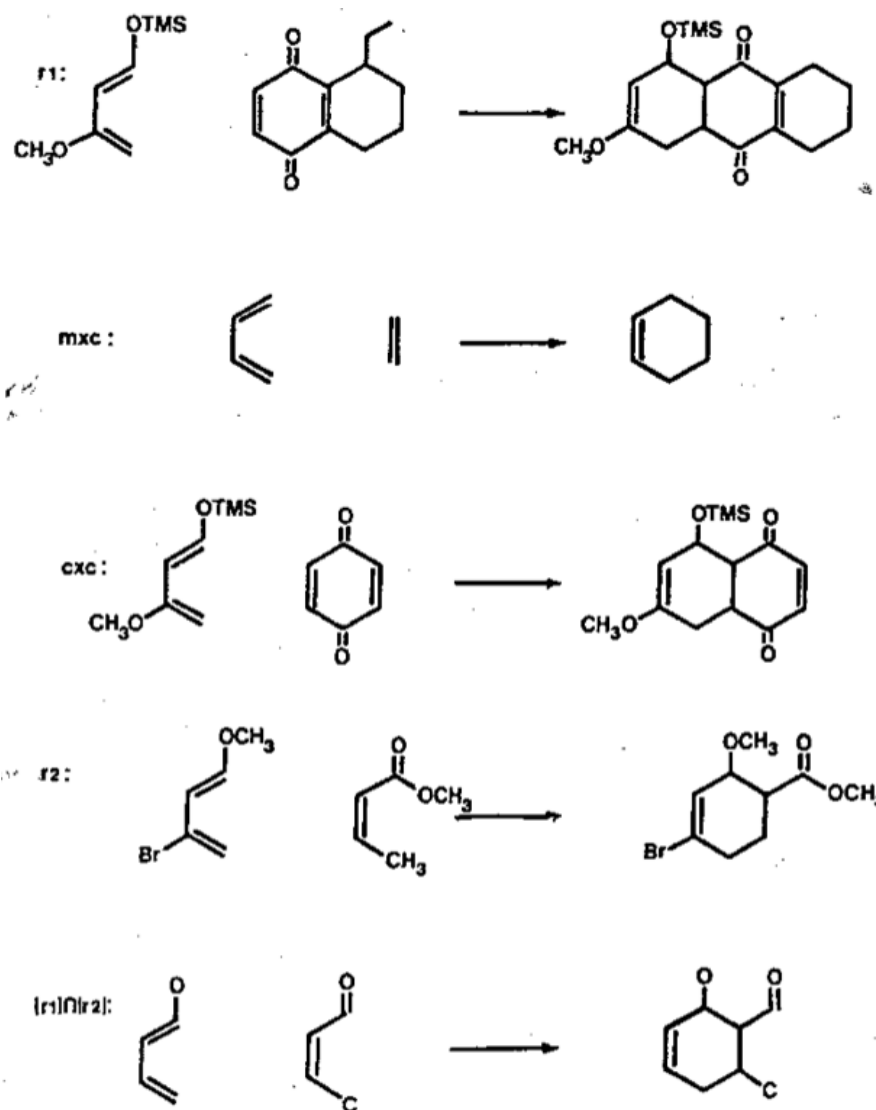


Figure 2.14: Diels-Alder example from Wilcox and Levinson (1986). Lines 1 and 4 show two reactions. Lines 2 and 3 are the MXC and CXC, respectively, for the first reaction. Line 5 shows the maximum common substructure from the two reactions (note the activating electron-withdrawing oxygen on the dienophile).

of fragments nor a discussion of how they were chosen. The AIPHOS system also defines the reaction neighborhood differently from Wilcox and Levinson's CXC. AIPHOS uses a simple bond radius definition, extending a neighbor from one to three bonds away from the changed bonds. This definition is simple and has been copied elsewhere, including the web-based searches of SciFinder, Reaxys, and Isentris (Eiblmaier et al., 2011, Wang et al., 2001).

Lawson and Kallies describe a technique, Random Access Black Box Indexing Term, for encoding multi-step reactions such that similar reactions would be clustered together (Lawson and Kallies, 1990). The authors correctly note that chemists, when searching reaction databases, seldom find exact examples of their desired reaction. Instead, chemists repurpose previously described reactions, which effect a desired structural transformation, to novel substrates when the substrates are judged to be sufficiently similar to the educts of the original reaction. To support this kind of workflow, a database query should return analogous reactions, as well as those that exactly match; this capability is supported by clustering related reactions. The authors describe two impediments to implementing such a database: first, the most ubiquitous reactions may be omitted from machine-readable reaction records precisely because they are so well known and, second, the judgements of molecular similarity and reaction applicability are subjective and, therefore, the results from any particular fixed set of rules are likely to meet with user objections. Unfortunately, the authors do not provide many details for how to address these critical issues; however, their assertions lend anecdotal support for the design choice of tuning a system to avoid false positives even at the expense of many false negatives.

The focus in Law et al. (2009) is a precise characterization of how the Wilcox and Levinson rules must be extended to handle a real-world dataset. These rule extensions were derived through an iterative process of creating a generalized reaction database from "Methods of Organic Synthesis" publications, having in-house chemists critique the generated database, and proposing new rules to address the deficiencies of the generated database. In effect, the available chemical transforms are derived from the literature but the rules governing transform extraction are manually defined.

Law et al. augmented their version of the minimum reaction concept, or *reaction cores*, to include all atoms which have changed attributes between reactant and product and the bonds connecting them. Attributes can include the number or element type of neighbors, charge, and radicals. These reaction core seeds are enlarged to produce complete reaction concepts as in Wilcox and Levinson (1986), which Law et al. term *extended cores*. An example is shown in Figure 2.15. Extended cores are produced by incorporating atoms into a reaction core until an atom with a carbon-carbon single bond is found. Additionally, the core is extended across adjacent double and triple bonds. If the core is connected by a single bond to a heteroaromatic atom, the entire aromatic system is added. Leaving group can be detected by finding connected unmapped atoms which are present in reactants but not the product. The extension includes the entire set of unmapped atoms and bonds.

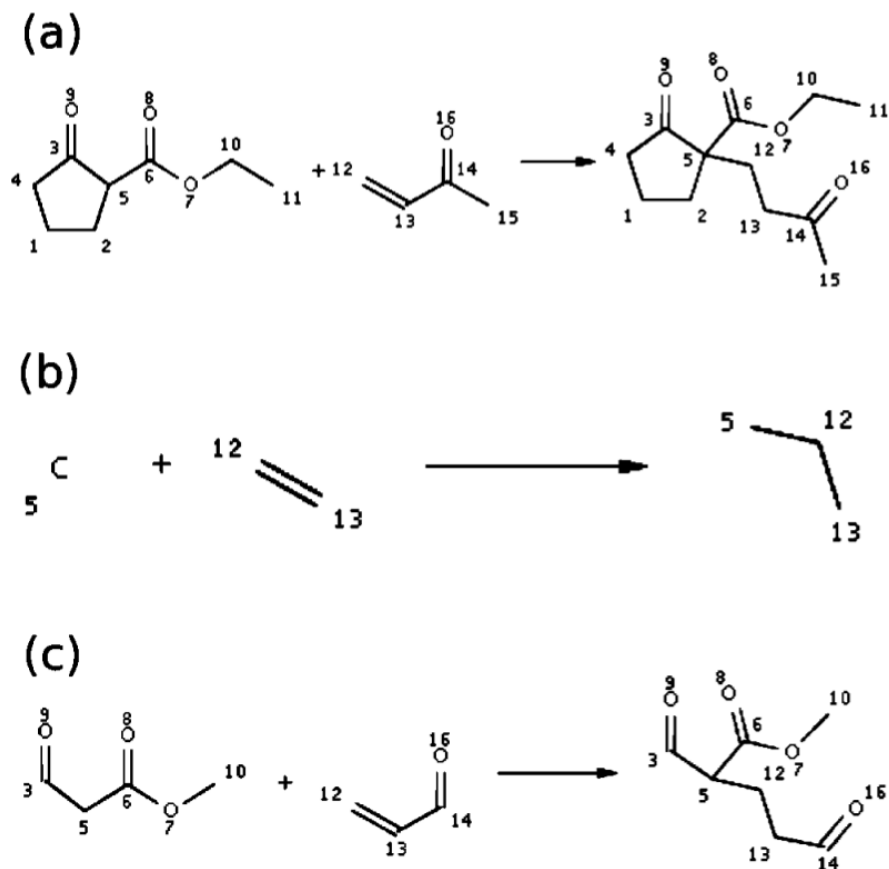


Figure 2.15: Example from Law et al. (2009). Figure (a) depicts a sample reaction, while (b) and (c) show an extracted core and extended core, respectively. Compare to the MXC and CXC from Figure 2.14. Law et al. note that the non-chemically-essential atom 2 is correctly not included in the extended core; in contrast, a bond radius approach such as (Sato and Funatsu, 1999) would have included it.

Given an extended reaction core, additional labels are computed, such as the electron withdrawing or donating nature of non-reacting adjacent functional groups. Reaction cores are clustered by labelled subgraph isomorphism and attributes such as permitted ring size or halogen type are grouped. The Methods of Organic Synthesis database containing 42,333 example reactions generated 4,028 unique rules. The full Beilstein reaction database containing 4.5 million example reactions generated 285,000 cluster groups. When these groups were pruned if they had fewer than five reaction examples, 92,781 unique rules remained. Finally, the extracted rules were augmented with 30 manually created common organic reactions, *e.g.*, esterification and amide formations, which are typically not reported in the literature (Lawson and Kallies, 1990). When reaction transforms are applied, subsequent checks are performed to ensure that implausible chemical structures are not created. These checks include charge balance, loss of aromaticity, and the creation of triple bonds in small rings.

2.8 The critical need for search guidance

The early organic synthesis planners set the fundamental design of heuristic state space search. In most planners, a target molecule is simplified by replacing matched retrons with the precursor substructures specified by the reaction reagents. These simplifying transformations may disconnect parts of the molecule to form simpler precursors. For example, rings may be disconnected to form linear molecules, and linear molecules may be disconnected to form two smaller precursors. These required precursors are, in turn, then recursively analyzed and disconnected. The search terminates when all required precursors are sufficiently simple or are found in starting material libraries. Some implementation details vary across previous planners. For example, the systems vary in how the reaction libraries are constructed and in how the next precursor to analyze is chosen. The retrons may be derived automatically or specified manually, and the search may be guided by algorithmic heuristics or user judgements. Still, the general approach to heuristic search was remarkably consistent across most previous attempts.

The critical confounding factors in computer aided organic synthesis planning are the accuracy of the chemical transformations and the combinatorial explosion of possible synthetic routes. It is possible to construct high fidelity transformation libraries through meticulous and painstaking effort (Corey and Wipke, 1969, Gelernter et al., 1977, Johnson et al., 1992, Lawson and Kallies, 1990, Ott and Noordik, 1997, Pirok et al., 2006, Takahashi et al., 1990, Wipke et al., 1978). Furthermore, when transformation libraries are constructed by hand, the user must trust the library author to precisely and accurately delineate the applicability of each transformation. Unfortunately, chemists must believe that a proposed reaction can be successfully executed on a given set of molecules, and chemists disagree about which reactions will succeed under different conditions. Chemists disagree with the opinions about chemical transformations embedded in reaction libraries even when these were constructed

by Nobel prize winners (Corey, 1991) and textbook authors (Corey et al., 1985). Furthermore, the derivation of reasonable syntheses based on rules derived statistically from the literature implies that such approaches can be sufficiently accurate to be empirically useful (Law et al., 2009).

The need for effective synthesis-graph exploration was noted early in the development of organic synthesis planners (Gelernter et al., 1977). However, even modern planners use search algorithms of limited sophistication. For example, Law et al. (2009) employ an exhaustive search to fixed depth. Over the last two decades, significant development has been achieved in both algorithms and heuristics applicable to organic synthesis planning. A number of search algorithms based on memory-efficient use of proof and disproof numbers have seen success in calculating game-theoretic values for game states (Nagai, 2002). While these games are much simpler than chemistry, the computer systems employing proof-number based algorithms far exceed human capabilities (Nagai, 2002, Schaeffer et al., 2007, Seo et al., 2001). Similarly, recent developments in molecular complexity measurements that incorporate starting material availability as well as difficulty of chemistry (Boda and Johnson, 2006, Boda et al., 2007, Ertl and Schuffenhauer, 2009) provide a qualitatively new kind of heuristic guidance for organic synthesis planners. The proper exploitation of these new techniques may lead to planners that are appropriate tools for contexts such as retrosynthetic analysis and starting material and chemical literature searching. Therefore, the focus of proximate research in automated organic synthesis planning should be a deeper integration of heuristic search techniques.

Chapter 3

A declarative description of chemistry

Before we can discuss our own design of computer systems for organic chemistry, we provide a declarative model suitable for computer representation. In contrast to previous axiomatization efforts (Masoumi and Soutchanski, 2012), we describe a representation that is convenient for translation to and from existing cheminformatics tools. Specifically, we choose a model of chemistry that is powerful enough to capture the way chemistry is described in practice, as previously discussed in Sections 2.3 and 2.7.

Chemists depict molecules and reactions as annotated graphs (see Figures 2.4-2.6). Our models are similar. We represent molecules as undirected graphs that are augmented with functions over their vertices and edges to convey information about different kinds of atoms and bonds, respectively. Reactions describe the changes in bonds that can be effected by the practicing chemist. Our representation can be interpreted as a generalization of classical state space search (Russell and Norvig, 1995, Chapters 3, 11). In the STRIPS language for state space search (Fikes and Nilsson, 1971), states and goals are represented by conjunctions of boolean variables, while operators specify a boolean formula that must be true for the operator to be applied (called preconditions) and a boolean formula (called effects) that describes the change to state variables when the operator is applied. The presence or absence of particular subgraphs in a molecule take the place of sets of boolean variables in state descriptions. Reactions replace operators, with reaction conditions corresponding to operator preconditions and the reaction product corresponding to a postcondition of the operator. We now make these intuitions precise.

3.1 Definitions

As molecules in nature are undirected graphs, we begin by defining atoms and bonds from nodes and edges.

Definition 2 (Atom). An *atom* is a member of a set of labelled nodes $A = \{a_0, a_1, \dots\}$.

Definition 3 (Bond). A *bond* is an unordered pair of atoms.

Real-world atoms and bonds carry additional information. For example, real-world atoms and bonds vary in their element type, atomic weight, electronegativity, bond order, or 3D configuration (Cahn et al., 1966). We define a set of annotation functions to capture these kinds of information.

Definition 4 (Atom Annotation). J is a set of functions $\{j : A \mapsto \mathbb{R}\}$, annotating additional information about atoms.

Definition 5 (Bond Annotation). K is a set of functions $\{k : A \times A \mapsto \mathbb{R}\}$, annotating additional information about bonds.

We now construct molecules out of atoms and bonds.

Definition 6 (Molecule). A molecule is a finite undirected graph, described as a tuple $M = \langle V, E \rangle$, where:

- $V \subseteq A$ is a finite set of atoms, and
- $E \subseteq (V \times V)$ is a finite set of bonds.

In addition to the properties covered by Definitions 4 and 5, atoms and bonds also have properties which depend on the molecule in which they are contained. For example, the electron density on each atom depends on the connectivity of the molecule and the type of element of each atom. The electron density molecular property would consider the entire molecule and annotate each atom with its specific electron density. Other properties, such as membership in a ring, might annotate only a subset of atoms.

Definition 7 (Molecule Annotation). L is a set of functions $\{l : \langle V, E \rangle \mapsto 2^{(A \cup (A \times A)) \times \mathbb{R}}\}$, annotating additional information about atoms and bonds based on their molecular neighborhoods.

We now construct a definition for molecular comparison (a kind of “annotated” subgraph isomorphism) from the standard definitions of *subgraph* and *isomorphism* (West, 2000). As a reminder, West writes

Definition 1.1.16: A *subgraph* of a graph G is a graph H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$ and the assignment of endpoints to edges in H is the same as in G . We then write $H \subseteq G$ and say that “ G contains H ”.

and

Definition 1.1.20: An *isomorphism* from a simple graph G to a simple graph H is a bijection $f : V(G) \mapsto V(H)$ such that $uv \in E(G)$ if and only if $f(u)f(v) \in E(H)$. We say “ G is *isomorphic* to H ”, written $G \cong H$, if there is an isomorphism from G to H .

For molecular operations, we want to extend the notion of equivalency over a graph’s nodes and edges to also respect to the properties of the atoms and bonds.

Definition 8 (Molecular Subgraph Isomorphism). Let $M = \langle V, E \rangle$ and $M' = \langle V', E' \rangle$ be molecules. We say that M' is isomorphic to a subgraph of M , written $M' \tilde{\subset} M$, if and only if

- there exists a subgraph $M_0 = \langle V_0, E_0 \rangle \subseteq M$, such that M' is isomorphic to M_0 , and
- under some such isomorphism f ,

$$- \forall j \in J, \forall v \in V', j(v) = j(f(v)) \text{ and}$$

$$- \forall k \in K, \forall \{v, u\} \in E', k(\{v, u\}) = k(\{f(v), f(u)\}) \text{ and}$$

$$- \forall l \in L,$$

$$[(\forall v \in V', \forall (x, r) \in l(M'), v = x \Leftrightarrow (f(v), r) \in l(M)) \text{ and}$$

$$(\forall \{v, u\} \in E', \forall (\{x, y\}, r) \in l(M'), (v = x \wedge u = y) \Leftrightarrow (\{f(v), f(u)\}, r) \in l(M))]$$

The annotations of M' are checked against M rather than M_0 because we want the subgraph to match the annotations of the original full molecule. For example, consider atoms that are in a ring in M ; if M_0 contains half of these, they may no longer be annotated as rings and could incorrectly match non-ring atoms in M' .

Definition 9 (Unmapped bonds). For convenience, we name the bonds which are not mapped under molecular subgraph isomorphism. Let $M = \langle V, E \rangle$ and $M' = \langle V', E' \rangle$ be molecules and $M' \tilde{\subset} M$, as above. Let $M_0 = \langle V_0, E_0 \rangle \subseteq M$ be a subgraph of M such that M' is isomorphic to M_0 under isomorphism f . Let $E_{unmapped} = E \setminus E_0$ and $V_{unmapped} = \left(\bigcup_{uv \in E_{unmapped}} u \right) \cup \left(\bigcup_{uv \in E_{unmapped}} v \right)$. Then we write “ $\overline{M' \tilde{\subset} M}$ ” to denote the molecule $\langle V_{unmapped}, E_{unmapped} \rangle$.

A reaction definition describes the conditions necessary to enact a particular set of bond changes in a molecule. Since the reactions are externally specified by chemists, we can constrain neither the necessary conditions nor the possible bond changes. Reactions are also described using graphs that specify reactive sites. These reactive sites are mapped to subgraphs found within the reagent molecules to which the reaction is applied. The reactive sites may be thought of as analogous to operator preconditions in a search problem (Russell and Norvig, 1995, Chapter 11): in the same way that operator preconditions must be true before the operator can be applied, reactive sites must be present in the reagents for the reaction to be applicable. Applying a reaction changes the bonds in the reagents to the bonds specified by the product subgraph. Examples are shown in Figures 2.13, 2.14 and 2.15 and discussed in Section 2.7.

Definition 10 (Reaction definition). A reaction is a tuple $\langle P, R, \phi \rangle$, such that:

- $P = \langle V_p, E_p \rangle$ is a molecule.
- $R = \{ \langle r_0^+, R_0^- \rangle, \langle r_1^+, R_1^- \rangle, \dots, \langle r_n^+, R_n^- \rangle \}$ is a finite set of pairs, where r_i^+ is a molecule and R_i^- is a set of molecules.
- $\phi : V_p \mapsto \bigcup_{\langle \langle V_{r^+}, E_{r^+} \rangle, R^- \rangle \in R} V_{r^+}$ is an injective function providing an atom-to-atom mapping from the atoms of P to the atoms of the r^+ molecules.

P describes what is created by the reaction. The r^+ 's in R represent the reactive sites in the reagents that react to form the product P . See Figure 2.5 for an example. The R^- 's in R represent interfering groups: molecular substructures which are forbidden from appearing in the reagents for the reactions to successfully proceed. See Figure 2.13 for an example. In the reaction application (*vide infra*), R^- 's are used to specify negative constraints on the molecules that can form product P successfully.

Given the definition of a reaction and a particular set of reagent molecules, we can ask whether the reagents will react in the manner described by the reaction and, if so, what product they would form. For example, in Figure 2.6, the salicylic acid contains a subgraph that matches the alcohol active site from the esterification reaction shown in Figure 2.5. At the same time, the complete molecule of acetic anhydride matches the anhydride active site. The esterification reaction then modifies the bonds in these active sites to form the ester substructure, producing aspirin. A reaction's conditions are satisfied by a set of reagents, if the reaction's reactive sites are isomorphic to subgraphs of the reagents, while none of the forbidden structures are isomorphic to a subgraph of that reagent. Examples of such interfering substructures may be seen in Figure 2.13.

Definition 11 (Reaction application). Given a set of molecules, S , and a reaction $O = \langle P, R, \phi \rangle$, we write $O(S) = M$ to denote that applying reaction O to molecules S can yield molecule M if and only if

- $P \tilde{c} M$, and
- $\forall \langle r^+, R^- \rangle \in R, \exists s \in S, \left[r^+ \tilde{c} s \wedge \left(\overline{r^+ \tilde{c} s} \right) \tilde{c} M \wedge \nexists f \in R^-, f \tilde{c} s \right]$.

We may say “ S react by O to form M ” or “ M is produced from S by O ”.

In other words, the structures formed by reaction O must be present in molecule M ; each required reactive site r^+ must be present in some molecule s in the set of starting materials S ; the parts of the starting material s which do not participate in the reaction must also appear in M ; and none of r^+ 's forbidden substructures can show up in its corresponding starting material s . Note that a single starting material can contain multiple reaction sites, as in intramolecular or polymerization reactions.

Observation 1 (Propositional encoding of reaction applicability is intractable). Testing whether a reaction can be applied to a set of starting materials requires a test of subgraph isomorphism, which is NP-complete (Garey

and Johnson, 1990). Worse, generating the set of reaction products requires enumerating all subgraphs of the starting materials that are isomorphic to the member of R ; this problem is #P-hard by reduction from the problem of counting perfect matchings in a bipartite graph (Valiant, 1979). These properties imply that the best-known encodings for organic chemistry into a propositional description language such as STRIPS (Fikes and Nilsson, 1971) require exponentially large operators. Of course, our work does not depend on any particular graph isomorphism algorithm and this observation does not preclude the possibility that domain-specific algorithms can be fast in practice on particular examples (Cao et al., 2008).

Given the definitions of molecules, reactions and reaction applications, we can formally state the problem of organic synthesis.

Definition 12 (Synthesis problem). A synthesis problem, Π , is a tuple $\langle G, \Delta, \Sigma \rangle$, where:

- G is a molecule.
- Δ is a set of reaction definitions.
- Σ is a finite set of molecules.

G specifies the desired goal product. Δ describes a library of the reactions available to the chemist to produce molecules. Σ provides a catalog of available starting materials.

Definition 13 (Synthesis). Let $\Pi = \langle G, \Delta, \Sigma \rangle$. A solution to synthesis problem Π is synthesis ψ , a finite set of pairs $\langle I, O \rangle$ providing a pairing of sets of molecules, I , to reaction definitions, O . ψ exists when:

- Every reaction definition used in the synthesis is part of the library of known reactions. Let $\Delta_\psi = \bigcup_{\langle I, O \rangle \in \psi} O$. Then $\Delta_\psi \subseteq \Delta$.
- The goal molecule, G , is provided as an available starting material or it is produced by a reaction, O_G , applied to O_G 's required precursor molecules, I_G . In other words, $(G \in \Sigma) \vee (\exists \langle I_G, O_G \rangle \in \psi, [O_G(I_G) = G \wedge (G \tilde{c} G' \wedge G' \tilde{c} G)])$.
- Every required precursor molecule is either available as a starting material or is provided a synthesis itself. Let $\pi_i = \langle i, \Delta, \Sigma \rangle$ be the synthesis problem for precursor i and ψ_i a synthesis for π_i . Then, we require $\forall i \in I_G, (i \in \Sigma) \vee (\exists \psi_i, \psi_i \subseteq \psi - \langle I_G, O_G \rangle)$.

These definitions present a somewhat idealized model of organic synthesis planning, but even this expressive description does not model chemistry completely. Molecules are physical structures and certain conformations of the graphs cannot be practically embedded in 3-dimensional space due to, for example, torsional strain. More detailed models could be derived from physical or quantum mechanical simulation. Our reaction definitions do

not describe the procedure to effect the molecular changes they denote. We assume the reaction library contains references into the chemical literature describing temperatures, solvents, catalysts, and so forth required to perform the reaction successfully. In general, even under this simplified model, the unbounded synthesis problem is undecidable by reduction from the word problem for semi-Thue systems (Post, 1947), so we now turn our attention to finding solutions for particular cases.

Chapter 4

Retrosynthetic search algorithms

As discussed in Chapter 3, organic syntheses can be modeled as branching paths in a discrete, fully-observable state space. We describe a model of organic chemistry that is amenable to traditional AI techniques from game tree search, regression, and automatic assembly sequencing. We demonstrate the applicability of AND/OR graph search by developing the first chemistry solver to use proof-number search. This is a kind of heuristic search that uses the shape of the search frontier to decide where to spend additional search time. Finally, we construct a benchmark suite of organic synthesis problems collected from undergraduate organic chemistry exams, and we analyze our solver's performance both on this suite and in recreating the synthetic plan for a multibillion dollar drug.

4.1 Introduction

Organic chemistry underlies medicine and materials science and provides modern conveniences in the forms of drugs, dyes, fragrances, and pesticides. A core task in organic chemistry is the planning of *organic syntheses*, the succession of chemical reactions that construct desired molecules from simple commercially-available starting materials. No medicine used in the developed world today would exist without solving these planning problems. Yet, despite 40 years of work by chemists on computer-aided organic synthesis Corey and Wipke (1969), Law et al. (2009), the chemical synthesis planning problem is largely unknown in the AI community. In this paper, we bridge that gap by modeling organic synthesis planning as discrete state-space search and show that it is amenable to heuristic search techniques.

This paper provides three contributions. First, we introduce the AI community to organic synthesis planning, a long-standing problem of medical and economic importance. Historically, the effort of constructing chemistry-aware software carried too much overhead for AI researchers; today, however, a number of toolkits that encapsu-

late chemical knowledge can be used as blackboxes. These include free open source software packages such as OpenBabel O'Boyle et al. (2011a) and RDKit Landrum (2006). Second, we describe the first application of proof number search to chemical synthesis planning. Third, we offer the first public synthetic planning benchmark. To encourage further development in this essential area, we freely distribute both our solver and benchmark. We conclude with a discussion of the need for specific new algorithmic developments to address the challenges of the organic synthesis planning problem.

Organic synthesis presents an interesting and challenging planning domain due to the complexity of the state description, the cost of checking whether a state satisfies the goal conditions, and the number of operators and their interaction. Here we provide the AI practitioner with a qualitative description of these factors. Due to lack of space, we do not present all of organic chemistry here. For a more comprehensive survey of computer-aided organic synthesis in particular and organic chemistry in general, we refer the reader to Todd (2005) and Clayden et al. (2000), respectively.

Figures 2.4, 2.5, and 2.6 show typical depictions of a molecule, a reaction, and a multistep synthesis. The “game board state” of the organic synthesis game is a molecule. Molecules are represented by graphs where vertices correspond to atoms (of elements such as carbon, oxygen, hydrogen, etc.) and edges correspond to bonds of different types (single, double, aromatic, etc.). The “game moves” in chemistry are reactions, which describe the change in bonds that can be enacted on molecules. These required activating and forbidden interfering substructures are similar to the positive and negative preconditions of planning operators. Despite eliding many factors of chemical reactions, such as temperature, solvents, catalysts, and side products, such transformation-based reaction descriptions capture the structural modifications that are achievable through a reaction, and are therefore widely-used by chemists Daylight (2008), Pirok et al. (2006).

It is important to note that the same reaction can be applied to many different molecules, provided that the molecules contain the appropriate reactive substructures. For example, the final step of Figure 2.6 is an application of the reaction described in Figure 2.5, and the alcohol is one small piece of salicylic acid. This applicability of reactions to many different molecules is similar to ungrounded operators from domain-independent planning.

Furthermore, one reaction may produce multiple possible products. Unlike deterministic games, where the same move applied to the same state produces a single outcome, a reaction can match several alternative reactive sites in a molecule. Consider again the reaction described in Figure 2.5. If it is applied to a molecule containing two alcohol groups, we could imagine 3 distinct product molecules, where the reaction has occurred at one alcohol, the other, or both.

Finally, one molecule may be produced from many precursors, even when using the same reaction. Every atom in the product originates in a reagent but the converse is not true because these reaction definitions are not assumed to be balanced or symmetric. While atoms are neither created nor destroyed in a chemical reaction,

chemists typically isolate a single product of interest, effectively ‘losing’ atoms in side-products. In Figure 2.5, the atoms labelled 6, 7, and 8 are lost into solution and are not incorporated into the final ester product. Given this reaction definition, there exist an infinite number of molecules that could produce the ester: any molecule that contains an anhydride substructure can hang alternative substituents off of the 7 atom, and would not be constrained by the specification of the product and the reaction.

4.2 A Proof Number Search-based Solver

First, we model the generation of organic syntheses as a two-player zero-sum game. This will enable us to use Proof Number Search to find syntheses.

The organic synthesis “game” can be described as follows: the first player picks a reaction that would synthesize the goal molecule, if the required reagents were available. Then, the second player picks one of that reaction’s required reagents, and demands that the first player choose a reaction that can synthesize it. The players continue to take turns until the molecule to be synthesized is found in the library of starting materials, yielding a win for the first player. Alternatively, if no reactions can generate the chosen molecule, the second player wins. The game is a forced win for the first player if and only if she can construct all required reagents. Once all of the required reagents have been constructed, she can apply the original reaction and produce the goal molecule. This recursive construction game provides an algorithm to generate syntheses.

Game proofs, which establish a game-theoretic value for game positions, closely match the structure we require in the production of organic synthesis. To demonstrate if a particular board configuration is a win for one player, a game tree proof provides, for every opponent move, a response that forces a win (although the responses can vary depending on the opponent’s move). Game tree proofs can be computed even for games with complicated search spaces, such as games with multiple distinct paths to given board positions or reversible moves. Proof number search and its memory-efficient variants Allis et al. (1994), Kishimoto (2010) have been used to generate game tree proofs for large real-world problems such as checkers Schaeffer et al. (2007).

Given a two-player zero-sum game-based formulation of organic synthesis planning, it is natural to apply AND/OR graph search. AND/OR graph search has been applied to the derivation of winning strategies in two-player games Allis et al. (1994), Nagai (1998, 2002), Schaeffer et al. (2007) as well as problems such as MDPs Bonet and Geffner (2006), assembly plans Homem de Mello and Sanderson (1990), and web service composition Liang and Su (2005). AND/OR graphs comprise two types of nodes that may be labeled as solved. An AND node is solved if all of its children are solved, while an OR node is solved if any of its children are solved. A solution graph therefore specifies (at least) one action for each non-goal node, which connect the initial state to some set of goal states.

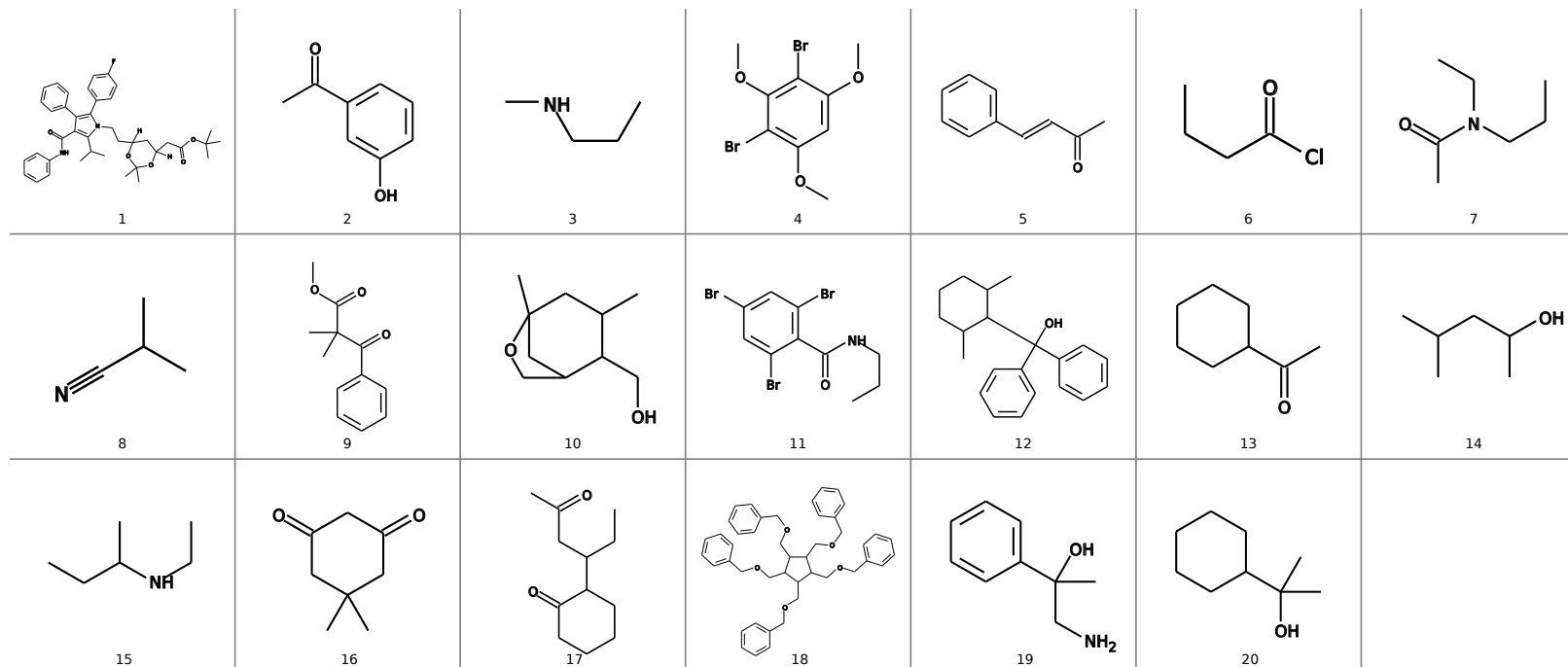


Figure 4.1: The benchmark target molecules. Images generated directly from the problem definition using OpenBabel O'Boyle et al. (2011a).

Proof number search, or *PNS* is a particular kind of AND/OR graph search and is given in Algorithm 3. *PNS* begins with an unevaluated root node; for organic synthesis, a root OR node is constructed for the goal molecule. Every reaction in the reaction library is checked to determine whether it could have produced the goal molecule. For each reaction that matches the goal, we generate possibly several sets of precursor molecules that could react to produce the goal. Each molecule in the set of precursors is mapped via a transposition table to an OR node, and then the set of precursor molecules is wrapped in an AND node. This AND node is connected as a child to the original goal molecule's OR node. In the case where a reaction only requires a single precursor, we perform a simple memory optimization and use the precursor's OR node as a child directly, omitting the usual wrapping AND node. The proof and disproof numbers of the root node are updated according to Definitions 2.1 and 2.2. For a detailed mapping of organic chemistry to AND/OR graph search, see Section 2.2.

Selection and expansion of leaves occurs as in normal *PNS* with several modifications. A leaf node is considered to be a terminal win if its molecule is found in the starting material library and it is considered to be a terminal loss if either no reactions would produce the molecule or if any required precursor set contains an ancestor molecule in the current synthesis path. ChemAxon's JChem 5.7.0 toolkit is used for all reaction applications and starting material matching ChemAxon (2011). Where possible, the precursor molecules are not grounded but are interpreted as minimal requirements over matching sets of molecules. Although the subgraph isomorphism necessary for reaction matching is intractable in the worst case, domain-specific algorithms can be fast in practice Cao et al. (2008). For example, Cao et al. demonstrate that tuning a molecular subgraph matching algorithm to, for example, match rare atoms such as oxygens before common atoms such as carbons, can yield significant improvements in runtime.

Usually, in these game proof algorithms, playing a move is considered to be free. This can lead to very large proof trees. Using the move-cost modification from dfpn^+ Nagai (2002), we define the cost of each reaction application to be 1, to help focus the search on shallow convergent syntheses. Additionally, this captures the fact that performing a reaction is costly in terms of chemist time.

The search space of organic chemistry forms a directed cyclical graph. In general there exist multiple alternative synthetic paths to reach a given molecule. Similarly, as with chess where board states can repeat, molecules can be transformed via a series of reactions back into themselves. For example, oxidations and reductions are a pair of reactions that are inverses of each other (increasing and decreasing the order of a bond). In an organic synthesis, a cyclical synthesis implies that a molecule is consumed in its own production. Such cyclic plans are not feasible. We add a check to terminate the investigation of a cyclical synthesis, as in games of first-player loss Kishimoto and Müller (2004).

Definitions 2.1 and 2.2 imply that parent AND/OR nodes contain a child with equal (dis)proof counts. However, this condition is only maintained if the search space is a tree. Since the chemical search space is a graph,

the search algorithm will modify the values of one node and update the values of only one of its parents. This can leave the other parents with (dis)proof numbers that are not equal to the (dis)proof numbers of any of their child nodes. Such a discrepancy in values between parents and children is called *inconsistency*. When the search encounters a node with (dis)proof values that are inconsistent with the values of its children, the selection or expansion process is skipped and the node's values are recomputed from its children's values. Often, this updating brings the node out of consistency with its own parents, so the process is repeated until every node along the current search path has values consistent with its children. Since this fix-up is performed only along the current search path through the search graph, this procedure may leave (or, indeed, cause new) inconsistencies in other pairs of parent and child nodes.

Problem	Instructor-provided solution			Proof-number search						Exhaustive search		
	operators	unique ops	depth	operators	unique ops	depth	expanded	generated	seconds	expanded	generated	seconds
1	5	5	4	5	5	4	1353	59522	10166	–	–	–
2	4	4	4	4	4	4	13	124	7	275	1897	39
3	5	5	4	5	5	4	22	145	8	176	993	26
4	4	4	4	4	4	4	25	702	15	1256	15968	224
5	3	3	2	3	3	2	11	769	31	23	1475	51
6	7	7	7	6	5	5	188	1877	31	474	4103	61
7	10	9	6	13	12	7	378	4301	70	24495	157198	2860
8	5	5	5	5	5	5	78	1810	96	474	7684	372
9	7	7	4	7	7	4	267	4614	94	2902	39256	639
10	3	3	3	3	3	3	168	3532	105	461	8985	247
11	12	12	9	12	12	9	787	15233	228	–	–	–
12	10	8	6	9	7	5	422	5932	258	9548	74865	2614
13	4	4	2	4	4	4	594	53781	3732	31	2773	121
14	4	4	3	4	4	3	406	64599	5138	149	4715	219
15	10	10	6	11	10	7	499	68335	5085	–	–	–
16	4	4	4	–	–	–	–	–	–	–	–	–
17	5	5	3	–	–	–	–	–	–	–	–	–
18	5	5	5	–	–	–	–	–	–	–	–	–
19	6	6	6	–	–	–	–	–	–	–	–	–
20	11	9	7	–	–	–	–	–	–	–	–	–

Table 4.1: Chemical benchmark. The instructor-provided solution describes the solutions given in the exam answer keys. Total runtime is given in seconds and includes initialization and output of solution. Dashes indicate that the problem did not complete in 6 hours.

4.3 Constructing a Public Chemistry Benchmark

Standardized benchmarks and competitions are important tools for direct comparison of a variety of algorithmic techniques Genesereth et al. (2005), Helmert et al. (2008). Unfortunately, no standard public benchmark exists for chemical synthesis planning. To permit principled comparison of such systems, we have created the first public chemistry benchmark. It is released to the community under a Creative Commons license and is available for download at www.cs.toronto.edu/~aheifets/ChemicalPlanning.

The benchmark contains 20 synthesis problems derived from undergraduate organic synthesis examinations. With the exception of the first problem, the benchmark problems appeared in exams at the Massachusetts Institute of Technology Course 5.13 “Organic Chemistry II” during 2001-2006. As such, the benchmark comprises problems captured from “the wild”, in that the questions were not designed for computers; rather, the tasks were designed to be challenging for humans and to test fundamental chemistry concepts. The original tests and instructor-provided answer keys are distributed via MIT OpenCourseWare MIT (2003).

We derived problem 1 from primary literature, rather than from course exams. The target of problem 1 is atorvastatin, marketed in North America as Lipitor[®]. It is an interesting molecule, because atorvastatin was the best-selling drug in history with yearly revenues in excess of USD\$12 billion before its patent expiration in 2011. The benchmark problem uses the starting materials and reactions reported by the inventors Brower et al. (1992), Roth (2002).

Figure 4.1 shows the target molecules from our benchmark, which are analogous to goal states in automated planning problems. Each synthesis problem also specifies a set of starting materials that are permitted (but not required) to be used in the construction of the target. We manually encoded these target and starting materials in a standard machine-readable format Weininger (1988). Each problem also contains a sample solution, although multiple correct answers are generally possible. From each solution, we created a corresponding set of machine-readable reaction files. We focused on encoding the transformations enacted by the reactions and did not attempt to document reaction conditions, selectivities, or stereochemical effects. In principle, these data may be represented and future editions of the benchmark could encode these reaction properties.

Table 4.1 describes the characteristics of the instructor-provided solutions for each problem. The ‘operators’ column lists the total reaction applications when the solution is flattened into a tree. The ‘unique operators’ column lists the number of reaction applications in the solution. When there are fewer unique operator applications than the number of operators, it means that the solution is a directed acyclic graph rather than a tree and a subproblem was reused in different parts of the synthesis. This provides an indication of the occurrence of shared synthetic intermediates that are used multiple times in the problem answer. ‘Depth’ describes the maximum number of reactions separating the target molecule from any starting material. An entirely linear problem (such as

problems #2 and #8) will have the same number of operators, unique operators, and depth, whereas a branching solution (*e.g.*, problem #11) will have a depth that is smaller than the number of operators.

In summary, the benchmark contains 20 problems, a library of 62 unique starting materials, and 50 reaction definitions. The benchmark is designed to be stand-alone; it is expressed in standard formats and requires no external reaction or starting material libraries. The problems cover a range of difficulty, each requiring between 3 and 12 unique steps. The canonical instructor answers described solutions that include directed acyclic graphs, trees, and simple sequences. The problems were developed to be difficult for undergraduate chemistry students and, we believe, are therefore a useful testing ground for automated chemistry solvers.

4.4 Results and Discussion

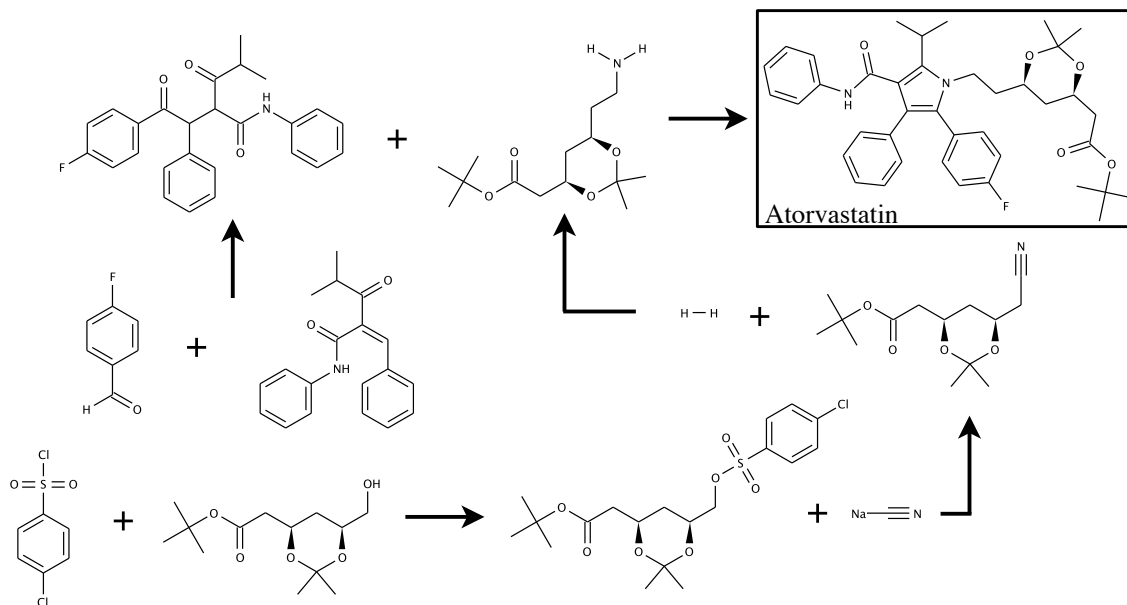


Figure 4.2: Computer-generated Atorvastatin synthesis matching the synthesis reported in Brower et al. (1992) and Roth (2002).

Benchmark tests were performed on an IBM CL1350 cluster with 1,344 cores over 168 Infiniband-connected HS21-XM BladeServers and a DCS9550 storage system. The cluster runs the CentOS operating system, version 5.1, uses OpenJDK Runtime Environment (IcedTea6 1.7.5), and manages coarse-grained parallelism with the Portable Batch System. ChemAxon's JChem 5.7.1 was used for the molecular manipulations ChemAxon (2011). Each test received a maximum of 6 hours CPU time and 8GB of RAM.

The benchmark provides a range of problem difficulties. Table 4.1 shows that 5 problems completed in less than one minute, 6 problems required between 1 minute and 5 minutes, 4 problems required between 1 hour and 3 hours, and 5 problems did not terminate within 6 hours. While problems with deep solutions generally require

more time and search effort than shallow problems, the nature of the target molecule strongly impacts the search space. When many reactions can generate the substructures in a molecule, many more paths must be explored. This is illustrated in problems #2 and #4; both are linear syntheses with four steps but problem #4 generated more than 5 times as many nodes as problem #2. Furthermore, applying reactions takes a significant amount of time, which varies depending on the molecules under consideration.

Today, organic synthesis problems are solved manually; the standard commercial solver is the human. In this context, we compare against the gold standard, because the benchmark consists of problems taken from university exams intended to challenge human students. Direct comparison of our solver against other chemical planners, while desirable, is impossible. The majority of previous synthesis systems were research projects in the 1970-80s and no longer available for comparison, since even the underlying operating systems do not necessarily run on modern hardware Agarwal et al. (1978), Corey and Wipke (1969). The algorithms consisted of exhaustive search moderated by *ad hoc* collections of forward pruning rules.

We did compare our algorithm to a reimplement of the algorithm used by the only existing commercial software Law et al. (2009), which is exhaustive search to a user-specified depth. We chose to reimplement the algorithm because a license costs several tens of thousands of dollars. Our implementation of *IDA** takes a user-specified bound so as to adhere as closely as possible to the published Law et al. algorithm. We specify the depth bound to be the minimum of the depths of the benchmark's provided solution and the solution returned by our proof-number searcher. Furthermore, we omitted node-construction to minimize algorithmic overhead and derive the best performance from the exhaustive algorithm.

On most problems, proof number search outperforms exhaustive search in terms of nodes expanded, nodes generated, and total time spent. For example, on problem #7, exhaustive search expanded over 64 times as many nodes as *PNS*. In three cases, proof-number search solved problems on which exhaustive search was terminated after 6 hours. In problems #13 and #14, *PNS* performed worse than exhaustive search. In these problems, the solver appears to be led away from the cheapest solution by the early discovery of a molecular precursor. As can be seen from Definitions 1 and 2, the proof number search does not optimize for shortest solutions, but rather for minimum remaining proof burden. If a particular reagent is solved early in the tree, it is considered 'proved' and the cost for using it anywhere else in the search is set to 0. Therefore, by proving early on one molecule from a complicated precursor set, the search may be driven into unfruitful areas of the search space. Integration of heuristics such as those discussed in Section 2.6, including Bertz (1981), Hendrickson et al. (1987), and Ertl and Schuffenhauer (2009), would help avoid such waste and speed the discovery of syntheses. Such integration would widen the margin between *PNS* and exhaustive search as, by definition, exhaustive search can not leverage heuristics to focus the search.

We observed that our solver suffers from the problems of over- and under-counting proof and disproof num-

bers. These problems are observed whenever the search space is a graph rather than a tree Kishimoto and Müller (2004), Seo et al. (2001) When cycles occur or when intermediate nodes are shared along different parts of the solution path, naive solutions will sum the (dis)proof numbers of leaves multiple times along overlapping paths. This causes the search to waste time in incorrect portions of the search space and may lead to infinite looping Kishimoto (2005), Kishimoto and Müller (2008). Solutions that avoid both high computational overhead and inaccurate proof and disproof counts are an area of active research Kishimoto (2005, 2010), Kishimoto and Müller (2003).

4.5 Chapter Conclusion and Future Work

This paper introduces an economically- and medically-critical problem to the AI community. It is clear that our solver does not yet exceed the abilities of a skilled human undergraduate chemistry student. However, it can already solve a number of challenging synthesis problems, including the the expert-derived synthesis of atorvastatin, the best selling drug in history.

Our prototype shows the feasibility of applying planning techniques to the chemical synthesis problem. We described the first application of proof-number search to chemical synthesis planning and the first benchmark for comparing synthesis solvers. We hope chemical synthesis planning will be a fruitful domain for AI research. To that end, we are distributing our code and benchmark, which are available at www.cs.toronto.edu/~aheifets/ChemicalPlanning.

A move from theory and prototype to a realistic system requires several extensions. Search efficiency could be improved by initializing the AND/OR (dis)proof numbers with a stronger heuristic. The reaction definitions should be expanded to encompass a broader set of reactions and the starting materials library should be scaled to a full commercial chemical database. These additional reactions and starting materials would permit the discovery of novel syntheses without blocking the recognition of the synthesis we report in Figure 4.2. We will investigate the derivation of stronger heuristics in Chapter 6 but, to do so we will need to construct a large set of synthesis examples from which to derive our domain-specific heuristics.

Chapter 5

Compilation of synthesis and chemical data

The patent literature is a rich catalog of biologically-relevant chemicals; many public and commercial molecular databases contain the structures disclosed in patent claims. However, patents are an equally rich source of metadata about bioactive molecules, including mechanism of action, disease class, homologous experimental series, structural alternatives, or the synthetic pathways used to produce molecules of interest. Unfortunately, this metadata is discarded when chemical structures are deposited separately in databases.

SCRIPDB is a chemical structure database designed to make this metadata accessible. SCRIPDB provides the full original patent text, reactions, and relationships described within any individual patent, in addition to the molecular files common to structural databases. We discuss how such information is valuable in medical text mining, chemical image analysis, reaction extraction, and *in silico* pharmaceutical lead optimization. SCRIPDB may be searched by exact chemical structure, substructure, or molecular similarity and the results may be restricted to patents describing synthetic routes. SCRIPDB is available at <http://dcv.uhnres.utoronto.ca/SCRIPDB/search/>

5.1 SCRIPDB

U.S. patent information is in the public domain and describes innovations in the medical, biological, chemical, and agricultural fields. Such relevant and accessible material is ideal for scientific analysis and, indeed, databases such as PubChem (Wang et al., 2009) and ChEBI (de Matos et al., 2010, Degtyarenko et al., 2008) contain chemical structures disclosed by patents. While such databases are highly useful, structural databases are insufficient for a number of scientific investigations.

The extraction of component structures from a patent discards information about chemical relationships. These relationships can be explicitly labelled, such as a molecule's role as reagent or product in a chemical synthesis. Relationships may also be implicitly embedded in the context of the complete patent. For example, molecules that co-occur in drug patent claims are likely to have similar biological behavior. Splitting a patent into component structures needlessly removes information about chemical relationships.

These exemplar relationships have been valuable in statistical analyses for automated reaction extraction (Law et al., 2009, Wilcox and Levinson, 1986) and bioisostere discovery (Southall and Ajay, 2006). Reaction extraction characterizes the molecular transformations that occur within a set of syntheses. Bioisostere discovery catalogs molecular substituents that participate in similar biological interactions. Such analyses require access to large datasets, which are often unavailable, proprietary, or expensive.

In addition to the relationships among a patent's molecular structures, a patent's data files are directly useful. One use of patent files is the creation of data sets for optical structure recognition. The task in optical structure recognition is to parse a chemical image and recover the depicted molecular structure. The training and tests sets therefore require correctly matched pairs of images and molecular structure files. As a final example, a patent's written contents can serve as a target for text analytics. Patent descriptions and claims constitute a large corpus of biomedical text, coarsely annotated by, *e.g.*, patent classification and drug-disease pairs.

To address these gaps in available metadata, we have created SCRIPDB. While providing intuitive searching of the patent literature, we have been careful not to eliminate underlying information. Users of the database may download the full text of the patent as well as molecular structure files and images. Additional summary files were generated and augment, rather than replace, the original data. Thus, SCRIPDB permits researchers to effectively access the full information contained in the patent literature.

The United States Patent and Trademark Office (USPTO) provides full patent text, drawings, and, since 2001, chemical structures in complex work units. This data became available as a free bulk download, hosted on Google servers, in June 2010. The raw data files comprise every granted patent, numbering several thousand per week, and totaling over 10 terabytes of data. However, most patents are not relevant to the biological, chemical, or medical domains.

Since 2001, patented chemical structures can be described using standard molecular file formats. The USPTO makes disclosed molecules available as either MDL Molfiles (MOL) or ChemDraw binary CDX files. We used the presence of these chemical structure files to identify patents of potential interest. Figure 5.1 shows the amount of data in SCRIPDB, as measured by the quantity of CDX files (Figure 5.1a), individual structures (Figure 5.1a), and patents (Figure 5.1b). These numbers are consistent with previous analyses of patent data (Li et al., 2010).

A particular structure will often be described with both a CDX and Molfile. However, the information contained in such parallel files is not always identical. For example, a CDX file can label a molecule's role in a

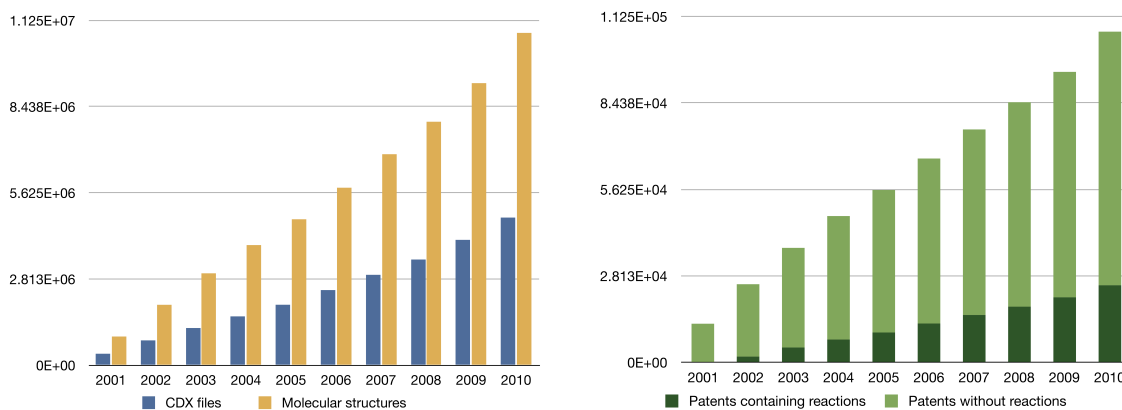


Figure 5.1: Cumulative SCRIPDB content. Although SCRIPDB includes patents from 2011, we show data through 2010, the last complete year. (a) Left panel shows the number of ChemDraw CDX structure files, and the structures described therein, available in SCRIPDB for various years. SCRIPDB contains 4,814,913 CDX files from 2001 through 2010, comprising 10,840,646 molecules. Duplicate molecules were filtered from each patent but not across patents, as described in the text. (b) Right panel shows the number of patents and details the subset containing reactions. For 2001 through 2010, SCRIPDB contains 107,560 patents, of which 25,048 contain synthetic reactions.

reaction as reagent or product, whereas a Molfile cannot. We provide both CDX and Molfile data formats, as well as original TIFF and generated SVG images. For additional convenience, we collated the individual structure files into one structure-data file (SDF) per patent. During collation, the generated SDF files were filtered to remove duplicate molecules. The filtering ensures that duplicated structures are removed from individual patents' SDF files but not across patents, as shared molecules may be used to uncover legitimate relationships among the patents.

Patents often describe sets of molecules rather than, or in addition to, specific structures. Markush structure notation is commonly used to succinctly describe large (or infinite) molecular classes by choosing constituent molecular fragments from alternative substituents, positions, frequencies, or homologies. The enumeration and comparison of Markush classes are significant challenges for cheminformatics systems (Barnard and Wright, 2009). For example,

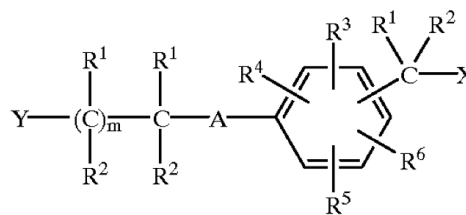


Figure 5.2: Example Markush structure from US Patent 6,268,504 (Raveendranath et al., 1999) defining a chemical class via substituent, positional, and frequency variations.

Markush structures typically depict variable substituent placement as bonds that cross rings, as in Figure 5.2. These bonds frequently appear in Molfiles as separate propane molecules laid over the core scaffold. SCRIPDB provides basic Markush handling by extracting the molecular core and canonicalizing variable substituents, which permits Markush structures to be found via substructure searches. The original Markush structure is then retrievable from the original patent's structure or image files.

Raw patent data was downloaded from Google's bulk download of USPTO granted patents with embed-

ded image data (<http://www.google.com/googlebooks/uspto-patents-redbook.html>). Filtering, collation, and deduplication were performed on an IBM CL1350 cluster with 1,344 cores over 168 Infiniband-connected HS21-XM BladeServers and a DCS9550 storage system. The cluster runs the CentOS operating system, version 5.1, and manages coarse-grained parallelism with the Portable Batch System. SDF files were generated and duplicate structures were removed using OpenBabel, SVN revision 4487 (Guha et al., 2006). OpenBabel was also used to generate SVG files and compute canonical SMILES strings (<http://www.daylight.com/smiles/>) for display of retrieved search results.

The web interface to SCRIpDB was implemented in Python 2.6 using the Django web application framework, version 1.1 (Forcier et al., 2008). Chemistry-specific search functionality, such as substructure searches or structural similarity using the Tanimoto coefficient (Tanimoto, 1958) of OpenBabel FP2 linear structural fingerprints, is provided via integration with Pybel (O'Boyle et al., 2008). Search structures may be specified via SMARTS queries, uploaded Molfiles, or via the interactive ChemWriter molecular editor (<http://metamolecular.com/chemwriter/>). ChemWriter is implemented in pure Javascript and permits SCRIpDB to be accessed from any major browser for desktop or iPad without the installation of external plugins.

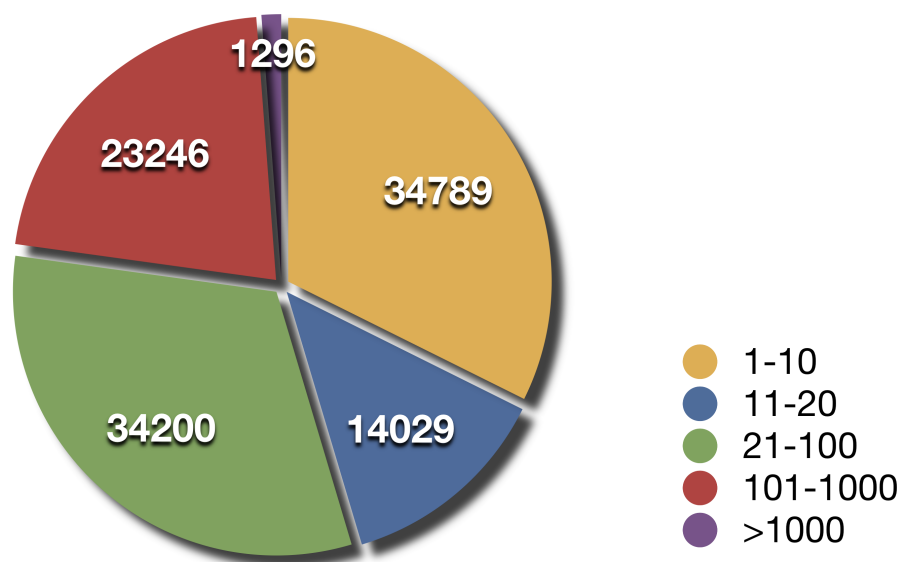


Figure 5.3: Structures per patent in SCRIpDB. While 34,789 patents contain ten or fewer structures, two-thirds of patents contain more than ten and 1,296 patents contain more than a thousand structures.

New patents are granted each week, providing a steady stream of additional data for SCRIpDB. Here, we report results to the end of 2010, which is the last complete year of patent data. At the end of 2010, SCRIpDB contained 107,560 patents, including 4,814,913 nonredundant CDX structure files. Many structure files describe multiple molecules which, after removal of duplicate molecules within a patent, yield a total of 10,840,646 molecules. Not only does this constitute a significant amount of total data (Li et al., 2010) but the rate of structure disclo-

sure appears to be growing. Both 2009 and 2010 had record numbers of molecules disclosed, at 1,259,097 and 1,639,522 structures, respectively. SCRIPDB data statistics are summarized in Figure 5.1.

However, focusing solely on the chemical structures ignores valuable chemical information. For many applications, molecular relationships need to be analyzed at various levels of granularity; for example, within a single CDX file, a particular patent, a group of patents that refer to a specific disease, or within the entirety of the database. As seen in Figure 5.1, patents typically contain multiple structural files which, in turn, contain multiple structures. For example, US Patent 6,884,815 contains 8,187 structure files (Thurkauf et al., 2003).

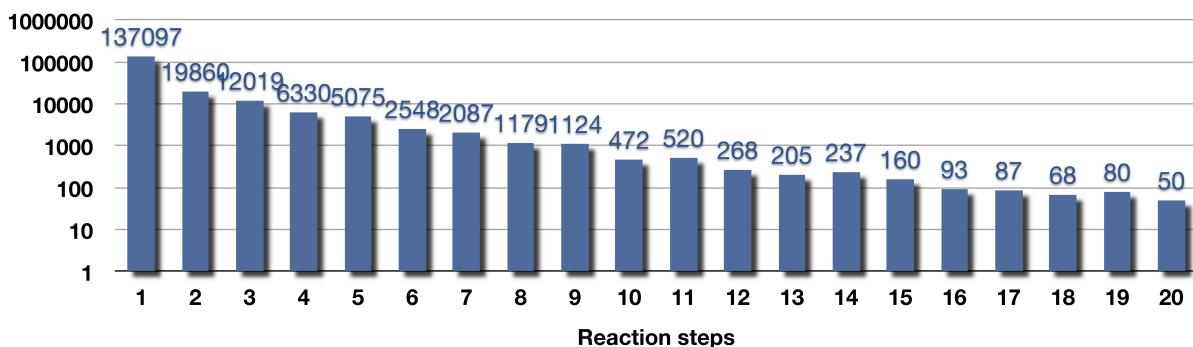


Figure 5.4: Number of CDX data files with that contain various numbers of reaction arrows.

Figure 5.3 shows the distribution of molecules per patent. The largest group contains patents that describe relatively few molecules (specifically, 10 or fewer). However, most patents catalog larger structural series. Two-thirds of patents contain more than 10 structures, over half of patents contain more than 20 structures, and almost 1,300 patents describe more than 1,000 structures each. A manual examination of a sample of patents shows that these structures often describe molecular analogues produced in the context of a medicinal chemistry optimization program.

An important relationship described in CDX files is chemical synthesis. Figure 5.1b details the subset of patents which contain reactions. These are the patents that contain a CDX file with a ReactionStep object (<http://www.cambridgesoft.com/services/documentation/sdk/chemdraw/cdx/ReactionStep.htm>). Of the 107,560 patents from 2001 to 2010, SCRIPDB contains 25,048 patents that describe syntheses.

Fundamental molecular roles, such as reagent and product, are lost when a synthesis is split into separate structures. These relationships may be difficult to rederive, especially in multistep syntheses. Figure 5.4, which shows the number of reaction steps in SCRIPDB's syntheses, demonstrates that synthetic pathways requiring multiple reaction steps occur frequently in the patent literature. The 25,048 synthesis-containing patents describe a total of 341,764 individual reaction steps. While the most common are single-step syntheses, 52,462 syntheses (27.6%) have at least two reaction steps.

JURISICA LAB
IBM Life Sciences Discovery Center

We found 77 relevant molecules. Download their patents. Download all patents.

Page 1 of 1. Showing 77 molecules.

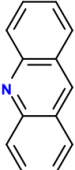
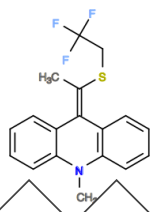
	<ul style="list-style-type: none"> ▶ Read US Patent 07842941 ▶ Download US Patent 07842941 ▶ SMILES: c1ccc2c(c1)nc1c(c2)cccc1 ▶ Download file: .CDX .MOL .TIFF .SVG
	<ul style="list-style-type: none"> ▶ Read US Patent 06270695 ▶ Download US Patent 06270695 ▶ SMILES: C/C(=c/1/c2cccc2n(c2c1cccc2)C)/SCC(F)(F)F ▶ Download file: .CDX .MOL .TIFF .SVG

Figure 5.5: Sample of search results for molecules containing an acridine substructure.

5.2 Discussion

In addition to their direct value in intellectual property licensing (Chen et al., 2009) and competitive business analysis (Hattori et al., 2008), patents can serve as a useful resource for a variety of academic research. We examine the datasets used in previous research and show that the data available in SCRIPDB is sufficient, quantitatively and qualitatively, to provide value for future investigations. Additionally, the information in patents is complementary to data available in the conventional academic literature (Thangaraj, 2007), suggesting additional insight may be derived from integration with existing datasets. Specifically, we survey patent data as a tool for the development of chemical image parsing, biological text mining, reaction extraction, and bioisostere discovery.

5.2.1 Patents as a source of chemical images

The problem of optical structure recognition is to automatically extract molecular structure information from images. A necessary resource in the training and evaluation of such systems is a set of molecular images for which the true molecular structure is already known. Ideally, the images are representative of chemical images as commonly used in practice.

Validation sets comprising 6,185 images (Filippov and Nicklaus, 2009) and 454 images (Valko and Johnson, 2009) were recently reported in the literature. By comparison, SCRIPDB provides millions of CDX structure and TIFF image files. This comprises a validation set several orders of magnitude larger than previously reported,

even after eliminating possibly-confounding complex synthetic schemes. Furthermore, SCRIPDB contains SVG images produced by OpenBabel in addition to the original patent's TIFF images. This redundancy permits testing structure recognition algorithms for robustness to the idiosyncrasies of alternative image generation tools.

5.2.2 Patents as biomedical literature

Statistical analysis of the biomedical literature requires large quantities of freely accessible documents. Much of the biological text mining research has therefore used PubMed abstracts of journal articles while more recent full-text analyses have focused on Open Access journals (Verspoor et al., 2009). A large corpus of 162,259 full-text journal articles in HTML were used in the TREC Genomics track (Hersh and Voorhees, 2009). The 107,560 patents in SCRIPDB constitute a document collection of comparable size.

Additionally, patents are interestingly complex. Patents are inherently semistructured and are partially annotated by their embedded chemical structures, gene sequences (Li et al., 2010), mathematical equations, and data tables. Patents also contain citations to related papers (Griffin et al., 2011) and patents, permitting co-citation analysis. Patent relationships may be derived and analyzed based on shared terms in patent text, shared molecules, patent assignee (Southall and Ajay, 2006) or ontological categorization (<http://www.uspto.gov/web/offices/opc/documents/classescombined.pdf>).

5.2.3 Patents as a reaction database

Libraries of chemical transforms are used for combinatorial library design (Pirok et al., 2006), static synthetic feasibility (Podolyan et al., 2010), and full retrosynthetic analysis (Law et al., 2009). Although such libraries can be constructed by hand (Corey and Wipke, 1969), automated extraction is an effective and labor-saving alternative (Law et al., 2009, Satoh and Funatsu, 1999, Wang et al., 2001, Wilcox and Levinson, 1986). Such systems determine molecular substructures that are modified in the same manner across many different syntheses. These modifications can then be treated as putative reactions, predicting that if the same molecular substructure is found in a new molecule, it can be changed in the same way. Automated transform extraction therefore requires a large corpus of example syntheses within which to find consistent molecular changes.

In this capacity, SCRIPDB with its 341,764 reactions compares favorably to the 42,333 reactions in the Methods in Organic Synthesis database recently used for reaction extraction (Law et al., 2009) or the 30,530 CCR reactions used to characterize functional group reactivity (Tanaka et al., 2010). While smaller than commercial databases containing millions of reactions, SCRIPDB has the virtue of being freely accessible.

5.2.4 Patents as a bioisostere catalog

After finding an initial lead molecule, medicinal chemists will typically create and test a large series of analogous compounds, seeking to increase binding affinity; improve absorption, distribution, metabolism, excretion, and toxicity profiles; or avoid the intellectual property restrictions of competitors' patents. One systematic approach for exploring chemical space is to substitute bioisosteres, which are molecular fragments that have similar shape and chemical properties. For example, hydrogen and fluorine have similar Van der Waals radii and the same valence. Exchanging a hydrogen with a fluorine permits the medicinal chemist to maintain the same molecular shape while optimizing the molecule's charge distribution.

There is strong interest both in techniques to determine bioisosteres (Langdon et al., 2010, Sheridan, 2002) and in idea-generation tools that propose alternative molecules *en masse* by modifying a lead molecule *in silico* using the same approach as a medicinal chemist (Stewart et al., 2006). Southall and Ajay investigated bioisosteric replacements in kinase patents by analyzing 116,550 compounds (Southall and Ajay, 2006). The maximum common substructure was computed for pairs of compounds and the remainder of the molecules were identified as exchangeable chemical replacements.

Southall and Ajay were interested in the research strategies of drug companies, so only compared compounds found in patents assigned to different companies. In principle, a similar analysis could be performed for the compound series within a single patent. Each patent defines a set of bioisosteric replacements which were determined to be reasonable, interesting, and synthetically feasible by a medicinal chemist. Figure 5.3 demonstrates that SCRIPDB contains sufficient patents with large chemical series to extract sensible chemical replacements.

5.3 Chapter Conclusion and Future Work

The impetus of intellectual property protection creates a deluge of patents that carries enormous quantities of chemical and biological information. While patented molecules are accessible via chemical databases, the extraction of component structures from a patent needlessly removes information about chemical relationships. SCRIPDB is designed to make such metadata broadly accessible. We examined the information used in medical text mining, chemical image parsing, reaction extraction, and the development of computational tools for lead optimization. We demonstrated that the quantity and quality of SCRIPDB's data compares favorably with existing commercial and free datasets. In many cases, it is complementary to existing data.

In the future, we plan to reduce the manual intervention necessary for the incorporation of new patents. In addition, we wish to pursue integration both with other value-adding databases such as ChEMBL (Warr, 2009) and CDIP (<http://ophid.utoronto.ca/cdip/>) and with additional countries' patent offices. Currently SCRIPDB incor-

porates only patents from the United States, because the USPTO provides distinct structure files. Robust optical recognition of chemical structures would permit future integration with patent offices that provide molecules as images, such as the European Patent Office, the Japanese Patent Office, World Intellectual Property Organization, the UK Intellectual Property Office and the Canadian Intellectual Property Office.

SCRIPDB is available at <http://dcv.uhnres.utoronto.ca/SCRIPDB/search/>

Chapter 6

Domain-specific heuristics for synthetic feasibility

6.1 Introduction

Chemists can imagine lots and lots of different molecules. Modern computer tools, such as *de novo* design or lead elaboration software (Langdon et al., 2010, Leach et al., 2006, Meanwell, 2011, Patani and LaVoie, 1996, Segall et al., 2011, Sheridan, 2002, Stewart et al., 2006, Wagener and Lommerse, 2006), help automate the process of brainstorming useful molecules. In some sense, they are too successful: that is, such tools can quickly suggest millions of molecular candidates, which is far too many for a human chemist to carefully analyze.

Consider Abbott's DrugGuru software as a case study (Stewart et al., 2006). Stewart et al. culled 186 molecular transformation rules from the medicinal chemistry literature. These describe pairs of "swappable" molecular substructures, such as replacing a hydrogen with a fluorine or a 5-carbon ring with a 6-carbon ring, that experienced medicinal chemists would consider as similar. Given an initial molecule, the software applies these rules to generate a list of similar-yet-novel molecules, which may be better tolerated by the organism or by patent-granting authorities. However, even this small set of carefully chosen rules can generate many molecules. For example, three applications of the 186 replacement rules described in (Stewart et al., 2006) would yield roughly $\binom{186}{3} = 1,055,240$ molecules¹. Worse, when the software was evaluated by medicinal chemists, most of the suggested molecules were considered too difficult to synthesize and it was found that

"[...] incorporation of synthetic chemistry knowledge [...] was the single most requested improve-

¹While all rules would not be applicable to a given molecule, some rules (such as moving heteroatoms around a ring) would generate many alternative molecules.

ment.” (Stewart et al., 2006)

In such a setting, there is a critical need to quickly eliminate from consideration molecules which are too difficult to synthesize. Truly impossible molecules can be eliminated by checking for the presence of molecular subgraphs that have been shown to be highly unstable (Nalin de Silva and Goodman, 2005). However, such filters leave large numbers of plausible yet unattractive molecules. Sadly, the ubiquitous economic constraints of time, money, resources, and available personnel compel us to explore the space of molecules in the most efficient manner possible. In addition to *e.g.* estimates of a molecule’s effectiveness at interacting with a particular disease target as in Appendix B, ease of production is an important criteria for ranking of proposed molecules. That is, even in the case of a set of molecules that are constructible in principle, we would like to know which molecules may be made quickly and efficiently for further testing.

The *synthetic feasibility* of a molecule (sometimes called synthetic accessibility, synthetic complexity, or synthetic tractability) measures the ease of its production. Many factors impact the time and resources needed for a molecule’s construction, including the number of chemical reactions that need to be performed to make the molecule, the level of skill required to perform the reactions successfully, the yield of the needed reactions, the cost of required reagents and catalysts, the difficulty of cleanly separating the desired molecule from other by-products, and even the irritation of the chemist at working with smelly or explosive compounds. Unsurprisingly, it is difficult to summarize all of these factors simultaneously. Indeed, several of these factors are subjective (*e.g.*, difficulty, inconvenience) or will vary between chemists or organizations (*e.g.*, reaction yield, reagent cost). To control for such factors, existing synthetic feasibility heuristics are tuned to recapitulate scores assigned to molecules by (groups of) human experts. However, human judgement is expensive to gather and rife with cognitive biases that limit the interpretation and applicability of synthetic feasibility estimators that are based on human opinions.

We take a fundamentally different approach. First, throughout this work, we approximate the synthetic feasibility of a molecule, which is difficult to quantify, with the number of synthetic steps needed to make that molecule. The time and effort needed to make a molecule scale with the number of chemical reactions that must be performed to produce it, and the number of synthetic steps in a given synthesis is both objective and quantifiable. Additionally, predicting the number of steps needed to synthesize a molecule can serve as a useful guidance in automated retrosynthetic analysis (Cook et al., 2012, Corey and Wipke, 1969, Heifets and Jurisica, 2012a), particularly as heuristic initialization of proof numbers in solvers such as the one described in Chapter 4.

We begin by constructing a database of organic syntheses described in US patents, as described in Chapter 5. For each molecule in each synthesis, we label the molecule with the number of synthetic steps needed to synthesize it. Through automated annotation of large databases, we are able to assemble a synthetic feasibility dataset that is an order of magnitude larger than previous efforts. Our data comprises 43,976 labeled molecules, compared to

the previous largest data set of 3,980 labeled molecules described in Takaoka et al. (2003).

These labels provide exact values for the number of steps required to synthesize any molecule present in the data set. Of course, we are most interested in evaluating the synthetic feasibility of molecules which have never been synthesized. Therefore, we need a means to generalize from the labeled molecules in our data set and to predict synthetic feasibility of unlabeled molecules. Using techniques from machine learning and computational linguistics, we construct a statistical model to predict synthetic feasibility and demonstrate that it exhibits accuracy and consistency comparable to groups of human experts.

6.1.1 Humans, eh?

As discussed in Section 2.6, many techniques have been proposed for synthetic feasibility estimation (Barone and Chanon, 2001, Bertz, 1981, 1983, Boda and Johnson, 2006, Boda et al., 2007, Ertl and Schuffenhauer, 2009, Huang et al., 2011, Randić and Plavić, 2002). These typically relate synthetic feasibility to the frequency of particular subgraphs in the evaluated molecule. Unfortunately, the evaluation of heuristic predictions is not straightforward. The underlying “gold standard” for synthetic feasibility has been based on human opinion, and human opinion is notoriously difficult to use.

In fields as diverse as medicinal chemistry and movie ratings, human opinions have been shown to depend on the context (*e.g.*, which other options are presented simultaneously) (Lajiness et al., 2004) and are known to vary over time (Koren, 2010). Even when people are presented with the same data at the same time, their opinions vary significantly. In a study where molecules were being selected for further investigation by medicinal chemists and could be rejected for any reason (such as difficulty of synthesis, unacceptable complexity, interference in screening experiments, or other unsuitability for further development), Lajiness et al. found that the expected agreement between chemists’ selections was only approximately 24%.

Worse, humans are not consistent *with themselves*. In the same study, a set of compound information had been duplicated and, therefore, evaluated multiple times by the medicinal chemists. The authors report that chemists selected the same compounds only 50% of the time. Similar results were reported in a study by Kutchukian et al. (2012), where medicinal chemists were asked to select, from several large sets of compounds, subsets of lead molecules which they would be willing to develop further. A subset of 227 compounds was present in each large set, providing the capability to assess selection consistency. Kutchukian et al. (2012) report that the similarity between chemists’ own selections of the 227 compounds ranged from 0.37 to 0.82, with an average of 0.52. While not particularly consistent, it is substantially better than inter-chemist agreement, where the similarity ranged from 0.05 to 0.52, with an average similarity of 0.28.

The results of studies on the consistency of human judgment in synthetic feasibility assessment vary. The

largest study to date showed correlation coefficients between the scores assigned by five chemists to 3,980 compounds range from 0.40 to 0.56 (Takaoka et al., 2003). On a set of 40 molecules, Ertl and Schuffenhauer (2009) report that the r^2 among 9 chemists ranged from 0.450 to 0.892, with an average of 0.718. On a set of 100 molecules, Boda et al. (2007) report that the scores given by 5 chemists range between 0.73 and 0.84. Factors that could influence consistency of scores include the diversity of kinds of molecules and the diversity of chemist expertise. (It is interesting that the largest data set showed the least consistency in scores.)

The variation in human opinions has been traditionally dealt with by averaging the scores given by a set of people. However, it is unclear that the average opinion is more correct than that of a single chemist: when the molecule requires specific techniques from *e.g.*, fluorine or carbohydrate chemistry, the opinion of a chemist with relevant experience should be preferred to the average of an arbitrary group.

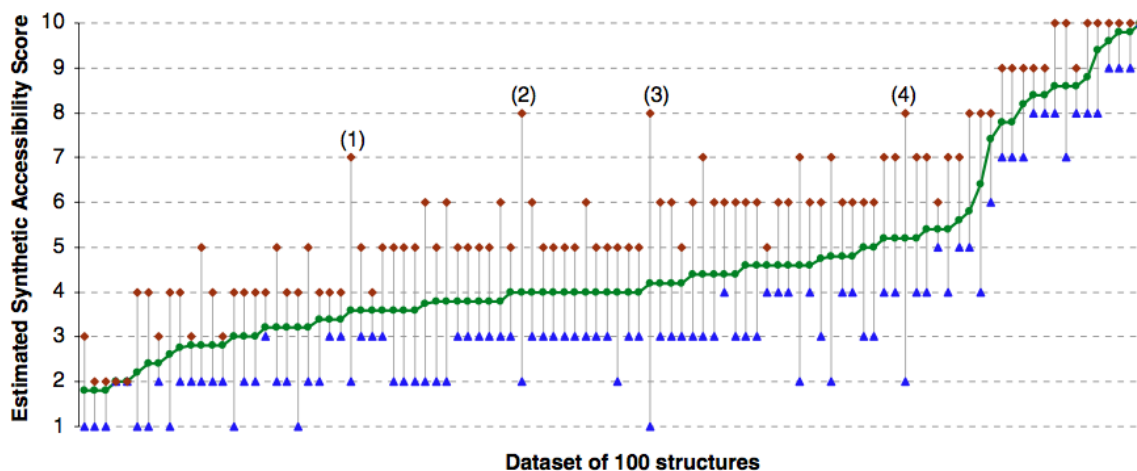


Figure 6.1: Chart from Boda et al. (2007) depicting minimum, maximum, and average synthetic accessibility scores by five medicinal chemists. Structures are sorted by average score. Molecules of particularly wide score disagreement are labeled. (This is Figure 2.10, duplicated here for reader convenience.)

Additionally, as can be seen in both Figure 6.1 and Figure 6.2, chemist agreement varies with the molecule's average score. Chemists agree that the simplest molecules are possible to make; chemists also agree that the most complex molecules are undesirably difficult. But, except for these absolute easiest and hardest molecules, we find a 2 to 3 fold range in manual synthetic feasibility assessment. That is, some people will give 3 where another gives 6. Additionally, molecules have large overlaps in score range. This is particularly problematic because it implies that a group of scores cannot dependably differentiate molecule complexity.

In practice, the large-scale extraction of opinions from human experts is costly and time-consuming. Data sets size and diversity is limited by economics and expediency. Both contribute to concerns about generalizability. Data sets sizes have been too small to permit the creation of meaningful held-out validation sets. In many cases, synthetic feasibility estimators were trained and tested on the same data points (Barone and Chanon, 2001, Bertz, 1981, 1983, Randić and Plavić, 2002). Diversity is a subtler issue but, when the test set molecules are very similar

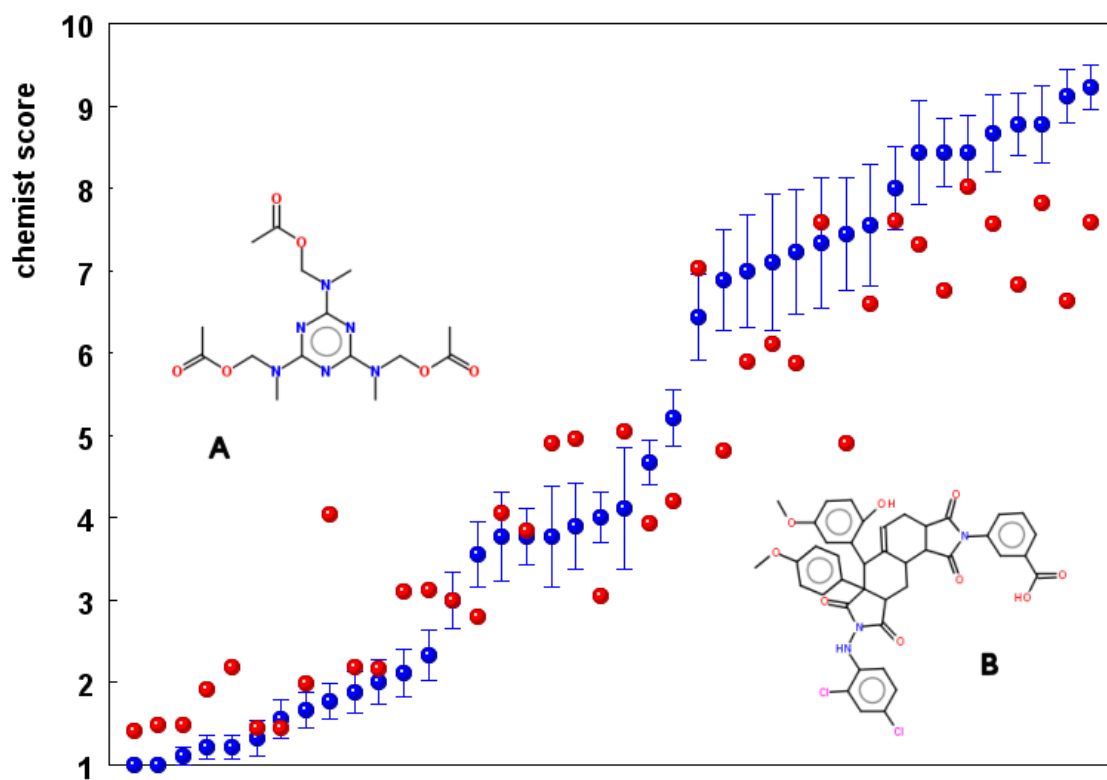


Figure 6.2: Chart from Ertl and Schuffenhauer (2009) depicting the average of chemist scores for 40 test molecules, in blue. The error bars denote the standard error of the mean of scores by 9 chemists. Therefore, to translate to standard deviation the bars would need to be tripled in size. Structures are sorted by average score. Molecules of particularly wide score disagreement between the average chemist score, in blue, and their software's scores, in red, are labeled.

to the molecules in the training data, a practitioner may be justifiably concerned about the generalizability of validation results.

6.1.2 Objective measures of synthetic feasibility

To address the problems of limited size, limited diversity, and limited consistency, let us decline to use manually-labelled synthetic datasets. Instead, we begin with the postulate that, to first-order approximation, the synthetic feasibility should correlate with the number of steps required to synthesize a molecule. Given an organic synthesis, the number of steps to reach each molecule ascribes a label to that molecule. If we can extract syntheses from chemical databases, such as those described in Chapter 5, then we can label each molecule within a synthesis with the number of reactions required to built it. These labels are empirically and objectively derived, the number of steps in a synthesis are not a matter of opinion, and a single synthesis yields many labels. Given a large enough dataset of syntheses, we can efficiently and economically create large independent training and test sets.

Unlike existing synthetic feasibility data where two chemists may have differing opinions on molecular difficulty, the number of steps in a synthesis is consistent and objective. Furthermore, human-labelled rankings of molecules - typically on a scale of 1 to 10 - do not contain intrinsic meaning. A molecule with a score of 6 is unlikely to be exactly twice as difficult as a molecule scored as a 3. In contrast, a molecule that requires 6 steps *does* require exactly twice as many steps as a molecule requiring 3. Scores that are physically meaningful allow the users of the synthetic feasibility estimations to interpret the predictions. The number of synthetic steps has a precise, real, and relevant interpretation, and serves as an meaningful approximation for synthetic difficulty.

Admittedly, a count of the number of steps does not capture all aspects of synthetic feasibility. In focusing on consistent universal properties of the molecular synthesis, this approach ignores reaction difficulty. For example, some reactions are toxic, explosive, expensive, odorous, low yielding, long running, or finicky. Unfortunately, no general synthetic feasibility algorithm would be capable of completely capturing this information. Different chemists have different experience and aptitude; a chemist with experience in carbohydrate chemistry and a chemist experienced in working with fluorine would likely assign differing difficulties to each other's chemistry. Reaction yields are known to vary widely across groups (Wernerova and Hudlicky, 2010). Similarly, the cost of a catalyst for an academic lab will be different than the cost for Pfizer's purchasing department. Therefore, any synthetic feasibility algorithm must be parameterized for the specific organization that wants to account for such factors. The current described approach is extensible to track cost, yield, or to assign differing difficulties for each reaction, where such information is available. It is possible that such tuning data can be extracted automatically from electronic lab notebooks (Christ et al., 2012).

6.2 Materials and Methods

Figure 6.3 depicts the steps we require to make predictions on synthetic feasibility. The first five steps concern extracting reactions from the raw real-world data, while the second five process it into a form suitable for use in synthetic feasibility prediction.

6.2.1 Data collection

Patents have recently been identified as a useful source of synthetic information, as 23% of chemical patents include syntheses (Heifets and Jurisica, 2012b, Jessop et al., 2011). An example is shown in Figure 6.4. Additionally, these syntheses are described in the machine-readable CambridgeSoft ChemDraw (CDX) format, in United States patents with “Complex Work Units”. Importantly, the United States explicitly places its patents in the public domain. This free and open access permits the independent validation of scientific results that is integral to the concept of Open Data (O’Boyle et al., 2011b).

The core difficulty to the data-driven approach presented here is the collection of the labelled data. SCRIPDB contains ChemDraw files with the capability of representing syntheses but this metadata is nearly always missing or inaccurate. For example, about half of the CDX files that contain arrows do not list ReactionStep information, without which it is impossible to ascribe reagents and products to a reaction. However, the TIF files are unambiguous because the drawings are the preferred medium of information exchange for chemists. Therefore, we visually interpret the layout of the CDX files directly. This is not quite a machine vision task because the CDX files directly describe molecular structures, positions, and the locations of synthetic arrow heads and tails, as depicted in Figure 6.5. We read this information by parsing the CDX file rather than by analyzing the patent images, thereby avoiding the need to solve the challenging problem of chemical image to structure translation (Filippov and Nicklaus, 2009). Specifically, we use the NextMove CDX File Parser (NextMove Software, 2010) to translate binary CDX files into an machine-parsable and human-readable text format. We parse the bounding boxes from the text files and annotate the molecules in the patents with their locations. In this way, we avoid low level machine vision problems, such as edge detection, and build a simpler and more robust system.

Our goal is the assignment of relationships and roles to the molecules based on their geometric positions. Because we are dealing with real-world data, the examples can be quite convoluted (as seen in Figure 6.6) and a challenge to label molecules as products or reagents. We perform this labeling by solving a system of constraints, governing how far a product or reagent can be from the terminus of an arrow to be considered participating in the reaction, where a ‘+’ can be legally placed to denote a charge versus the combination of molecules, and what role is played by molecules over or under arrows.

The process begins by reading molecular fragments, free text, and reaction arrows. Bounding boxes are

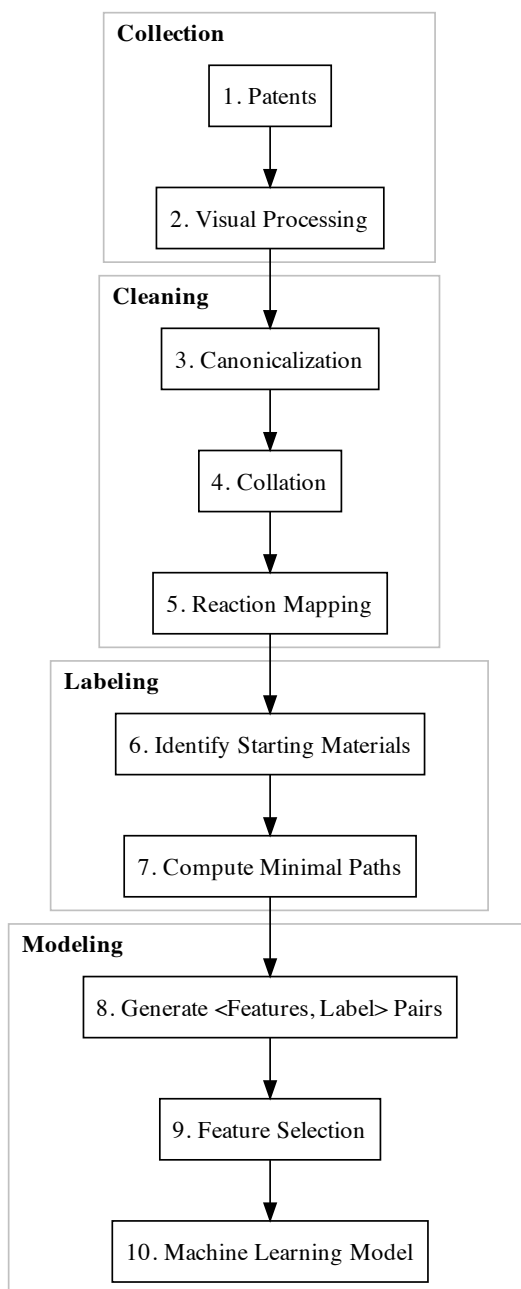


Figure 6.3: Data flow schematic depicting the processing steps in each of the collection, cleaning, labeling, and modeling stages.

Scheme II:
 Synthesis of N-(2-Benzoyl-Phenyl)-Benzenesulfonamides and
 N-(2-Benzoyl-Phenyl)-Heteroarylsulfonamides.

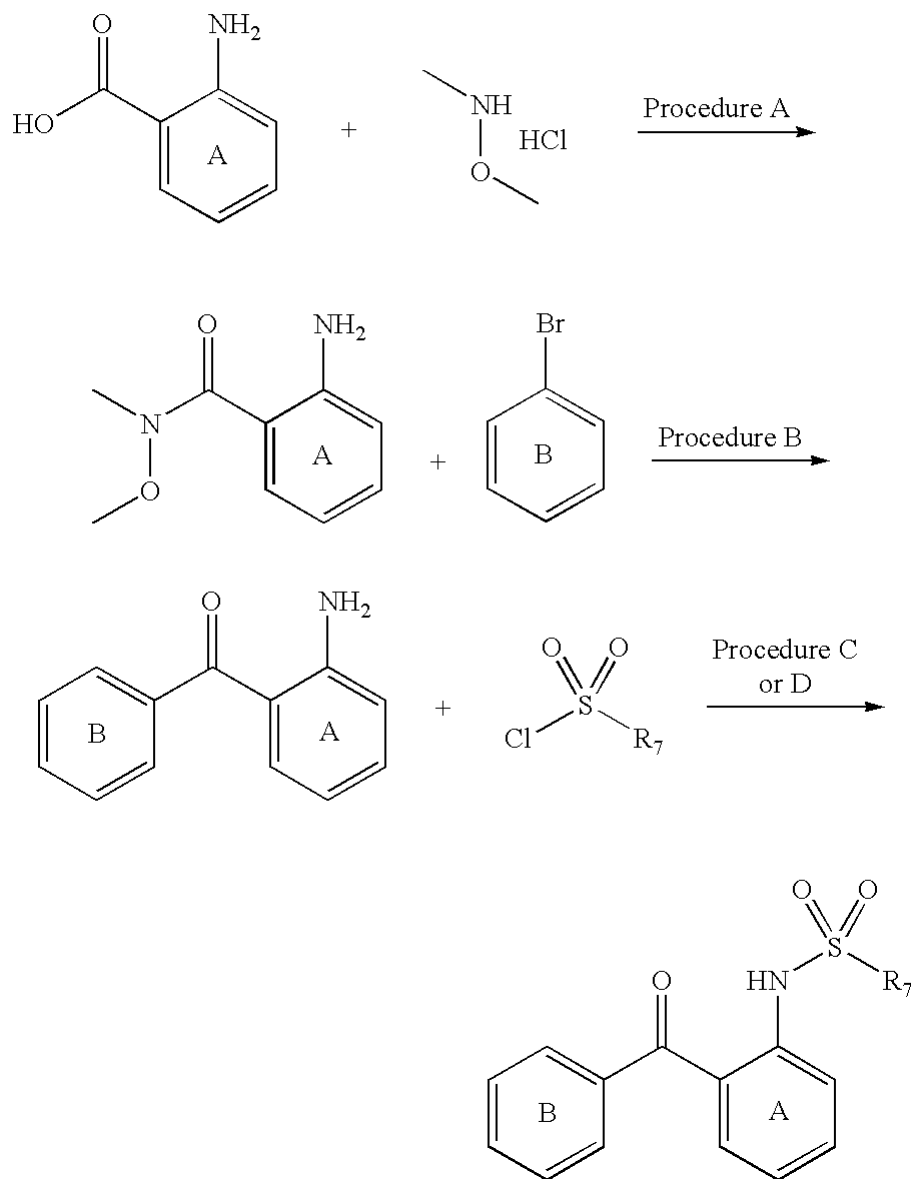


Figure 6.4: Example synthesis from US Patent 7,238,717 (Fleming et al., 2007). “Procedure A (Synthesis of Weinreb Amides)” is described in the patent as: “A substituted anthranilic acid (24 mmol) was dissolved in acetonitrile (200 mL), 1.05 equivalents of N,O dimethylhydroxylamine hydrochloride, 1.05 equivalents of EDC, 0.05 equivalents of dimethylaminopyridine and 1.0 equivalent of triethylamine were added and the reaction was stirred at room temperature overnight. The acetonitrile was removed by rotary evaporation and the residue was partitioned between ethyl acetate and water. The organic layer was washed with brine then concentrated to a residue. The residue was chromatographed on silica gel (ethyl acetate as eluent) to give the product. Typical yields are 70-90%”.

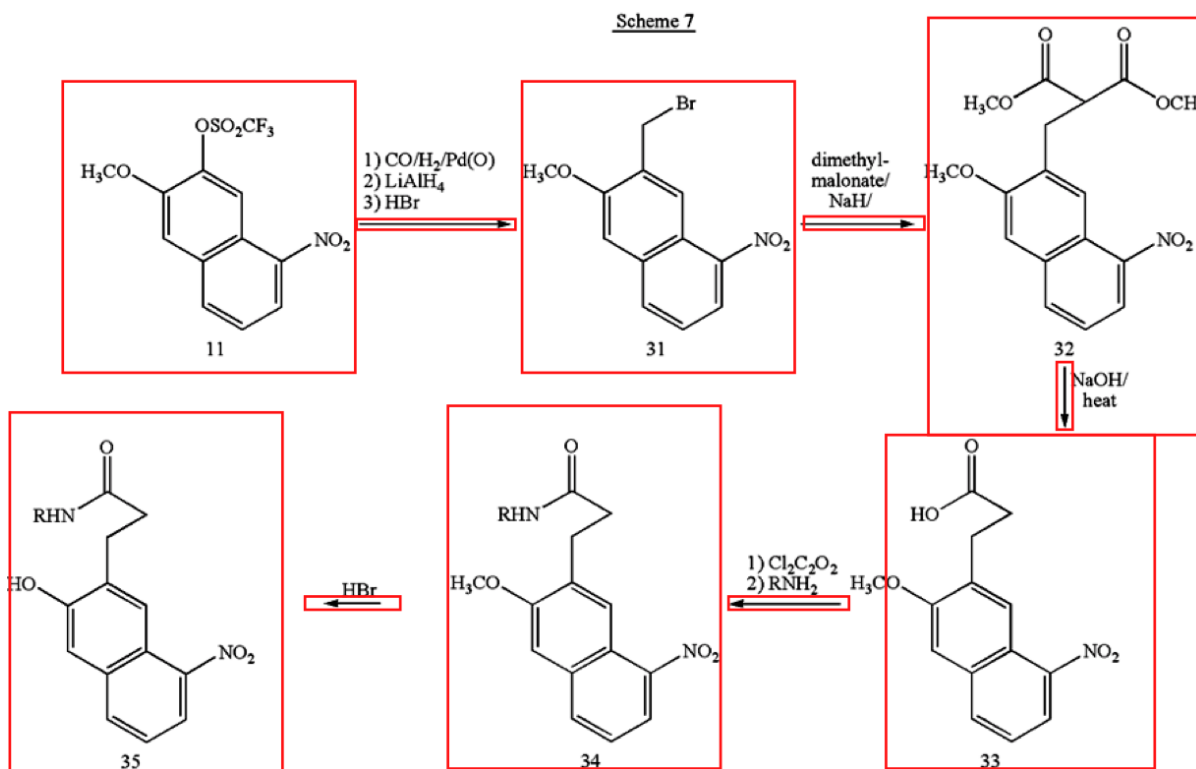
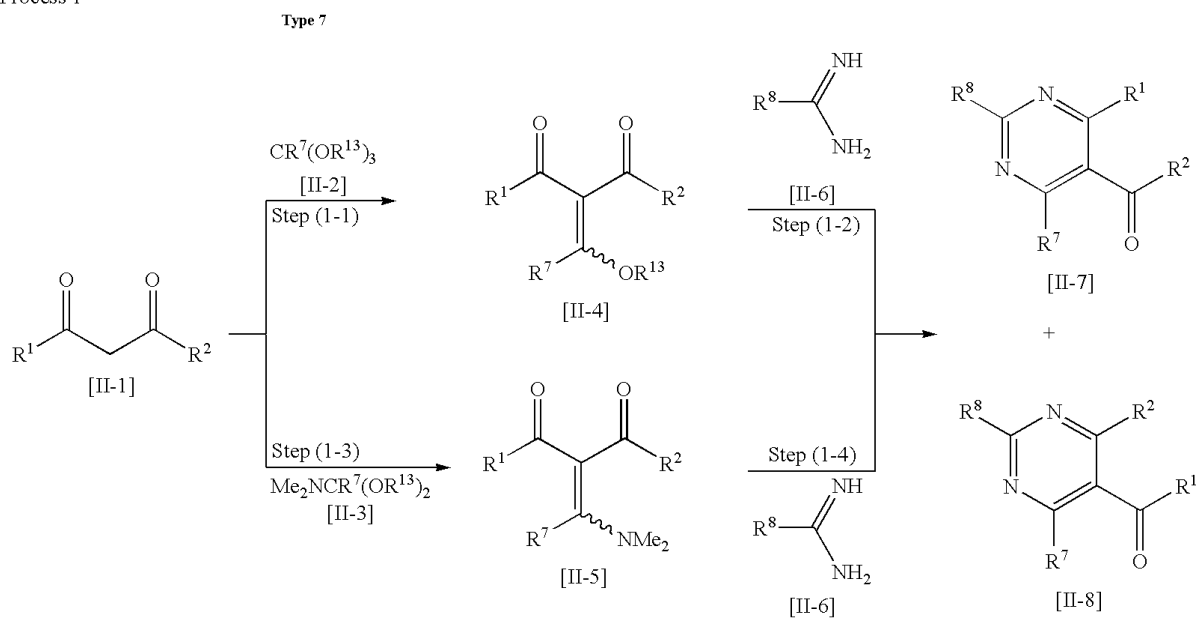


Figure 6.5: Available bounding boxes around geometric entities in the ChemDraw files distributed with USPTO patents.

Process 1



computed for each, from the size of contained atoms or text. The geometric centre and area of each box is computed and stored, and arrows are assigned a direction, *i.e.* left, right, up, down or diagonal combinations of these.

To maximize the accuracy of the extracted syntheses, we currently only process the simplest of synthetic representations. Syntheses must begin at the top of the CDX page area and flow down. We eliminate curved ('arc') arrows, multiple arrows that are connected by line segments, arrows that point to other arrows, or arrows that overlap with lines because unique directions cannot be easily assigned, as in Figure 6.6. We also reject CDX files that contain text boxes with both characters and a '+' sign, because it is unclear whether this notation describes charges, ions or a list of reagents. Text boxes containing only a '+' are kept. We remove text boxes that contain Roman numerals and whose area is much smaller than the average bounding box area, as these are common numbering schemes in patents. If the bounding boxes of two or more molecular fragments overlap or if the molecules appear on opposite sides of a text '+', then these molecules are grouped and their behavior as reagents or products are assigned together. Alternatively, if no fragments remain after filtering, we terminate the processing of the CDX file.

Once we have a clean set of molecules and arrows, we assign the product and reagent relationships to each reaction arrow. First, we calculate the average distance between the edges of bounding boxes containing either molecules or arrows. Then, for the head (tail) of each reaction arrow, we select as possible products (reagents) those molecules with an edge closer than 2 standard deviations of the average distance. Additional constraints are then considered. If the arrow points directly down, a molecule is assigned as a reagent if it is located above the arrow or if it is to the side of the arrow with the center of its bounding box between the arrow head and the arrow tail; molecules are assigned as products if they are below the arrow. If the arrow points directly to the right (left), a molecule is assigned as a reagent if it is to the left (right) of the molecule or if it is above or below the arrow with the center of its bounding box between the arrow head and the arrow tail; molecules are assigned as products if they are to the right (left) of the arrow. Alternatively, if the arrow points right and there is no molecule directly to the right, the closest molecule below the arrow is assigned as the product. For diagonal arrows, a ray is extended and the first molecule that intersects the ray from the head (tail) of the arrow is considered a product (reagent) An example is given in Figure 6.7

Once each reaction arrow is assigned products and reagents, the relationship is written out in standard SMILES format. The shape of the entire synthesis may be elucidated by following products generated in one step which are used as reagents in the next step.

Additionally, we augment the visual processing depicted in Step 2 of Figure 6.3 with text analysis of patents using OPSIN (Jessop et al., 2011, Lowe et al., 2011). OPSIN will read text descriptions of chemical syntheses and generate reaction steps similar to the output of our visual processing. However, OPSIN only processes the

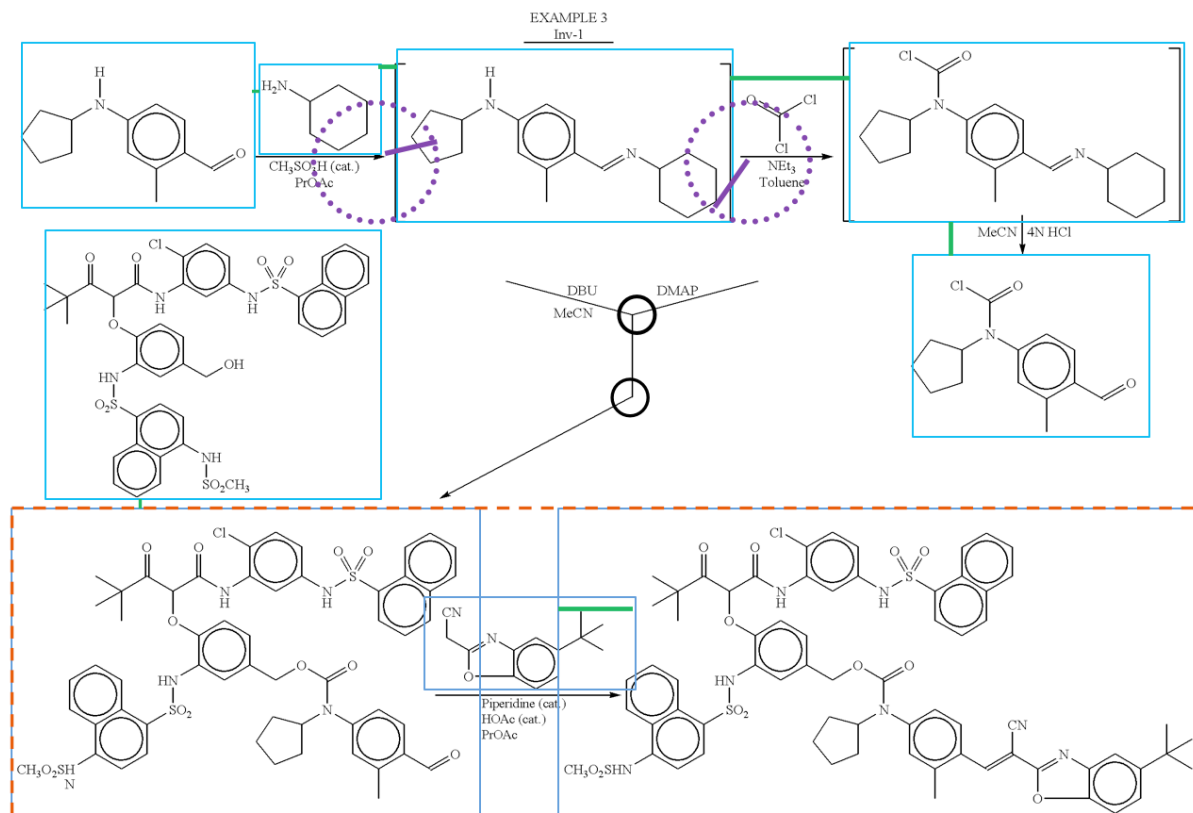


Figure 6.7: CDX reading example. First bounding boxes (in blue) are computed for each molecule. Then the minimum distance between fragments is measured (in green), and used to compute mean and standard deviation for inter-molecule distance. Molecules intersecting with 2 standard deviations from an arrow head or tail (dashed purple arrow) are considered possible reagents or products, and are classified as described in the text. The three bottom molecules overlap and, therefore, are grouped into an enclosing bounding box (in orange). Because this box encloses an arrow, these molecules are rejected. Similarly, because the central arrow intersects line segments (in the center black circles) we cannot assign unique direction for the arrow and it is rejected.

XML format used in patents from 2005 onward. An example of the patent input text is given in the caption of Figure 6.4.

6.2.2 Data cleaning

Once the synthesis from each patent is read, several steps are taken to ensure high quality output data. We begin by unifying the results from each patent are into one consistent global representation, since molecules are not described identically across all companies and all chemists. We use OpenBabel (O’Boyle et al., 2011a) to generate canonical SMILES string representations for every molecule, and then match identical structures found in different patents. For example, the product of a synthesis found in one patent may be used as a starting material in a different patent. Merging the two syntheses yields a better reflection of synthetic feasibility.

An important special case for molecular and synthetic correspondence in patents occurs when a synthesis crosses the page boundary of the physical patent document. Under such circumstances, the chemical representation of the synthesis is also split into two or more CDX files. In the patents, these will end up being sequentially numbered CDX files. We detect arrows that point off of the end of a CDX file and “glue” it together with the subsequent CDX file into a single synthesis.

We validate the extracted reactions by using the Indigo toolkit (GGASoftware, 2012) to perform an atom-to-atom mapping between products and reagents. The atom mapping describes the structural changes caused by the reaction. If a reaction product contains an unmapped atom, it is because the toolkit could not find a suitable reagent responsible for the presence of the atom. We discard any reaction that has a product containing any unmapped atoms. For example, in Figure 6.5, the author of the patent has collapsed several reaction steps above the first arrow; if such collapsing is too aggressive, the mapping algorithm will not be able to find an appropriate relationship between reagents and products, and reject the reaction step. This helps ensure that each arrow in the patent actually encodes (close to) one reaction step. This mapping-based filter may eliminate reasonable reactions, but it is more important to minimize the chance that that computer “invents” unrealistic chemistry than the computer failing to translate a particular reaction (which can be mitigated through redundancy in the data).

To help clean the data, we focus on the subset of elements that are commonly used in organic chemistry. Cheminformatic software packages can be overzealous in their attempts to translate the chemistry found in the free-form text in real-world ChemDraw files. For example, tert-butyl (^tBu) has been observed to be translated into a compound containing boron and uranium. Additionally, a compound with two alkyl substituents, R_a and R_b, may be translated into a molecule containing radium and rubidium. We reduce the occurrence of such errors by using RDKit (Landrum, 2006) to filter out reactions containing compounds that are frequently mistranslated ([Ra][Rb][Re][Rh][Ru][V][W][Y]), the noble gases ([He][Ne][Ar][Kr][Xe][Rn]), lanthanoids, and

actinoids, which rarely appear in organic syntheses.

6.2.3 Data labeling

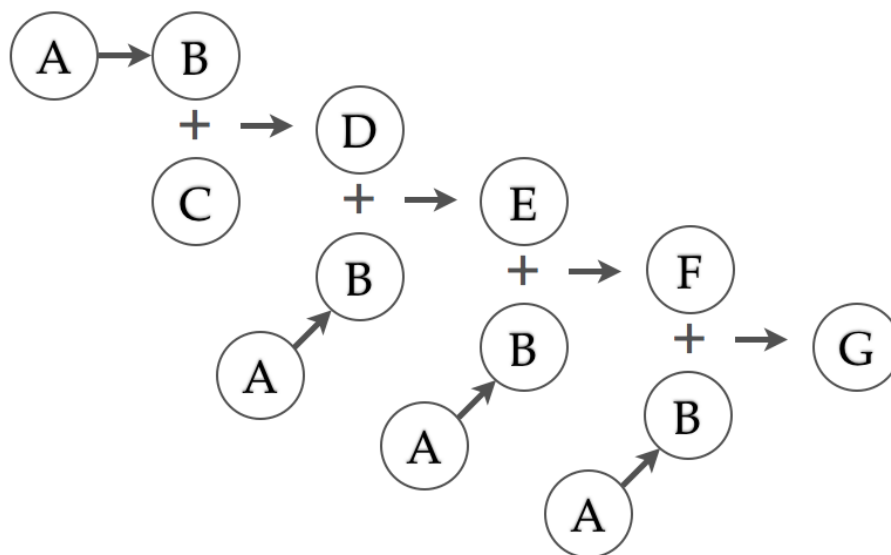


Figure 6.8: Illustrative schematic of organic synthesis. Although there are 8 reaction arrows depicted, the $A \rightarrow B$ reaction is repeated four times and it is likely that only 5 reactions were run at the bench. Assuming that A and C are starting materials, Algorithm 5 labels A and C as requiring zero steps, B requiring 1, D requiring 2, E requiring 3, F requiring 4, and G requiring 5.

Given a set of syntheses, we must compute the minimal number of reactions needed for each molecule. We use a message-passing paradigm, with the following intuition. Each molecule has a list of reactions in which it participates as a reagent. When a cheaper path to creating a molecule is discovered, the cost is assigned to the molecule and the molecule announces its new cost to each of the reactions in which it participates. These reactions compute a new total reaction cost and announce the new cost to their product(s). Each product, in turn, computes a new minimal cost and, if this is lower than the old cost, informs the reactions in which it participates. The algorithm terminates when no new messages get sent.

Finally, computation of the minimal synthetic cost is complicated because the synthesis may be a directed acyclic graph, as shown in Figure 6.8, rather than a tree. That is, key intermediate molecules may be used several times in a synthesis. These intermediates are synthesized once in large quantities and reused throughout the synthesis, so the chemist will only incur the synthetic cost once. Therefore, the proper synthetic feasibility for molecule G in Figure 6.8 is 5 reactions; however, naively counting the number of reactions arrows would yield a score of 8.

This problem is related to the overcounting problem of (dis)proof-numbers. When cycles occur or when intermediate nodes are shared along different parts of the solution path, naive solutions will sum the (dis)proof numbers of leaves multiple times along overlapping paths (Kishimoto, 2005, Kishimoto and Müller, 2004, 2008,

Algorithm 5: Proof Set Search-based algorithm for computing minimal synthesis size.

```

1 procedure ComputeMinimalSyntheticEffort
  Input:  $M$ , the set of all molecules
2    $SM$ , the set of all starting material molecules
3   foreach  $m \in M$  do
4      $m$ .precursors  $\leftarrow \infty$ 
5   foreach  $s \in SM$  do
6     UpdateMoleculeRequiredSet( $s$ ,  $\{\}$ )
7   foreach  $m \in M$  do
8      $m$ .synthesisSize  $\leftarrow |m$ .precursors|

9 procedure UpdateMoleculeRequiredSet
  Input:  $m$ , the molecule with a new synthetic route
10    $P$ , the set of all reactions needed to synthesize  $m$ 
11   if  $|P| < |m$ .precursors| then
12      $m$ .precursors  $\leftarrow P$ 
13     foreach  $r \in m$ .participatingReactions do
14       UpdateReactionRequiredSet( $r$ )

15 procedure UpdateReactionRequiredSet
  Input:  $r$ , the reaction with a newly-available reagent
16   if  $\forall m \in r$ .reagents,  $m$ .precursors  $< \infty$  then
17      $x = \bigcup_{m \in r$ .reagents} m.precursors
18     foreach  $p \in r$ .products do
19       UpdateMoleculeRequiredSet( $p$ ,  $\{r\} \cup x$ )
  
```

Seo et al., 2001). Solutions that avoid both high computational overhead and inaccurate proof and disproof counts are an area of active research (Kishimoto, 2005, 2010, Kishimoto and Müller, 2003). We solve this issue with a new message-passing algorithm, which is inspired by Proof-Set Search (Müller, 2002), and which uses dynamic programming to maintain a frontier of reactions used to produce each molecule. Unlike Proof-Set Search, which was not practical for game proofs due to large overhead, the algorithm is sufficient for the analysis of syntheses because the average industrial pharmaceutical synthesis is relatively short, at only 8.1 steps (Carey et al., 2006) and display a small branching factor, when compared with game proofs.

The minimum synthetic effort computation is shown in Algorithm 5. As in Proof-Set Search (Müller, 2002), the score of a node is represented by its set of required precursors, rather than a single integer. Rather than adding single integer scores, synthetic paths are computed by taking the union of these precursor sets which ensures that key intermediates are counted once. The synthetic feasibility is then simply the size of the precursor sets (line 8).

Briefly, we begin with a bipartite graph comprising a set of nodes representing molecules and a set of nodes representing reactions. Each molecule node maintains the set of reactions in the current best path to synthesize the node, and a list of outbound links to the reactions in which it participates as a reagent. Every molecule is initialized to have an infinitely difficult synthetic path, denoting no currently known synthetic route has been

discovered (line 4). Starting materials do not need to be synthesized so, after initialization, starting materials have their best synthesis reduced to the empty set (line 6). When the path length of a molecule drops below its current best synthetic cost (line 11), the molecule sends a message to the reactions for which it is a reagent (line 14). If any such reaction has a noninfinite path for each required reagent, it computes the union of syntheses for each required reagent (line 17) plus itself and sends a message with this new synthesis to its products (line 19).

Because reactions are invertible (*e.g.*, oxidation and reduction), organic synthesis space contains cycles. Message-passing algorithms are, in general, not guaranteed to converge under such conditions. However, for a message to be propagated on line 14 of Algorithm 5, the cost of the molecule must have been reduced (due to the inequality on line 11 and the assignment on line 12). Termination of Algorithm 5 is guaranteed because the number of steps to synthesize any molecule cannot drop below zero; therefore messages must eventually stop being sent.

An additional consideration is the set of starting materials, which the message passing algorithm requires as input. Trivially, the leaves of the synthetic tree may be considered starting materials; in Figure 6.8, this design decision would correspond to molecules A and C. However, data sets are rarely complete. In our case, the patent data begins in 2001. If the synthesis for molecule C had been published in 2000, this analysis would mislabel the advanced intermediate as a basic starting material. This error propagates: the computed cost for any synthesis using molecule C would be underestimated. To prevent mislabeling complicated molecules as starting materials, we define only simple molecules as starting materials; specifically, we consider starting materials to be molecules with no more than 10 carbon atoms. It is important to note that there is no universal set of starting materials: the same molecules that are considered starting materials inside of pharmaceutical companies are the final synthetic products of the fine chemical manufacturing industry. As with reaction yields, the set of available starting materials must be determined by the requirements and resources of each specific organization. That said, our approach still captures the idea used by starting material oriented estimators (Baber and Feher, 2004), that molecular complexity that can be purchased rather than build is easier on the practicing chemist.

6.2.4 Data modeling

The output of the preceding step yields a `pathLength` label for each molecule, although molecules can have ∞ path length in cases where no synthetic path exists from starting materials. Each molecule, including intermediates, produced in the syntheses is labelled with the number of synthetic steps that were required to create the molecule. These labelled molecules will comprise our large, objectively-derived, and physically-meaningful data set.

However, learning directly from molecules is an unexplored problem. Supervised machine learning algorithms

typically take as training input a vector of real-valued features and a label to be learned. Therefore, the first step in constructing a statistical model of synthetic feasibility is translating molecules into a vector of features. Generally, the performance of a statistical model depends critically on the selected features; to ensure that our results are both easily reproducible and independent from manual feature tuning, we have chosen our features to be commonly-used structural fingerprints. Structural fingerprints enumerate all subgraphs of a given molecule up to some fixed size. These subgraphs are assigned unique IDs, which are consistent across different molecules. A given molecule’s fingerprint is the sparse vector of present subgraph IDs, with each ID associated with the number of times that subgraph appears in the molecule. To enable reproducibility, we use standard toolkits and implementations. Specifically, we use the cheminformatics toolkit RDKit (Landrum, 2006) to generate Morgan fingerprints (Morgan, 1965) of radius 2, which correspond roughly to the ubiquitous ECFP4 (Rogers and Hahn, 2010) implementation of structural fingerprints.

We construct a feature vector, where each entry corresponds to the presence of a given molecular substructure. The value of each entry is the number of times the substructure is found in the molecule. The length of the feature vector is determined by the size of the set of all substructure fingerprints found in the training data. Taking molecules as “sentences” and substructures as “words”, the use of substructure counts precisely corresponds to the “bag of words” model used ubiquitously in computational linguistics, information retrieval, and computer vision (Manning and Schütze, 1999, page 237). More precisely, the radius 2 fingerprints correspond to linguistic length 5 n-grams, although obviously words in a sentence are arranged in a linear sequence, whereas the fingerprint subgraphs can be branched or cyclical. Although such features show strong conditional dependence, “bag of words”-based statistical models have been frequently shown to work well and, under certain conditions, achieve optimal performance (Domingos and Pazzani, 1997).

We used scikits-learn (Pedregosa et al., 2011) to implement our feature selection, machine learning models, parameter-based grid search, testing, and crossvalidation. We evaluated the performance of a number of statistical models provided by scikits-learn including support vector machines with linear and RBF kernels, linear-regression, stochastic gradient descent for least squares fit, random forest, k-nearest neighbor, and baseline prediction of the mean training label. Because the training data is highly imbalanced, we also implemented stratified (or balanced) random forests (Chen et al., 2004). We also built a version of stratified random forests where under-sampling was performed using cost-proportionate rejection sampling, where the cost of misclassifying a molecule was inversely proportional to frequency of a molecule’s class Zadrozny et al. (2003). For the purposes of sampling classes, molecules were grouped by label, except labels 8 and above were clumped together because they were so few. For statistical models with tunable hyperparameters, we evaluated several options, *e.g.* number of neighbors for kNN of 3, 5, and 10.

As in gene expression and text analysis, the number of features often exceeds the number of data points. We

adopted a standard technique from computational linguistics, namely to prune any feature that lacked a sufficient number of examples in the training data. This keeps rare substructures that only occur in families of related molecules with similar synthetic paths from being used as identifying characteristics. In our results, we required each feature to be present in at least 100 molecules.

We experimented with additional forms of feature selection to reduce the dimensionality of the data. We also explored using PCA to project the data to a lower dimensional representation and then keeping only the top-K features, and we used scikit-learn's univariate feature selection to prune all but the top 10% of features. Neither approach substantively changed the results, although fewer features sometimes led to faster completion of training. In some cases, this improved runtime permitted algorithms to complete where using the full feature set did not.

6.3 Results

We augmented the database described in Chapter 5 with patent applications and recent patents. In total, our dataset comprised 337,284 patents and patent applications, of which 128,149 had arrows in one of their CDX files. Often, patents have more than one CDX file with an arrow; in total, these patents yielded 1,238,217 synthesis-containing CDX files.

These CDX files are processed as described in section 6.2.1; 789,345 reactions are extracted via the visual processing, including duplicates from patents and patent applications. Additionally, 203,036 reactions are found via OPSIN text analysis. 52,082 reactions are eliminated by our structural filters. Reaction mapping eliminates 405,430 (or 43%) of the remaining reactions. After deduplicating reactions, we are left with 334,753 unique reactions.

We then split this data into a training set, an internal test set, and a held-out external validation set. Typically, evaluation of machine learning performance is conducted with cross-validation and a held-out data set. However, our data points are highly correlated: adjacent molecules in a synthesis are likely to be very similar. Therefore, in addition to standard random folds, we use a partitioning based on time (Sheridan, 2013) which is meant to model the practical use of the prediction system. Consider a chemist in December of 2009. She would have access to everything published up to that point, and the question is whether she would be able to predict the synthetic feasibility for the molecules of 2010. Therefore, we split the reactions into a training set comprising the data published before 2007, an internal test set comprising the data published before 2009, and a validation set comprising everything in our initial patent data. We ran the labeling procedure for each set. Since a molecule could be published twice (*e.g.*, disclosed in a patent application in 2006 and again in the granted patent in 2010), we ensured the novelty of the test and validation sets by eliminating any molecule present in the training or test,

respectively.

Path length distribution (N=40,223)

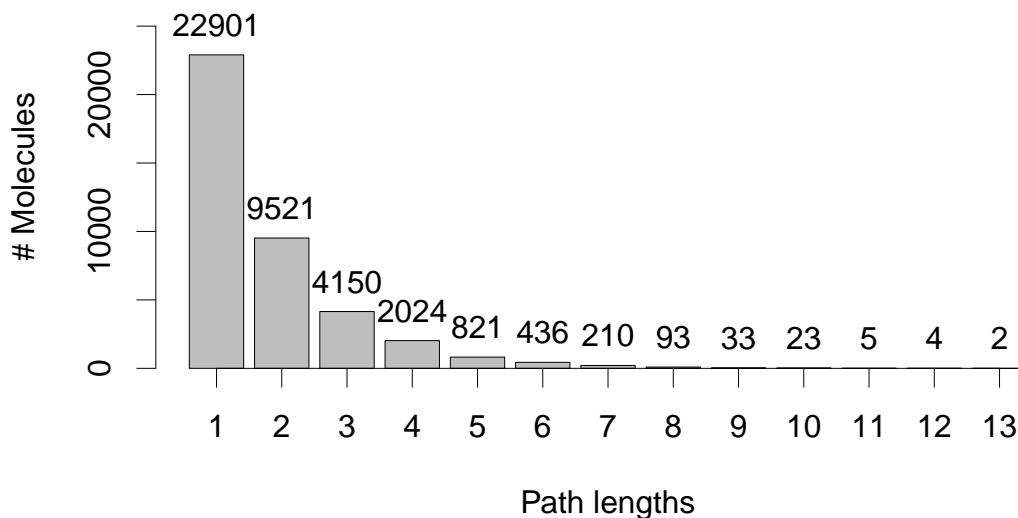


Figure 6.9: Distribution of the labels for molecules in the training data.

The labeling procedure prunes many reactions because of our strict requirement that starting materials contain no more than 10 carbon atoms. In the training data we find syntheses for 40,223 molecules, with the synthetic difficulty depicted in Figure 6.9. As can be seen, the path lengths follow a roughly geometric distribution with the synthetic difficulty highly skewed toward short syntheses. This is to be expected from our labeling of intermediates during the construction of our dataset (and in contrast to previous reports that only described the final molecule (Carey et al., 2006, Christ et al., 2012, Roughley and Jordan, 2011)); a 5-step linear synthesis also contains a 4-step synthesis of the penultimate molecule, the 4-step contains a 3-step and so on. This skew has not been discussed in previous work describing the manually-curated synthetic feasibility datasets. The internal test set data permits us to label 43,976 molecules, with 3,109 molecules not present in the training data. The validation data permits us to label 49,048 molecules, with 5,280 molecules not present in the internal test set data.

As discussed, the number of features is greater than the number of molecules. In the training data, 40,223 molecules contain 75,624 features. In the internal test set, the 43,976 training examples have 81,542 features. Most of these are singletons; when we prune any feature that is not present in 100 molecules, we keep 1,422 features for the training set and 1,509 features in the internal test. (When we use a particular set as training, any novel features in the test molecules are ignored.) Not only does reducing the feature counts make the learning process tractable, but it reduces the likelihood of overfitting. As can be seen from Figure 6.9, the cutoff of 100

Algorithm	Internal Test		Cross Validation		Parameters		
	RMSE	Weighted RMSE	RMSE	Weighted RMSE	PCA	Percentile	Settings
kNN	1.09	11.02	0.93	12.26	Y	Y	$k=3$
SRF	1.97	12.47	1.94	6.63	N	N	$e=20, s=100$
CPRSRF	1.84	12.73	1.99	5.37	N	N	$e=500$
RF	0.95	25.70	-	-	N	N	$e=40$
GBR	0.98	27.05	0.99	18.10	N	N	-
SVM	1.00	31.11	-	-	N	Y	kern=RBF
baseline	1.21	48.83	1.23	33.22	N	N	-

Table 6.1: The best synthetic feasibility prediction algorithms on the internal test set. The first three were chosen as the algorithms that yielded the lowest weighted RMSE and, for comparison, the second three yielded the lowest RMSE. Cross validation was performed on 3 runs of 5 folds; some algorithms did not complete due to the increased computational time and are denoted with dashes. kNN is k-nearest neighbor; SRF is stratified random forest; CPRSRF is cost-proportionate rejection sampling random forest; RF is random forest; GBR is gradient boosted regressor; SVM is support vector machine. PCA and Percentile describe feature selection methods. Under “Settings”, k denotes the number of neighbors, e the number of estimators in an ensemble method, and s the size of the (up- or down-) sampled classes.

molecules is quite strict; in particular, it implies that even if there were a feature that uniquely identified the molecules labeled 8 or above, our statistical models would not be able to use the feature.

As discussed in Section 6.1.1, standard practice is to compare heuristics against the average of scores given by a group of chemists. Figures 6.1 and 6.2 demonstrate that these averages have real values rather than categorical values. We could treat the prediction task as a classification problem rather than regression: our machine-generated labels are ordinal. Indeed one could see an advantage in framing the synthetic feasibility prediction problem as a classification task, since a synthesis of 4.5 steps has no physical meaning. However, a predicted score of 4.5 *is* meaningful: namely that the molecule looks more difficult than what the model considers to be 4-step molecules but less difficult than 5-step molecules. Therefore, we keep to established practice and treat the prediction as a regression.

As is typical in regression tasks, we report root mean squared error on our internal test set, rather than classification metrics such as precision and recall. However, because we have a skewed distribution, nearly all of the score comes from the data points with a label of 1 or 2. The effect is that the systems are tuned to predict low scores, and predictions for molecules with high labels become very inaccurate. In fact, a baseline prediction of the mean label (1.76) performs fairly well in terms of RMSE. Therefore, we additionally computed a weighted RMSE where rare data points are overweighted so that every score label has equal weight. (This is equivalent to scoring, e.g. the score 13 molecules in Figure 6.9 11,450.5 times each.)

We used the internal test set to choose hyperparameter settings for our machine learning algorithms. The results of the best parameter settings for the algorithms are reported in Table 6.1. We then used those settings to train on the data in the training and internal data, and report RMSE for the external validation set. We also tested the algorithms on 3 runs of 5-fold cross validation. However, due to redundancy in the data, cross validation

overestimates the accuracy of the predictions (Sheridan, 2013).

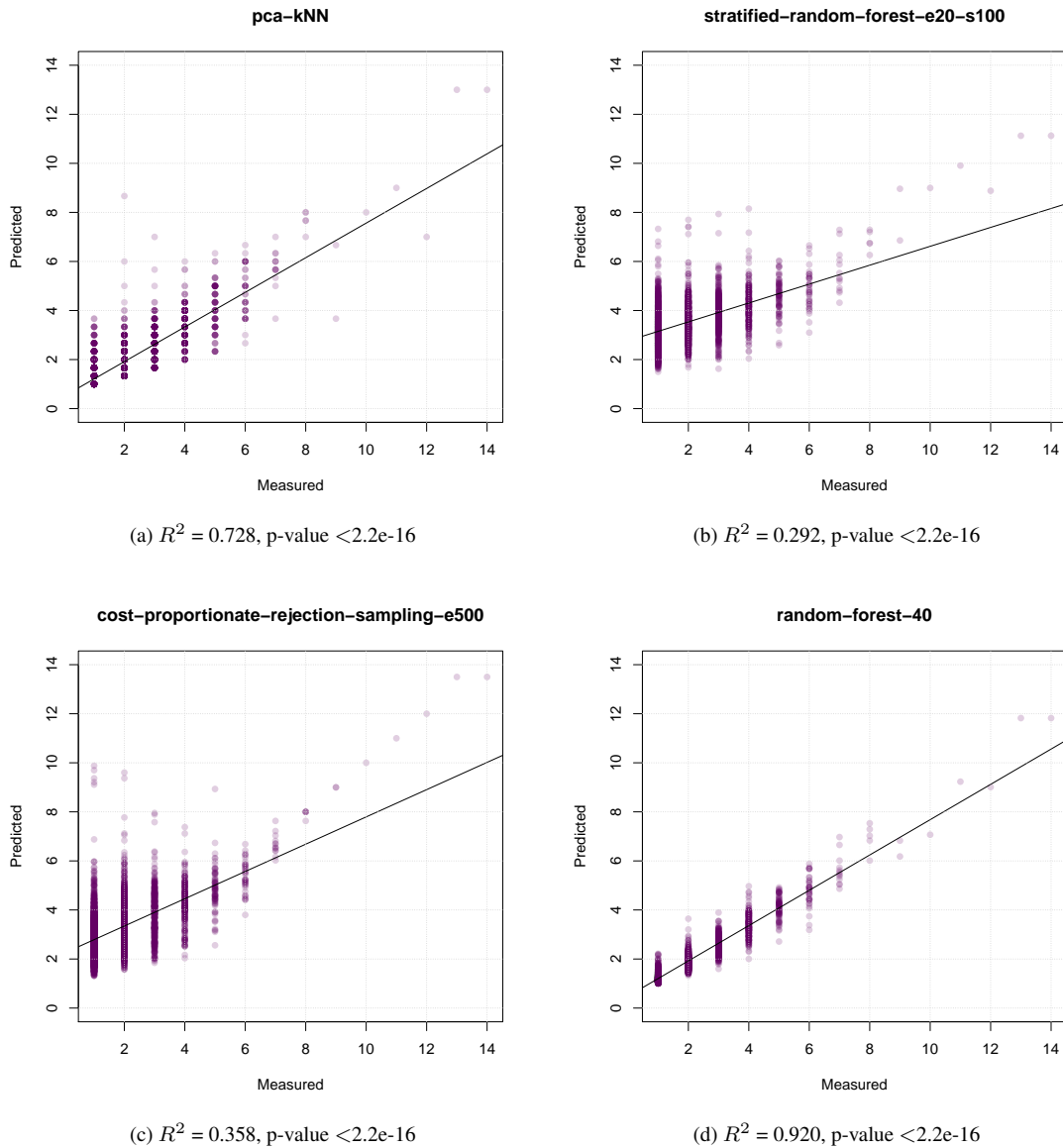


Figure 6.10: The predictive performance on the automatically-labelled training data for the top 3 algorithms from Table 6.1 and random forest for comparison. The plots were generated with R; the linear fit, R^2 and p-values were computed using R's `lm` linear model.

6.4 Discussion

The first question for a machine learning method is whether the features it has available are sufficient to capture the variation in the data. We show in Figure 6.10 the predictions of the machine learning systems on their training data (that is, data they have already seen). Distinct trends can be seen, but there remains significant variation in the predictions.

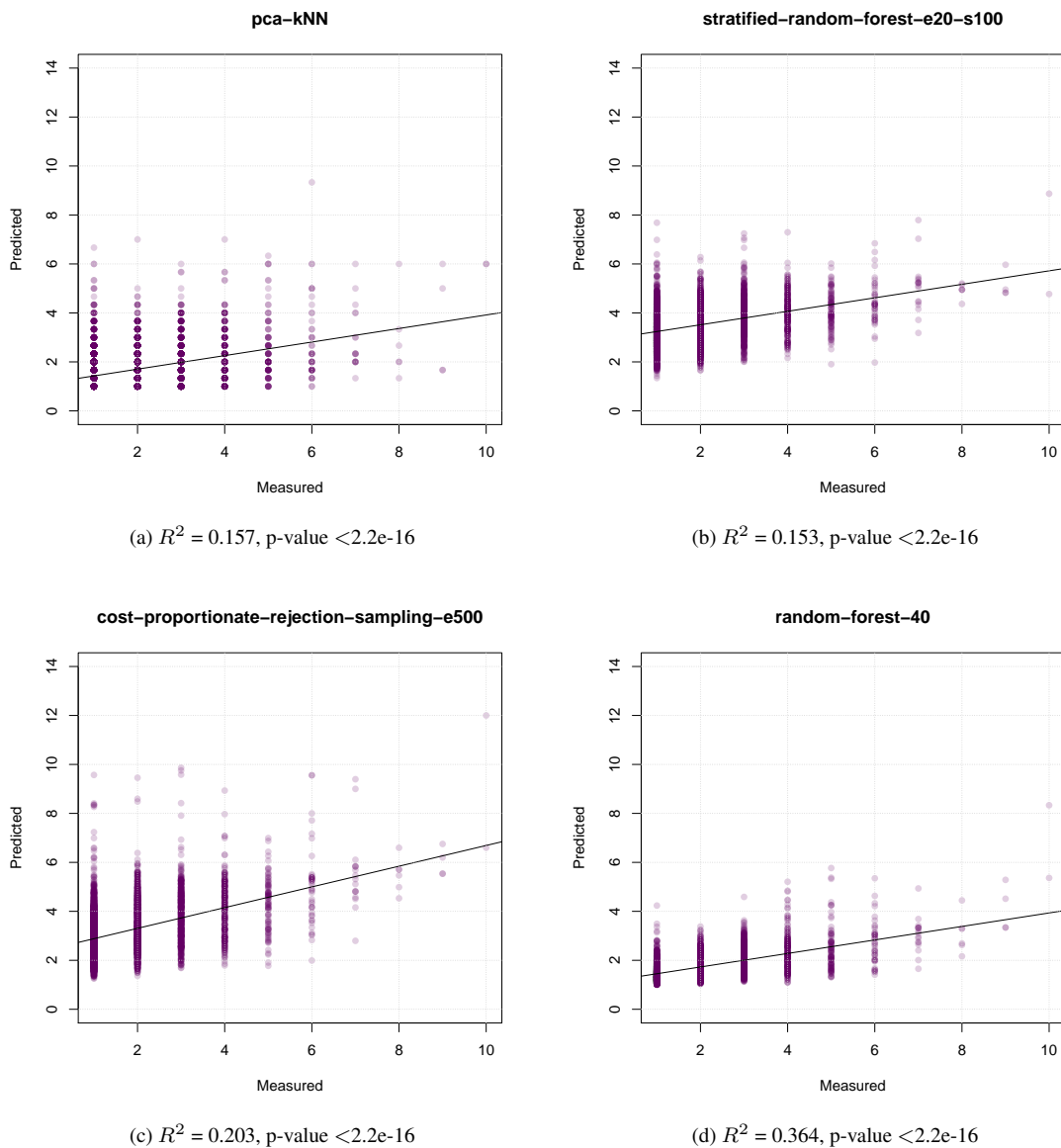


Figure 6.11: The predictive performance on the automatically-labelled external validation data for the top 3 algorithms from Table 6.1 and random forest for comparison. The plots were generated with R; the linear fit, R^2 and p-values were computed using R's `lm` linear model.

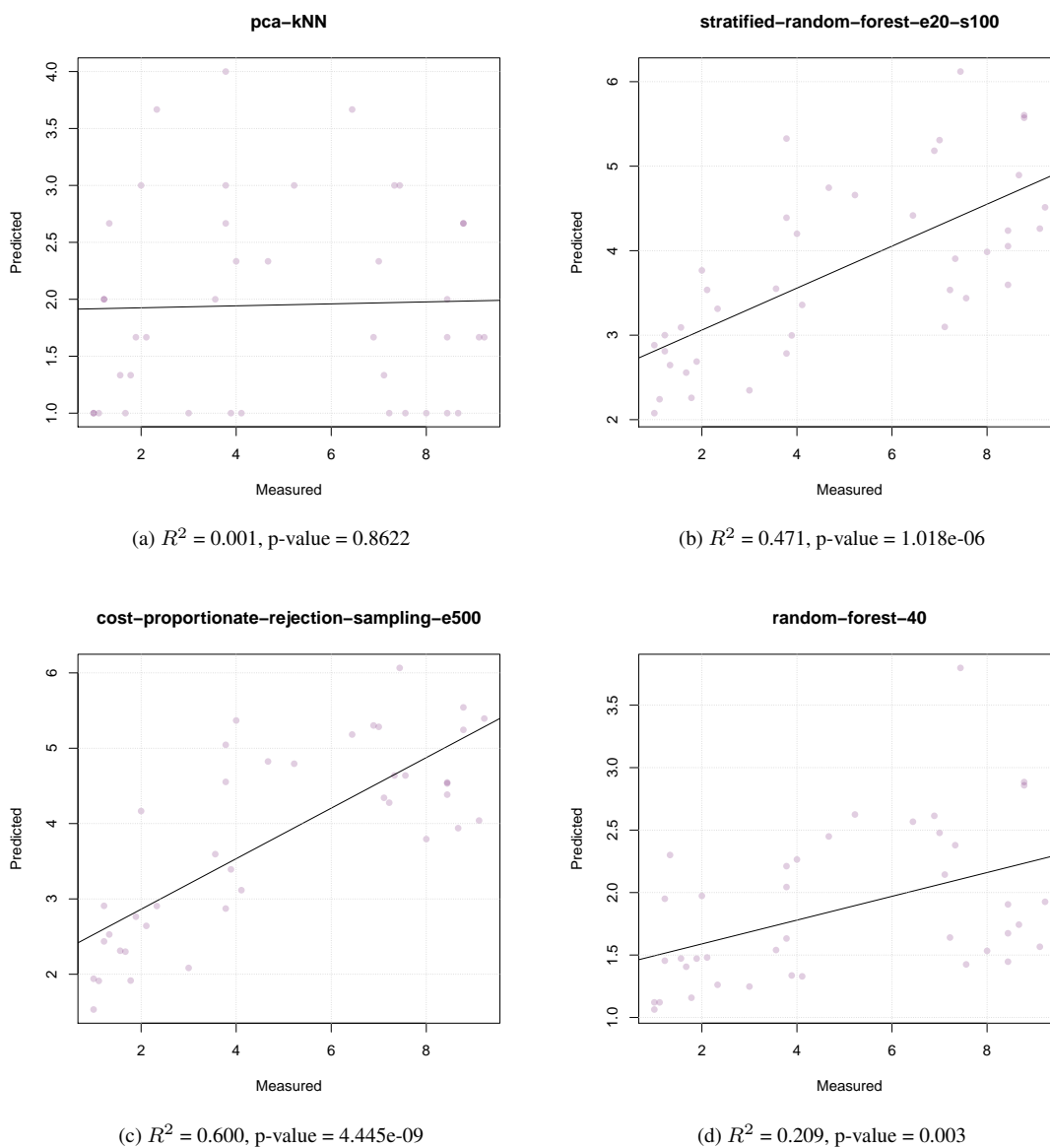


Figure 6.12: The predictive performance on the manually scored data from Ertl and Schuffenhauer (2009) for the top 3 algorithms from Table 6.1 and random forest for comparison. The plots were generated with R; the linear fit, R^2 and p-values were computed using R's `lm` linear model. For comparison, Ertl and Schuffenhauer (2009) report that the r^2 among 9 chemists on these 40 molecules ranged from 0.450 to 0.892, with an average of 0.718.

The mean of the predictions is higher for the algorithms that are aware of the skewed data distribution. (This can be seen in the intercept of the linear model.) This leads to a better accuracy for higher-labelled molecules. For example, the `cost-proportionate forest` has a quite good accuracy for high-cost molecules; this effect is summarized in the nearly-doubled weighted RMSE between CPRSRF and the unbalanced `random forest` in Table 6.1 There seems to be a trade-off in accuracy between the numerous low-cost molecules and the rare high-cost molecules. This is apparent when we compare the `stratified` and `cost-proportionate forests` to the performance of `pca-kNN`: *e.g.*, the `cost-proportionate forest` has a very tight grouping for the molecules labelled above 8 but a larger spread for molecules labelled 1, 2, and 3. Similar trends are present for the external validation set in Figure 6.11. As expected, these trends are weaker, but each of these trends is still statistically-significant.

More interesting is the performance of the predictors, depicted in Figure 6.12, on the set of 40 manually-labelled molecules from Ertl and Schuffenhauer (2009). In some sense, this is the most rigorous test of predictor performance, because humans must be convinced that the predictions of the model correspond to labels they would give. Additionally, these molecules and labels were generated independently of our labeling procedure and, therefore, provide an important check against systemic errors or biases. This data set is also interesting because, in contrast to our data, contains as many difficult molecules as easy ones. On this dataset, the relative performance of the imbalance-aware algorithms (`stratified` and `cost-proportionate forests`) is even better than the imbalance-blind algorithm (`k nearest neighbor` and `random forests`). Importantly, our algorithmic performance is comparable to the inter-chemist score agreement for this data set. Ertl and Schuffenhauer (2009) report that the amount of variance that one chemist would explain for another chemist's scores ranged from 0.450 to 0.892. The `cost-proportionate` and `stratified forests` have an R^2 of 0.471 and 0.600, respectively, and fall within the range of agreement between human experts.

6.5 Chapter Conclusion and Future Work

We have developed a method to economically generate large amounts of synthetic feasibility data that avoids problems with human biases and inconsistency. We have demonstrated that the data carries real information about synthetic feasibility, by demonstrating predictions in agreement with that of human chemists.

Currently, our system requires ChemDraw files from the Complex Work Units of US patents and patent applications. Much larger data sources already exist, such as patents from other countries and synthesis schemes in Wikipedia images. Such data could be harnessed by extending our processing to read images of chemical structure directly (Filippov and Nicklaus, 2009), rather than from CDX bounding boxes. Even without additional programatic effort, the amount of US patent data increases yearly, as shown in Figure 5.1, so the techniques

described here should improve over time.

In addition to the quantity of data available for analysis, the accuracy of predictive methods hinge critically on the features presented to the statistical learner. Here, I presented machine learning methods with standard structural features from computational chemistry, the ECFP-style subgraphs of radius 1 and 2. These structures were designed for predicting pharmacokinetics and pharmacodynamics, rather than synthetic feasibility. It would be interesting to see if other molecular features, designed specifically for synthetic feasibility, would be better at predicting the effort needed to make a given molecule.

Chapter 7

Summary & Conclusion

Organic synthesis planning is a difficult problem. In fact, the general case is undecidable. It may therefore seem foolhardy to devote a PhD to solving the problem computationally. However, we may still have hope because we can build useful tools without needing to solve all instances optimally.

Unfortunately, even the creation of algorithms that are useful in practice is challenging. Organic chemistry is far larger than most problems investigated today, whether we measure the number of variables in our state descriptions, the size of the search space, the average branching factor, or the computational effort to determine attainment of goal nodes. Additionally, in computer science, we are used to leveraging pre-defined data sets or on working on problems where data collection is straight-forward. Chemistry is a rich and complicated field, with most information not publicly available, and it is laborious to encode a sufficiently large subset in formats that are amenable to automated analysis.

Despite these difficulties, computer science can contribute to organic synthesis planning. First, as shown in Chapter 5, we can - with effort - compile and analyze far larger datasets than a person ever could. The extracted information is useful for efforts in medical text mining, chemical image parsing, reaction extraction, the development of computational tools for lead optimization, and - most importantly for our efforts - the derivation of synthetic feasibility heuristics. Powerful heuristics have yielded the biggest impact in other applications of automated computer reasoning and, since chemistry is such a large domain, they are particularly important here.

I present two novel heuristic classes for organic synthesis. In Chapter 4, we describe proof-number-based search which chooses which search nodes to expand so as to minimize required exploration effort. It is a heuristic method which balances the effort of finding syntheses for needed molecules against the effort to show that a current line of attack is doomed to fail because a needed molecule is unsynthesizable. This was the first application of proof-number methods to chemical synthesis planning and we showed that it outperformed previous unguided

search systems. However, the efficiency of proof number search is improved if the proof numbers are heuristically initialized to values that are close to the eventual effort needed for that node. To push the utility of heuristic search further, more precise heuristics are needed.

I therefore set out to derive a heuristic to predict the synthetic effort for a molecule, which could be used to initialize the proof number search. Previous heuristics were based on human opinions, and people have opinions which often do not correlate with other people or, indeed, with their own opinions when asked later. There is a distinct lack of ground truth in this field. I therefore built the database described in Chapter 5 with the express purpose of extracting information about syntheses. This need informed the core design decision that differentiates my database from other databases of molecules such as PubChem (Wang et al., 2009): to track metadata about the molecular relationships, which permits the extraction of reactions and, therefore, syntheses.

Having assembled the largest extant dataset that describes how hard it is to make molecules, I could now create my predictor of synthetic feasibility. This heuristic is given in Chapter 6, where I derive a heuristic to predict synthetic feasibility from this objective data. Without the database assembled in Chapter 5, this fundamentally new approach would have been impossible. Previous synthetic feasibility assessment systems were based on manually-annotated molecular scores, often on a scale from 1 to 10. Such rankings are not directly comparable to synthetic effort and it is not straightforward to incorporate such scores in heuristic search algorithms. In contrast, our heuristic is the first to use the same units for prediction as a heuristic planner would use for searching: number of required synthetic steps. It is therefore the first heuristic to be appropriate to serve as the heuristic in an heuristic search planner (that is, the h in $f(n) = g(h) + h(n)$). However, even as a stand alone tool, metrics for synthetic feasibility can prove be useful for chemists who want to rank or filter sets of molecules.

Indeed, as with every PhD thesis, much future research remains to be done. I describe new approaches for retrosynthetic search and synthetic feasibility heuristic estimators. The proper exploitation of these new techniques may lead to planners that are more powerful than previous attempts. Therefore, the ultimate goal of research in automated organic synthesis planning should be a deeper integration of heuristic search techniques.

While the investigations described in this thesis extend the range of what is currently possible, much more effort will need to be invested to make practical contributions to chemistry. Improvements can also be made within each of the research topics investigated in this dissertation. Reaction libraries must become larger, and the predictive accuracy of the synthetic feasibility heuristics could be improved. A statistical model's predictive ability depends on the quantity and quality of data with which it was built. The well known quip that there is no data like more data holds here, too, and the databases I build could be extended with synthesis data from other sources, such as patent offices in other nations, academic papers, and Wikipedia. Indeed, the amount of US patent data grows each week, so simply waiting and applying the techniques I describe in Chapters 5 and 6 will yield more data. The approaches and databases described herein should scale as more information becomes available

and their utility should scale along with it.

Even so, most of the existing data is excluded from analysis because the syntheses do not begin at sufficiently simple starting materials. New machine learning algorithms for learning relative rankings directly, rather than assigning scores as in Chapter 6, could additionally leverage this excluded data. Also, we used standard cheminformatic features to aid in the reproducibility of this work, but the accuracy of statistical models often hinges on features tuned to particular domains. Finally, the largest public collection of syntheses is a by-product of the work described in this thesis; the accuracy and utility of generated retrosynthetic analyses could be improved by extracting transformations from these, as described in Section 2.7.

These steps would maximize the benefit of merging Proof-Number-Search with powerful heuristic initializations and help achieve what is always our hope: to transcend what is currently possible.

Appendices

Appendix A

Glossary

Actions Operations which may be performed by an agent to affect the world.

Arc An edge in a graph. A 2-set of graph nodes. A directed edge is a 2-tuple, with a distinguished start and end node.

Automated perception Perception is the recognition of prespecified structures of interest. For example, in mammalian and computer vision, low level perceptual filters detect lines at specific angles. Automated perception is programatic perception.

Best first heuristic search When an evaluation function exists for states in a search problem, and the choice of which node to expand in the search is the node with the best evaluation.

Branched planning domain Any planning problem where solutions are composed of independent subsolutions. For example, for a plan to succeed in nondeterministic planning, a plan has to properly account for the various possible outcomes of taking a specified action. Alternatively, in game proofs, a (possibly different) response may be necessary for any opponent move.

Bredt's rule In bridged rings which contain fewer than 8 atoms, a double bond cannot be connected to a bridgehead atom.

Checking whether a state satisfies the goal conditions Applying the goal test to a state.

Domain-independent planning A subfield of artificial intelligence which investigates the automated discovery of solutions to problems. These solutions take the form of sets of actions that are chosen depending on the state of the world. In contrast to other fields of artificial intelligence, where knowledge specific to the domain helps guide the discovery of solutions, domain-independent planning derives its guidance only at

planning time from the description of the domain and problem. For example, domain-specific chess players may use large database of chess openings. In contrast, domain-independent planners may solve a simplified (*relaxed*) chess game to derive insights for the full game.

Hierarchical Task Network (HTN) planning A style of planning where complex actions are recursively decomposed into sets of smaller actions, with dependencies specified between the smaller actions.

Independently Not connected with another or with each other; separate. (Apple Dictionary version 2.1.3)

Markov Decision Problem The problem of calculating an optimal policy in an accessible, stochastic environment with a known transition model (Russell and Norvig, 1995).

Masked Hidden.

Non-deterministic planning A planning problem where a single action may lead to a set of states. For example, flipping a coin may lead to one future world where the coin landed showing its obverse or another future world where the coin landed showing its reverse.

Ordinal regression A type of regression for predicting relative orderings. Commonly, the output is a predicted ranking over a set of items, but some algorithms return a partial ordering rather than a total ordering.

Probabilistic interpretation (in nondeterministic planning) Nondeterministic planning permits actions to have multiple outcomes. Often, these outcomes have some probability distribution associated with their occurrence. If we accept solutions which merely *may* succeed (or seek the solution with maximum likelihood of success), then we can choose algorithms which reason over the probabilities of the outcomes of actions. However, if we seek solutions which are *guaranteed* to succeed, then we must ignore the probabilities of the outcomes and reason only about worst-case possibilities. See Section 2.2.3 for a more in depth discussion.

Pseudorings When two or more rings are fused in a molecule, the outer boundary is also a ring and may be labelled as such by naive cheminformatics programs. However, this ring is usually not thought of as a ring by practicing chemists. For example, most reactions that depend on whether a group of atoms are in a ring operate on the smaller constituent rings and not the “convex hull” ring.

Retrosynthetic disconnections A retrosynthesis proceeds backwards to simple starting materials from the goal molecule. Therefore, a retrosynthetic disconnection corresponds to the formation of bonds in the forward, synthetic direction.

Situation calculus “Situation calculus is the name for a particular way of describing change in first-order logic. It conceives of the world as consisting of a sequence of situations, each of which is a ‘snapshot’ of the state

of the world.” (Russell and Norvig, 1995). For a more complete description, see (Russell and Norvig, 1995, Chapter 11).

State description Information about the world that uniquely determines a state. In planning problems, such information is often represented as conjunctions of Boolean predicates. In chess, for example, the game state would be described by the positions of the pieces, whether castling for each player is still permitted (whether each side has moved its king and rooks), and which player’s turn it is to move.

Strategic bonds In a retrosynthetic analysis, the bonds which decompose the molecule into substantially simpler pieces.

Synthetic accessibility The ease of producing a molecule. (Also called synthetic accessibility, synthetic complexity, or synthetic tractability.)

Synthetic scheme See ‘synthesis’.

Synthesis A succession of chemical reactions that constructs a desired molecule from simple commercially-available starting materials.

Synthesis planning problem The determination of a synthesis, given a goal molecule, a set of reactions known to the chemist, and a set of available starting materials.

Transition model The set of probabilities associated with the possible transitions between states after any given action (Russell and Norvig, 1995).

Appendix B

Which molecules should be built?

Molecular appropriateness is an orthogonal approach to minimizing the number of molecules we need to consider for synthetic feasibility analyses. Judgements of molecular appropriateness for interaction with particular protein targets hinge on an understanding of the critical amino acid residues that govern ligand binding. In this appendix, I describe a tool to find conserved residues in protein active sites, and a demonstration of its use in the previously-undescribed annotation of basic residues in the binding of NAD.

Ligand-based active site alignment is a widely adopted technique for the structural analysis of protein-ligand complexes. However, existing tools for ligand alignment treat the ligands as rigid objects even though most biological ligands are flexible. We present LigAlign, an automated system for flexible ligand alignment and analysis. When performing rigid alignments, LigAlign produces results consistent with manually annotated structural motifs. In performing flexible alignments, LigAlign automatically produces biochemically reasonable ligand fragmentations and subsequently identifies conserved structural motifs that are not detected by rigid alignment.

B.1 Introduction

Comparison of macromolecular structures yields insights into protein function and evolutionary relationships. These comparisons are typically performed after a global alignment of the target protein structures. Unfortunately, the non-local geometric constraints imposed by a global alignment will often prevent the discovery of locally conserved active site structure. This behavior is undesirable in the context of studying protein-ligand binding as active sites are often more conserved than the global protein shape Denessiouk et al. (1998), Kobayashi and Go (1997). One solution is to restrict alignment to the active site which induces a local alignment and reveals functionally relevant conserved structures.

One established method of local alignment is the ligand-based alignment of protein active sites. In ligand-

based alignment, two or more protein-ligand complexes are superimposed by computing the transformations between the bound ligand of each complex. The computed transformations are then applied to the unbound proteins. The superimposed structures can then be examined to identify conserved geometric relationships among chemically similar residues. Consistent arrangements of amino acids provide evidence of shared protein function.

We define two categories of ligand-based protein alignment: rigid and flexible. The rigid alignment of two protein complexes requires first determining a pairwise atom correspondence between the two ligands and then identifying the single transformation which minimizes the Root Mean Square Deviation (RMSD) of the corresponding atoms. Previous work in ligand-based protein alignment has used the rigid model. In contrast, this work presents the first algorithm for flexible ligand-based protein alignment. In the flexible model, each ligand is considered a group of rigid fragments connected by hinges. After automatically identifying the hinge locations, a rigid ligand-based protein alignment is performed for each rigid fragment. Conceptually, the flexible model independently aligns the subcavities surrounding corresponding rigid fragments, thereby allowing the identification of conserved structure proximal to each fragment. We propose that flexible ligand-based protein alignment should be the method of choice when comparing complexes with flexible ligands.

Rigid ligand-based alignment of proteins has been used to uncover the evolutionary history between protein families Nebel et al. (2007), to predict protein function Nebel et al. (2007), to extract common structural patterns for ligand binding Carugo and Argos (1997), Denessiouk et al. (2001), Kuttner et al. (2003), and to explain enzymatic substrate specificity Koehler et al. (1997). These investigations considered ligands with only a single chemical moiety: heme in human cytochrome P450 CYP17 Nebel (2005, 2006), adenine Denessiouk et al. (2001), Kuttner et al. (2003), Nebel et al. (2007), and the bioactive conformation of Glutathione S-Transferase inhibitor analogues Koehler et al. (1997). Their approaches consisted of two phases. In the first phase, they performed a rigid alignment of these rigid ligands to produce a single superposition of the complexes. In the second phase, a number of closely related strategies were employed to identify conserved binding motifs. For example, after alignment, Kuttner et al. discovered conserved regions by iteratively identifying and removing the most densely populated 1.5\AA sphere of protein atoms. As in k -median clustering, an exemplar atom was chosen for each group of atoms near the center of the group. In another example, Nebel computed atom clusters by iterative pairwise protein comparison and elimination of distant or chemically dissimilar atoms, similar to average-linkage hierarchical agglomerative clustering. After the clustering terminated, a consensus virtual atom denoting the cluster was generated by averaging the positions of the remaining atoms. In contrast to these examples of explicit clustering, Koehler et al. estimated and compared the electrostatic and lipophilic potentials within the aligned binding sites.

The previous work succeeded because it applied a rigid alignment to effectively rigid ligands. Flexible ligands may be effectively rigid if, in practice, they exclusively bind in highly similar poses Koehler et al. (1997). For

example, in the rigid alignment of the flexible ligand NAD, Carugo and Argos report conserved structure around the well-aligned adenine moiety but not near the disordered nicotine nucleotide Carugo and Argos (1997). In other words, conserved structure is only found where the ligands happen to assume highly similar conformations. Therefore, rigid ligand-based protein alignment and its use of a single computed transformation is not ideal for the study of flexible ligands. Unfortunately, there are a large number of ligands with at least one internal hinge, and these ligands bind in widely varying conformations Erickson et al. (2004), Kahraman et al. (2007), Najmanovich et al. (2008), Stockwell and Thornton (2006).

We present a new algorithm, LigAlign, for both rigid and flexible ligand-based protein alignment and comparison. Our work is the first to use the flexible alignment of ligands to detect conserved protein structure. LigAlign automatically computes an atom correspondence, identifies hinges, determines rigid fragments, computes the transformation for each fragment, and detects clusters of conserved residues. LigAlign implements a novel method for computing the correspondence between ligand atoms using a neighborhood-aware bipartite matching. LigAlign efficiently locates hinge sites using dynamic programming and branch-and-bound search and detects conserved residue clusters via clique enumeration. Notably, the use of dynamic programming for flexible ligand alignment was previously investigated for pharmacophoric inference over unbound ligands in the Pharmagist system Schneidman-Duhovny et al. (2008). While Pharmagist's analysis does not involve protein structure nor does it perform a ligand-based protein alignment, its dynamic programming approach to rigid fragment assembly is similar to LigAlign's use of dynamic programming to identify hinge sites.

We first validate our ligand alignment and cluster detection on the rigid heme system analyzed by Nebel Nebel (2006) and show that LigAlign can automatically generate a set of residue clusters consistent with the manually annotated profile. We then compare rigid and flexible ligand alignment of the proteins from the Carugo and Argos test set Carugo and Argos (1997). When LigAlign performs a rigid alignment, the results are consistent with previously described binding patterns. More importantly, when LigAlign performs a flexible alignment, it identifies additional clusters of conserved residues that are undetected by rigid superposition.

LigAlign's source code is freely available under the GNU LGPL at <http://compbio.cs.toronto.edu/ligalign>. LigAlign is implemented in Python and supported on Apple OS X, Linux, and MS Windows. Our software is implemented as an extension to the PyMOL molecular visualization software DeLano (2002) permitting integration with other PyMOL-based tools and allowing the user to easily generate high quality visualizations of the alignments.

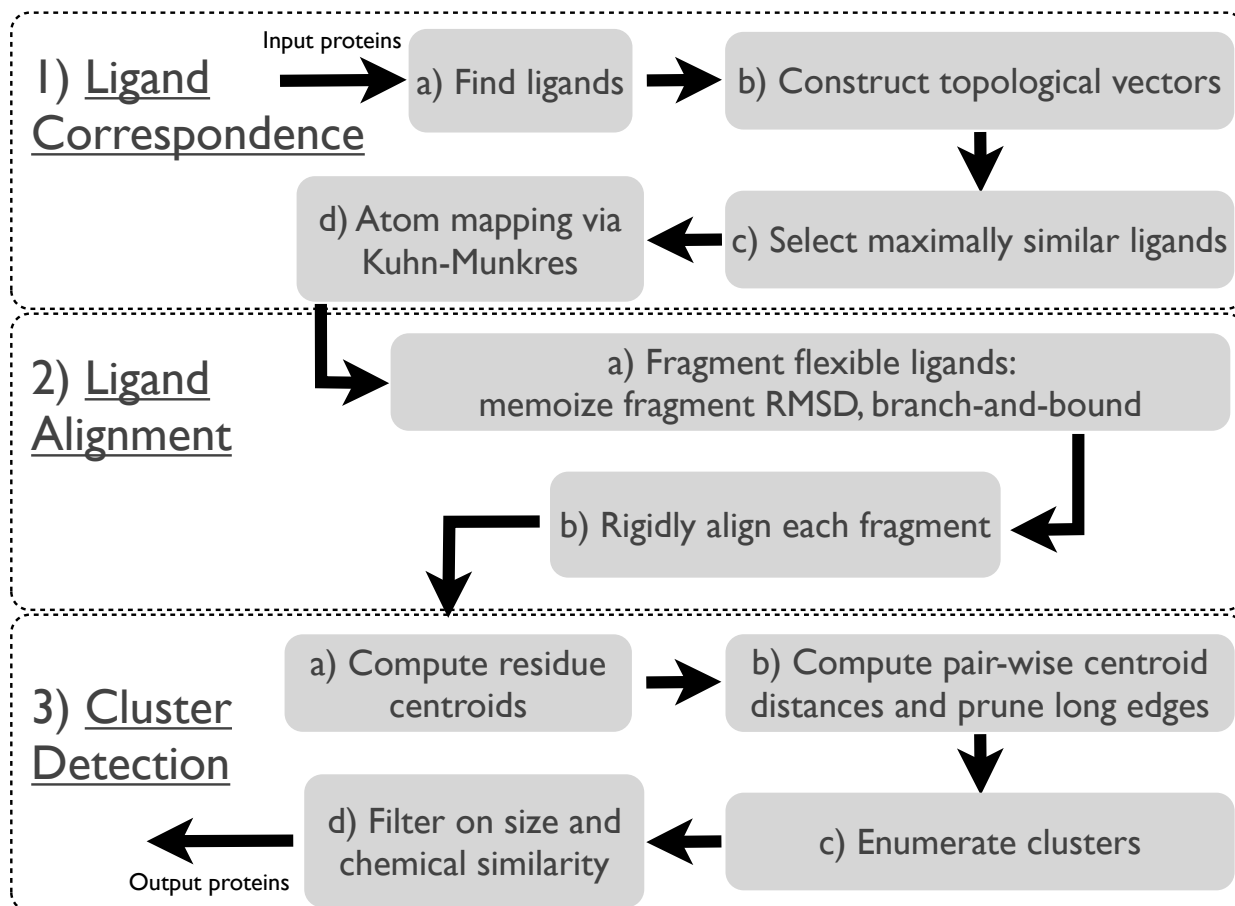


Figure B.1: LigAlign Flowchart. The three stages of ligand-based active site alignment, namely ligand correspondence, ligand alignment, and cluster detection, are shown along with the techniques used to accomplish each stage.

B.2 System and methods

Flexible ligand-based protein alignment and analysis (which we will now simply refer to as flexible alignment) in LigAlign consists of three stages (Figure B.1). Stage 1, ligand correspondence (Section B.2.1), computes a mapping between the atoms of the bound ligands in each input structure. Stage 2, ligand alignment (Section B.2.2), automatically detects hinges and aligns the identified rigid fragments of each ligand. Finally, Stage 3, cluster detection (Section B.2.3), identifies structurally and chemically conserved residues.

B.2.1 Correspondence of bound ligands

The goal of the ligand correspondence stage (Figure B.1) is to identify a set of ligands (one per protein complex) and an associated mapping between the corresponding atoms of each ligand. While the selected ligands are not required to be identical, they should be similar enough that when superimposed, they induce a clear alignment on their bound proteins and facilitate the identification of conserved protein structure. An automated system should select the most sensible single ligand from the multiple molecules frequently present in each Protein Data Bank (PDB) file. In practice, we find that LigAlign identifies the desired ligand from each protein complex, avoiding manual intervention.

Ligand extraction

The LigAlign system extracts the set of ligands that is most self-consistent across the submitted structures. This ligand extraction problem may be solved by reduction to weighted maximal-clique where nodes represent ligands and edge weights are proportional to the computed ligand similarity. Unfortunately, solving the NP-complete maximal-clique problem is prohibitively expensive when aligning multiple proteins with many ligands.

Instead of an exhaustive search, our ligand extraction algorithm identifies a *pivot* ligand in the first protein complex (the pivot protein) and the ligand from each non-pivot protein that is most similar to this pivot. We first eliminate a number of distracting molecules by assuming that ligands of interest must contain at least six heavy (*i.e.*, non-hydrogen) atoms. Ligands with fewer than six heavy atoms, such as lone metal ions or water, carry too little information to unambiguously align the proteins. To identify the pivot ligand, we compute a distance score (described in Section B.2.1) between each ligand in the pivot protein and every ligand in each non-pivot protein. The pivot ligand is chosen as the ligand in the pivot protein that minimizes the sum of the distance scores between itself and the most similar ligand in each non-pivot protein. Ties are broken in favor of larger ligands.

Atomic and molecular similarity

Our algorithms for atom correspondence and ligand extraction require distance measures that are both accurate and robust to minor structural variations. LigAlign derives its measures of atomic and molecular distance from a novel graph-based neighborhood comparison. For every atom, we generate a topology vector encoding the molecular branching and connectivity of its local neighborhood. Specifically, we count the number of atoms at each bond distance from the starting atom (Figure B.2). This count can be computed efficiently using the Floyd-Warshall all-pairs distance algorithm Floyd (1962). The molecular topology vector is defined as the average of the constituent atomic fingerprints.

The atomic and molecular distances are defined as the weighted L_1 distance over these topology vectors. We take the difference of the atom counts at each bond distance and sum the weighted difference, with the weight, w , decreasing linearly with the bond distance, where

$w = (\text{maximum bond distance} - \text{atom pair distance})$. This ensures that differences in two atoms' local neighborhoods have a greater impact on their correspondence than remote differences. When mapped atoms are identified as the same element, we provide a bonus and reduce the distance score by the largest computed L_1 weight (*i.e.*, the maximum bond distance in the molecule), thereby improving the similarity. The molecular distance score is used in Ligand Extraction (Section B.2.1) and the atomic distance score is used in Ligand Atom Mapping (Section B.2.1).

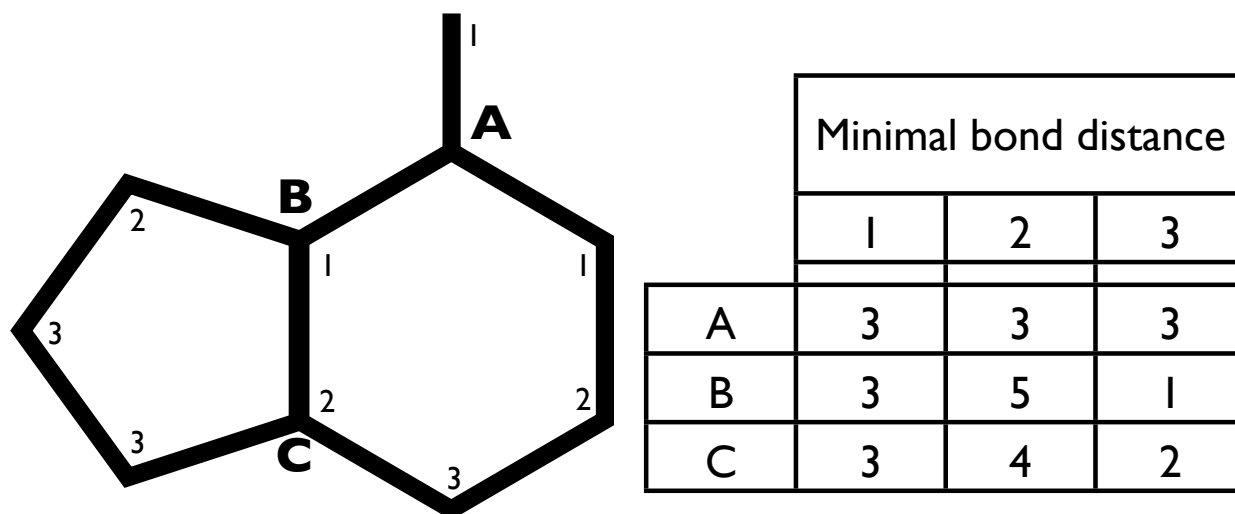


Figure B.2: Topological vectors encode atom counts as a function of bond distance from a selected atom. For example, the numbers next to the atoms depict the minimal bond distance from atom A. The topology vector for B is [3, 5, 1] indicating three atoms at distance 1, five atoms at distance 2, and one atom at distance 3.

Ligand atom mapping

Once the ligands have been extracted, a mapping or correspondence is computed between the atoms of the pivot ligand and the atoms of each non-pivot, or *query*, ligand. Ideally, one could determine a correspondence directly from the atom names in the PDB files. Unfortunately, the oftentimes arbitrary naming of equivalent atoms Bottoms and Xu (2008), Kleywegt (1999), Stockwell and Thornton (2006) and the desire to compare similar but not identical ligands (*e.g.*, analogs from SAR studies Koehler et al. (1997)) makes this approach problematic.

LigAlign computes an atomic correspondence using the neighborhood-based atomic topology vectors described in section B.2.1. We perform a weighted bipartite matching between the atoms in the query ligand and the atoms in the pivot ligand with edge weights equal to the atomic distance score. We solve for the minimal cost matching in polynomial time with the Kuhn-Munkres algorithm Munkres (1957). The minimal cost matching defines an atom-to-atom correspondence, or mapping, for the atoms in the query ligand to the atoms in the pivot ligand. The computed mappings are injective but may be partial when not all atoms of the query or pivot ligands have a corresponding atom in the other molecule.

This neighborhood-based scoring function can yield ties in symmetric molecular regions, *e.g.* the ortho and meta positions of six-membered para-substituted rings, which can produce multiple equally consistent correspondences. Although the neighborhood connectivity and the minimal-cost matchings are identical for symmetric regions, these alternative atom mappings will yield different ligand alignments. We break these ties by selecting the mapping that generates the smallest global RMSD after rigid alignment.

B.2.2 Ligand alignment

After the ligands have been selected and mapped in the ligand correspondence stage (Section B.2.1) we compute the transformations that yield the minimal RMSD between the bound ligands. We will discuss the flexible case in Section B.2.2. In the rigid case, given the atoms of the query, V_q , and pivot, V_p , and an atom-to-atom correspondence $m : V_q \mapsto V_p$ (such as the mapping defined in Section B.2.1), we define the RMSD of the rigid ligands as

$$\text{RMSD}(V_q, m) = \sqrt{\frac{\sum_{(v,u) \in \text{MappedAtoms}(V_q, m)} \text{distance}(v, u)^2}{|\text{MappedAtoms}(V_q, m)|}} \quad (\text{B.1})$$

$$\text{MappedAtoms}(V_q, m) = \{(v, u) \mid v \in V_q, u = m(v)\}. \quad (\text{B.2})$$

If the ligands are not identical, the mapping will be incomplete, *i.e.*, m will be undefined for some atoms in V_q . Therefore, this score is only computed over mapped atoms, as determined by Eq. (B.2). Given an atomic

correspondence, the transformation that minimizes the RMSD between two rigid ligands can be computed in closed form Kabsch (1978). Directly computing a minimal-RMSD ligand alignment, without an atomic mapping as computed in section B.2.1, is NP-hard Schneidman-Duhovny et al. (2008).

Fragmentation of flexible ligands

Our algorithm models molecular flexibility by splitting ligands into a set of rigid fragments connected by flexible hinges. Each rigid fragment is aligned independently and is not constrained by the alignment of the other fragments. The user may either specify a limit on the number of hinges or allow LigAlign to automatically determine the number of fragments. In the second case, LigAlign iteratively increases the number of hinges until the improvement in RMSD gained by the incorporation of additional hinges is below a threshold amount¹. Because the system lacks foreknowledge of the correct hinge placements, LigAlign performs a complete search of possible hinge placements. Similar to other flexible ligand analyses Schneidman-Duhovny et al. (2008), this search is exhaustive; however, dynamic programming and branch-and-bound techniques typically prune the search space efficiently.

A bond connecting atoms x and y decomposes V_q into two subsets: a rigid molecular fragment, F_{x,y,V_q} , and the possibly empty remainder of the molecule, R_{x,y,V_q} . We can efficiently compute the RMSD between F_{x,y,V_q} and the corresponding part of the pivot ligand using Eq. (B.1) because F_{x,y,V_q} is rigid. Because the F_{x,y,V_q} and R_{x,y,V_q} subsets may be of different sizes, the contributions to the achievable RMSD for the entire molecule must be weighted by the number of atoms in each piece. If we have fewer than the maximum number of allowed hinges, we attempt to further reduce the RMSD of R_{x,y,V_q} by splitting it into smaller pieces.

$$F_{x,y,V_q} = \{q \mid q \in V_q, q \text{ has a path to } x \text{ which does not include } y\} . \quad (\text{B.3})$$

$$R_{x,y,V_q} = V_q \setminus F_{x,y,V_q} . \quad (\text{B.4})$$

$$\text{FlexibleRMSD}(V_q, m, \text{bonds}, k) = \begin{cases} \text{RMSD}(V_q, m) & \text{if } k = 0 \\ \min_{(x,y) \in \text{bonds}} \sqrt{\frac{1}{|V_q|} \cdot \left(\text{RMSD}(F_{x,y,V_q}, m)^2 \cdot |F_{x,y,V_q}| + \text{FlexibleRMSD}(R_{x,y,V_q}, m, \text{bonds}, k-1)^2 \cdot |R_{x,y,V_q}| \right)} & \text{if } k \geq 1 . \end{cases} \quad (\text{B.5})$$

Equation (B.5) simultaneously determines which bonds should have hinges and computes the minimal RMSD of the two ligands after flexible alignment. The recursive description presented allows caching of partial solu-

¹The default threshold is 10%.

tions for molecular fragments and necessary intermediate computations such as RMSD scoring. Memoizing the best fragmentation for a particular molecular piece yields an exponential reduction in the search space, especially in long chain or branched molecules. In the extreme case of a linear molecule, this dynamic programming formulation reduces the runtime from exponential to polynomial, analogous to CKY parsing of context-free grammars Kozen (1997). Furthermore, the alternative extreme case, a ligand composed mainly of conjugated ring systems such as heme, is also handled quickly. As a consequence of Eq. (B.3), rings are treated as rigid fragments and no time is spent attempting to split them.

The recursive search for the best hinges in a molecular fragment has a worst-case time exponential in the number of hinges. Fortunately, the $\text{RMSD}(\cdot)$ function of Eq. (B.5) is non-negative and can be used to compute a lower bound on the FlexibleRMSD of the current fragmentation. This allows us to implement an efficient branch-and-bound search over possible fragmentations. The current implementation of LigAlign incorporates all of the algorithmic extensions described in this section.

B.2.3 Residue cluster extraction via clique detection

The result of the second stage (Section B.2.2) is a set of ligands split into rigid fragments and a set of transformations capable of aligning these fragments. The third stage of flexible alignment and analysis in LigAlign is the identification of common chemical or structural protein features. In the case of flexible ligand-based alignment, each rigid fragment and its accompanying protein is superimposed against the pivot ligand and the pivot protein. Common features can be found by identifying regions of the active site where several proteins have placed chemically similar amino acids. These residue clusters are identified and reported by LigAlign. The now-aligned proteins are examined for conserved residue clusters and these clusters are displayed through the PyMOL interface to the user. Although, for clarity, we restrict our discussion here to protein residues, LigAlign will report clusters of solvent molecules. An example is found in Section B.3.2.

For k proteins, residue cluster detection is an instance of k -Partite 3-D matching, an NP-hard problem Shatsky et al. (2006). Our algorithm for computing residue clusters, which completes quickly on typical test cases, is exact and complete, and therefore requires exponential time in the worst case. LigAlign computes the centroids of each residue in each protein, as in SPASM Kleywegt (1999), and creates a graph with a node corresponding to each centroid. Weighted edges connect nodes from one protein to nodes from the other proteins, where the edges have weight proportional to the Euclidian distance between the centroids. Edges with a distance above a specified threshold are removed (see next paragraph). Possible clusters are then determined by enumerating cliques in this graph. Cliques are ranked so as to maximize cardinality and, secondarily, minimize total edge weight. Any clique that is a proper subset of another clique is removed.

We extend this basic algorithm in three ways. First, the requirement that every protein must contribute to every cluster is often too restrictive. Therefore, LigAlign accepts a lower bound on the fraction of input proteins that must have a residue present in a cluster for the cluster to be reported. Second, as the diameter of the cluster increases, the biological significance of the cluster is likely to decrease. Our clustering algorithm takes a bound for the maximum acceptable cluster diameter to prevent reporting of biologically meaningless large-diameter clusters. We remove edges from the centroid-distance graph if the distance between the aligned residues is more than the user-specified maximum cluster diameter. This pruning ensures that the distance between every pair of clustered residues is smaller than the maximum diameter while simultaneously improving runtime. Although clusters that are proper subsets of another cluster are removed, it is possible that reported clusters can partially overlap. For example, given a set of core residues and several outliers, multiple alternative clusters may be reported with a different outlier in each cluster. Such boundary cases can occur regardless of the chosen maximum cluster diameter. Instead of arbitrarily selecting only one of the overlapping clusters, LigAlign returns all clustering alternatives.

Finally, the third extension allows the user to specify chemical constraints on the membership of residue clusters. Depending on the biological system, residues with different chemical properties may or may not be considered meaningful evidence of a conserved template. LigAlign includes three measures of residue similarity for filtering returned clusters. The least restrictive measure is geometric, which requires no chemical similarity. A second measure categorizes amino acids into chemical classes and requires that all residues in a cluster belong to the same group. We use the same seven non-exclusive chemical classes as Nebel: acidic (D, E), basic (R, H, K), amidic (N, Q), nonpolar (L, V, A, G, I, M, P), aromatic (F, W, Y), hydroxyl (S, R, Y), and sulphide bond forming (C) Nebel (2006). The third measure of similarity requires all residues in a cluster to be the same amino acid type. These requirements are applied during clique generation to prune prospective clusters and further speed cluster detection.

B.3 Results and discussion

We performed a series of experiments using LigAlign in both its rigid and flexible alignment modes to study the structures of two protein families. We first examined the cytochrome P450 active site and its rigid heme cofactor. As validation, we directly compared the structural motifs found by LigAlign to those previously described Nebel (2006). We then examined LigAlign's performance on the more difficult case of the NAD-binding proteins originally investigated by Carugo and Argos Carugo and Argos (1997). Although NAD is a flexible ligand, we first performed a rigid ligand-based protein alignment to allow comparison to previous work. We then performed a flexible ligand alignment on the NAD-binding protein family and compared the flexible results to those obtained

via rigid alignment.

B.3.1 Heme

We demonstrate LigAlign's ability to extract biologically significant residue motifs from the structure of human cytochrome P450 CYP17 by performing an alignment of the rigid heme moiety in PDB files 1BU7, 1H5Z, 1N97, 1PQ2, and 1W0G. We validate LigAlign's results by comparing the detected clusters to results previously reported by Nebel Nebel (2006). The patterns discussed by Nebel are the P450 conserved tetrapeptide G-x-[DEH]-T, the E-x-x-R and P-E-R-F motifs, and the P450 cysteine heme-iron ligand signature [FW]-[SGNH]-x-[GD]-{F}-[RKHPT]-{P}-C-[LIVM]. These patterns are highlighted with boxes and labeled 1, 2, 3, and 4, respectively, in Figure B.3. Pattern 4 is PROSITE database pattern PS00086 Hulo et al. (2008), and its conserved cysteine residue is marked with an arrow.

Figure B.3 shows our results for this system. The first five rows of Figure B.3 show a sequence alignment of 1BU7, 1H5Z, 1N97, 1PQ2, and 1W0G, as computed by MUSCLE Edgar (2004) using default settings. The residues are colored by MUSCLE according to the standard Clustal coloring rules. LigAlign's clusters are shown in the middle five rows. These are the residues selected from each protein by LigAlign via ligand alignment and cluster detection². The acceptable cluster diameter was set at 2.5Å to permit direct comparison with previously reported results, as described below. LigAlign pruned any clusters with less than 80% representation from the proteins (*i.e.*, fewer than four out of the five proteins considered). LigAlign was set to reject clusters of residues that were not of the same chemical class. This experiment completed in approximately 2 minutes on our test system, a 2.53GHz MacBook Pro laptop running OS X 10.6.2.

LigAlign detects clusters that match the entirety of the G-x-[DEH]-T and E-x-x-R motifs and the P450 cysteine heme-iron ligand signature. For patterns 1 and 2, LigAlign's reported patterns include the known motif. Ligalign reports G-x-E-T for pattern 1, additionally specifying a glutamic acid, and E-x-L-R for pattern 2, additionally specifying a leucine. These extensions to the known patterns reflect the conservation of the residues present in our five selected proteins. As seen by examining the first five rows of Figure B.3, this particular set of P450 homologous proteins shows consistent clusters of glutamic acid and leucine. A different set of input proteins would be required to discover patterns as permissive as the known motifs.

On pattern 3, LigAlign's performance matches Nebel's results, with both systems finding the proline and phenylalanine of the P-E-R-F motif. LigAlign does not detect the glutamic acid or arginine amino acids of the pattern. These residues turn out to be less spatially conserved than the proline and phenylalanine. The glutamic acid residues form a cluster with a diameter of 4.32Å and the arginine amino acids form a cluster with a diameter

²The simplicity of using LigAlign is demonstrated by the fact that this experiment required only two commands. Ligand alignment was performed with the command "ligalign 1pq2, 1w0g, 1bu7, 1h5z, 1n97". Clusters were computed and reported with the command "ligalign.find.template 2.5, 0.8, chemical, 1pq2, 1w0g, 1bu7, 1h5z, 1n97".

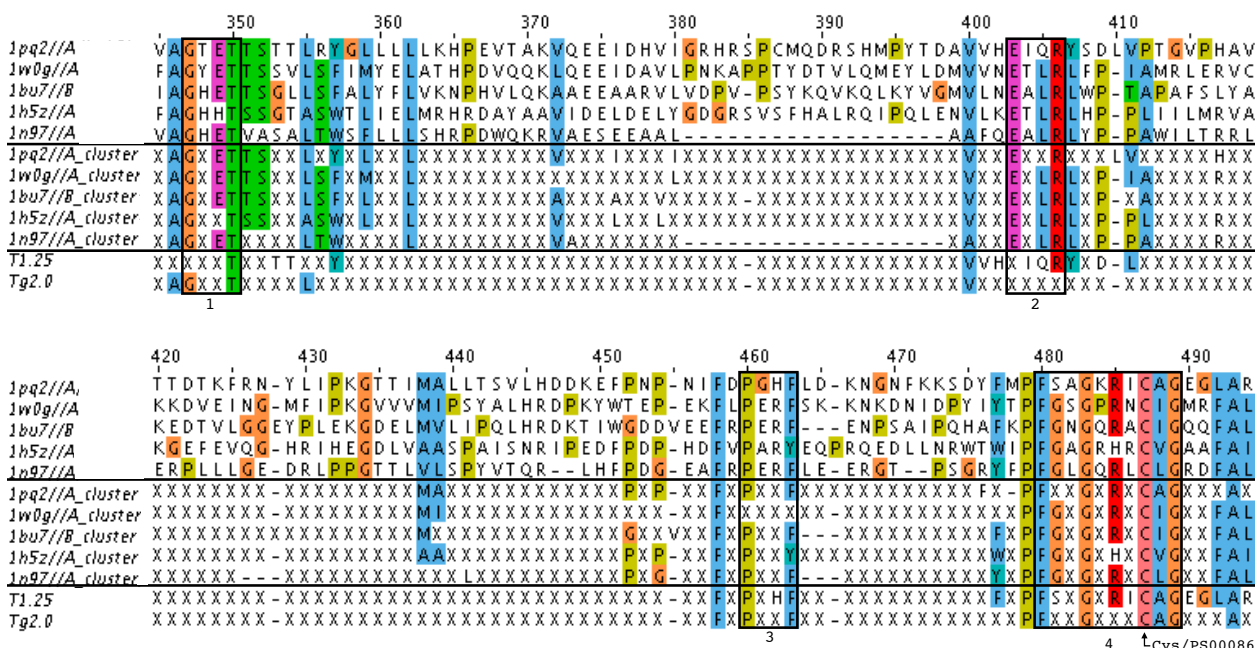


Figure B.3: A sequence alignment of P450 homologs and extracted patterns, as computed by MUSCLE Edgar (2004) using default settings. The colors correspond to the default Clustal coloring scheme. Four biologically-significant regions Nebel (2006) are boxed and numbered and the conserved cysteine residue from PROSITE pattern PS00086 is marked. The first five rows comprise the input proteins; the next five rows show the subset of residues marked as conserved by LigAlign; the last two rows are the conserved residues in the results reported by Nebel.

of 2.69Å. LigAlign detects these clusters only when run with a larger acceptable diameter. Under the set cutoff of 2.5Å, LigAlign correctly prunes both the glutamic acid and arginine residues of pattern 3.

We detect a number of residue clusters that fall outside of the four previously discussed patterns. These clusters are colored in light grey in Figure B.4. Although the residues are not contained in the four discussed patterns, examination of Figure B.4 shows that such clusters have good spatial and chemical agreement. These clusters are unsurprising, given that this set of proteins shows consistent sequence alignments outside of the four patterns (Figure B.3).

The last two rows of Figure B.3 are reproduced from Nebel's best template extraction results. T1.25 is determined from clusters formed from the atoms of the active site by merging the nearest atoms of the same elemental type, as long as these atoms are within 1.25Å. T1.25 lists all residues containing any such clustered atom. Our acceptable cluster diameter of 2.5Å was selected to correspond to T1.25, since LigAlign defines cluster size in terms of maximum diameter and Nebel's agglomerative clustering bound determines a radius. Tg2.0 reports the clusters Nebel generated from residue centroids, rather than atoms, with a distance bound of 2.0Å.

Nebel performs cluster detection on atoms rather than amino acids and, for sequence representation, chooses a single exemplar from the set of amino acids that contributed atoms to the cluster. LigAlign reports the residues of the original proteins, thereby avoiding the need to choose a single (possibly misleading) residue to represent

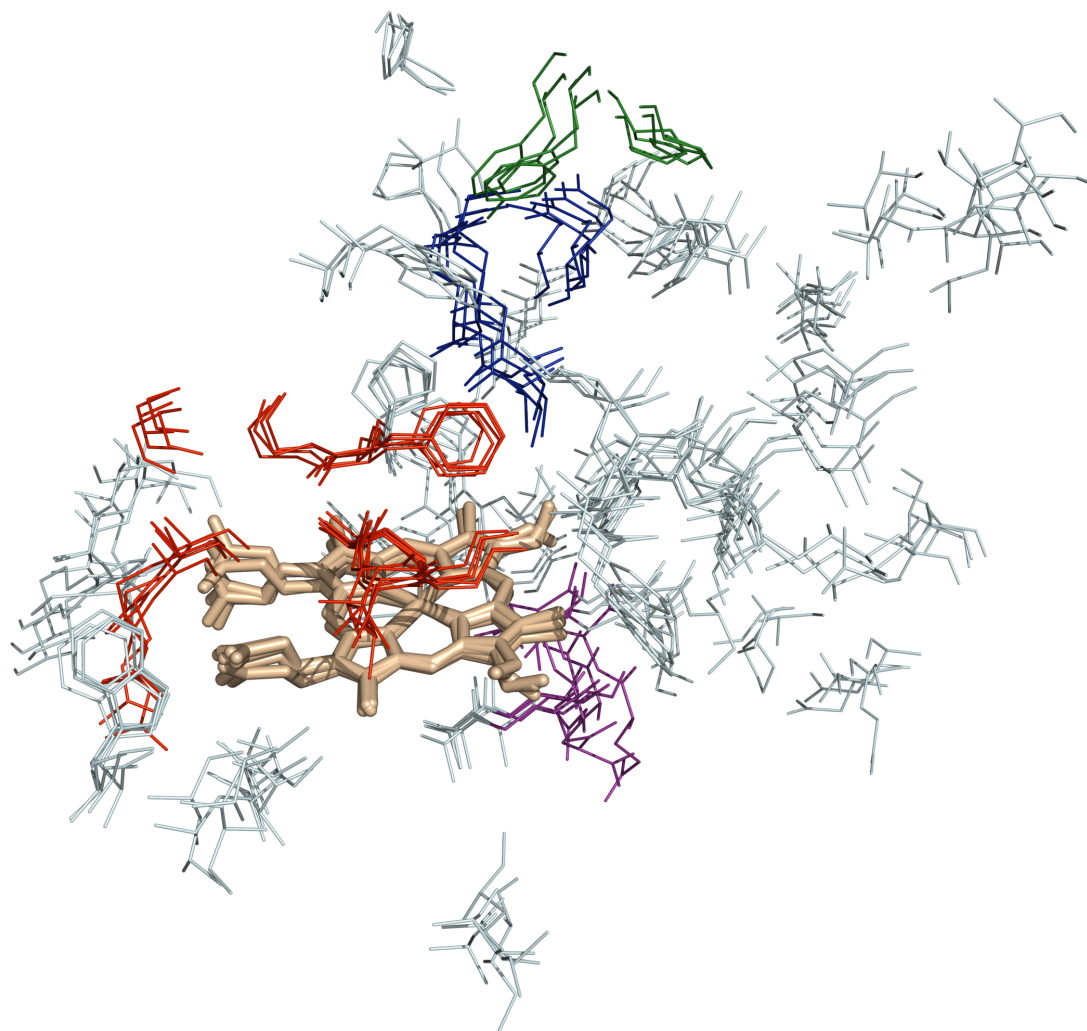


Figure B.4: A ligand-based alignment of P450 homologs showing the shared heme moiety (colored tan) surrounded by the residue clusters detected by LigAlign. The patterns labelled in Figure B.3 are distinguished by color. Pattern 1 is shown in purple; pattern 2 is dark blue; pattern 3 is green; and pattern 4 is presented in red. Clustered residues which are not members of one of these patterns are shown in light grey.

clusters that may contain several different amino acids. This explicit presentation of residues makes the variation within clusters clear and can be helpful in determining if clusters are missing representative residues due to variation in chemical class (such as the serine in pattern 4 of 1PQ2) or diverged spatial placement (as with pattern 3 of 1W0G).

B.3.2 Nicotinamide adenine dinucleotide

NAD is a ubiquitous protein ligand and is known to adopt multiple conformations Stockwell and Thornton (2006). To demonstrate the impact of flexible alignment, we compared the results of LigAlign’s flexible and rigid alignments of NAD. Before this comparison, we first validated the clusters reported under rigid alignment against previously known binding patterns. Carugo and Argos describe nicotinamide adenine dinucleotide binding motifs present in a wide variety of NAD-dependent proteins Carugo and Argos (1997). They perform a rigid minimal-RMSD alignment of 21 NAD cofactors and visually determine consistent patches of enzyme residues and solvent atoms that fall within 4.5Å of any NAD atom. These results are summarized in NAD-binding sequence patterns such as G-x-G-x-x-G or G-x-x-G-x-x-G Bottoms et al. (2002), where the ‘x’ represents any amino acid and reflects the variability exhibited in the residue sites.

Rigid NAD alignment

We use LigAlign to rigidly align the NAD ligand of the 21 proteins examined by Carugo and Argos, which are listed in Table B.1, and automatically extract clusters of conserved residues. Like Carugo and Argos, we specify 1HDX as the pivot ligand conformation. We choose a maximum cluster diameter of 3.0Å and prune clusters that lack residues from at least half of the proteins. We also keep clusters only when the residues are in the same chemical class. The ligand alignment and cluster detection for the 21 NAD-binding proteins required just under 24 minutes of runtime on our test system. The aligned NAD ligands and extracted clusters are shown in Figure B.5. The disorder in the alignment of the nicotine and nicotine ribose moieties is clearly visible, as noted by Carugo and Argos. Groups of water molecules, which are visible near the diphosphate and adenine moieties, are also detected by LigAlign. These are structurally conserved solvent molecules responsible for mediating hydrogen bonding Bottoms et al. (2002), Carugo and Argos (1997).

An alternative presentation of the extracted residue clusters is shown in Table B.1. Table B.1 lists 17 clusters comprising the residues extracted as conserved by LigAlign. Cluster numbers are assigned such that the residue positions in the pivot protein are non-decreasing. For each cluster, the amino acid type and sequence position of the participating residues are reported. These clusters are generated by post-processing the original clusters reported by LigAlign. Similar to Carugo and Argos, any cluster consisting of residues further than 4.5Å from the

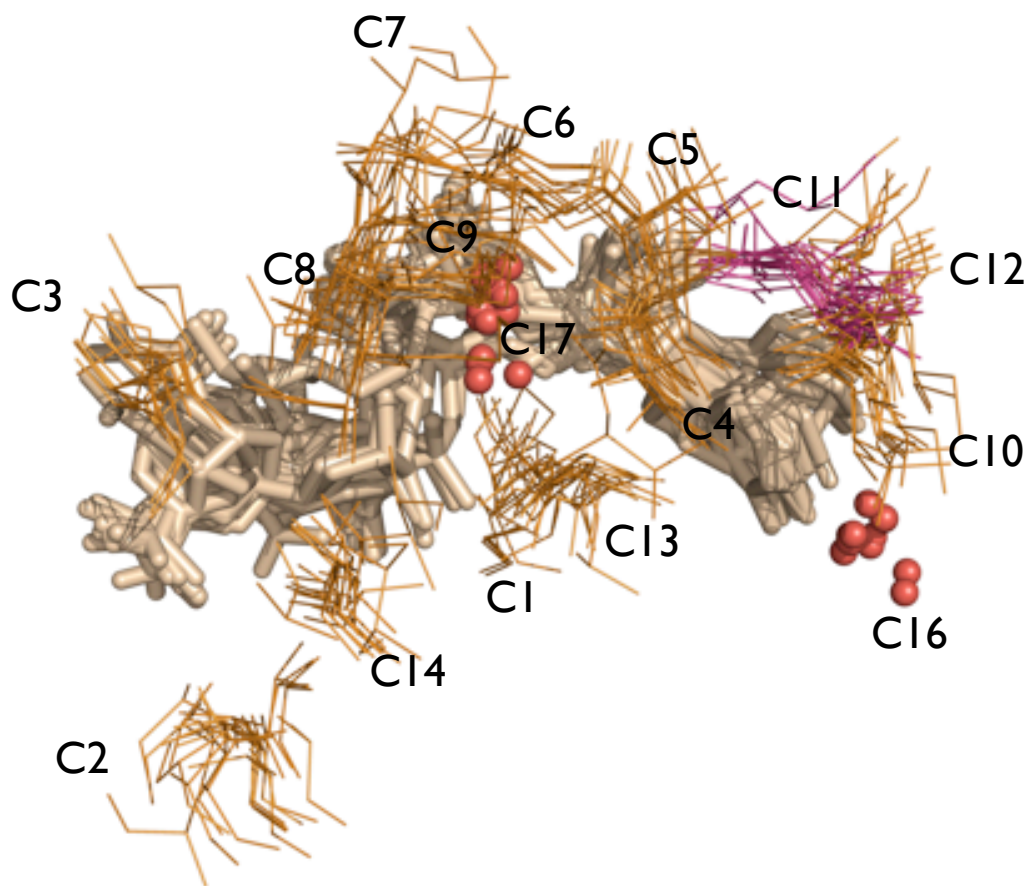


Figure B.5: A rigid minimal-RMSD ligand-based alignment of the NAD ligand from the 21 NAD-binding proteins listed in Table B.1. The shared NAD ligand is colored tan. Consistent clusters of residues are shown surrounding the ligand. Hydrophobic residues (LVAGIMP) are depicted as orange lines; acidic residues (DE) are shown as pink lines. Red spheres correspond to the location of structurally conserved water molecules.

PDB ID	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17
1HDX	-	-	G199	L200	G201	G202	V203	G204	V222	D223	I224	I269	-	V292	?377	-	?385
1BMD	A132	A245	G10	A11	G13	-	I15	G16	L40	E41	I42	G87	A88	V128	?334	-	?342
1DHR	A133	-	G13	G14	G16	A17	L18	-	I36	D37	V38	A83	G84	-	?241	?245	?248
1EMD	V121	M227	G7	A8	G10	G11	I12	G13	-	D34	I35	A77	G78	I117	?314	?317	?316
1GD1	A120	-	G7	-	G9	-	I11	G12	-	D32	L33	-	G97	-	?336	?380	?352
1GEU	-	-	G174	-	-	L203	-	-	-	E197	M198	A260	G262	I178	?452	-	-
1GGA	A134	-	G8	-	G10	-	I12	-	V36	D37	M38	-	G111	-	?361	-	-
1HDG	A120	-	G7	-	G9	-	I11	G12	-	D32	L33	-	G97	-	?336	?422	?369
1HDR	A136	-	G16	G17	G19	A20	L21	L21	V39	D40	V41	A86	G87	-	?244	?247	?249
1HLP	V140	P249	G27	A27A	G29	-	V31	G32	V51	D52	I53	A96	G97	-	?330	-	-
1LDM	V140	I249	G27	V28	G29	A30	V31	G32	V51	D52	V53	A96	G97	V136	?330	?373	-
1LDN	V140	I251	G27	A28	G29	-	V31	G32	I51	D52	A53	A96	G97	A136	?352	?364	?356
1LLD	V127	I240	G14	A15	G16	A17	V18	G19	-	D39	I40	A83	G84	I123	?320	-	?324
1PSD	V265	V112	G158	-	G160	-	I162	G163	-	D181	I182	V211	P212	A238	?450	-	?566
1LVL	P268	-	A263	-	G180	-	-	I182	-	-	-	V264	-	-	?460	-	?583
2HSD	-	I217	G13	G14	-	G17	L18	G19	A36	D37	V38	A88	G89	I137	?256	-	-
2NAD	V309	V150	A198	A199	G200	-	I202	G203	-	D221	-	-	P256	A283	?394	?578	?409
2NPX	-	-	G156	-	G158	L185	-	I162	-	D179	-	-	-	-	?818	-	?1147
2OHX	L309	-	G199	L200	G201	G202	V203	G204	V222	D223	I224	I269	-	V292	?403	?616	?477
4MDH	A132	A245	G10	A11	G13	-	I15	A16	L40	D41	I42	G87	-	V128	?335	?396	?362
9LDT	V142	I250	G28	V29	G30	A31	V32	G33	V52	D53	V54	A98	G99	V138	?401	?442	?424
Diameter	3.99	2.59	3.00	2.29	4.09	2.98	3.76	4.41	2.86	3.45	3.65	3.18	3.61	3.69	0.00	2.91	3.38
Carugo et al.	-	-	S1	S2,3	S4	S5	S6	S7	S8	S9	S10	S11,15	S12,16	-	-	-	-

Table B.1: Merged clusters as detected after rigid superposition of bound NAD ligands. Diameter is the maximum pairwise distance of residue centroids in a cluster in Angstroms. Residues of type ‘?’ are either water or small-molecule ligands. Entries in grey are further than 4.5Å from any of their protein’s NAD atoms. The last row shows the correspondence of LigAlign’s clusters with the conserved interacting positions reported by Carugo and Argos Carugo and Argos (1997). Cluster 15 has a representative from each protein and a cluster diameter of 0Å because it is the bound NAD ligand. Cluster 16 is a water molecule also reported by Carugo and Argos (See Fig 11 Carugo and Argos (1997), *ibid.*). Cluster 17 corresponds to a structurally-conserved water molecule described by Bottoms et al. Bottoms et al. (2002).

aligned ligands is discarded. When a cluster lies on the boundary, with some residues further than 4.5Å, Carugo and Argos report the distant residues; we highlight such residues in grey in Table B.1.

As discussed in section B.2.3, LigAlign reports alternative clusterings of outlier residues which, when visualized, produce the appearance of a single larger cluster. Merging the overlapping clusters produces a list of clusters that more closely matches the presentation shown in Figure B.5. In the case when overlapping clusters cannot be unambiguously merged (*i.e.*, when one protein contributes different amino acids to each of the different clusters) we report the cluster of larger cardinality, breaking ties to favor the cluster with smaller diameter. The diameters of the merged clusters are listed in Angstroms in the second to last row of Table B.1, and the original clusters may be inspected in Tables 1 through 22 of the supplementary material.

The final row of Table B.1 shows the correspondence between LigAlign’s clusters and the sequence positions reported by Carugo and Argos. Carugo and Argos manually identify 17 residue sites, labelled S1 through S17, as interacting with NAD. The clusters reported by LigAlign include interacting positions manually identified by Carugo and Argos. In particular, columns C5, C8, and C13 of Table B.1 correspond to the glycines in the binding motif of G-x-G-x-x-G. Carugo and Argos’ positions S13, S14, and S17 were not identified by LigAlign as these

clusters contained residues from fewer than half of the proteins considered.

LigAlign also reports several clusters, C1, C2, and C14, which were not described by Carugo and Argos. Although we cannot use Carugo and Argos's results to validate these clusters, Figure B.5 shows they are as spatially and chemically consistent as the other detected clusters.

Flexible NAD alignment

The rigid alignment results of Section B.3.2 are consistent with those previously reported in the literature. Of course, a rigid alignment of a flexible ligand, such as NAD, has inherent limitations. Having verified our clustering methodology for rigid molecules, we next used LigAlign to perform a fragment-based flexible alignment of NAD.

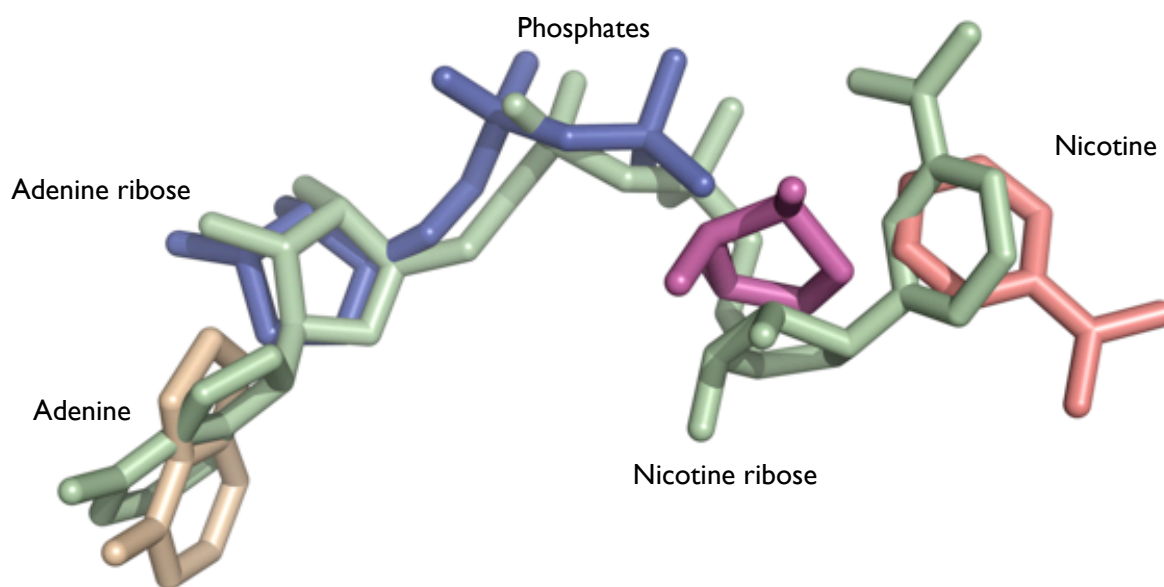


Figure B.6: Fragmentation of the bound NAD ligand from 1HDR (each rigid fragment is shown in a different color), after fragmentation but before alignment to the green pivot NAD ligand in 1HDX. The NAD moieties of nicotine, nicotine ribose, nicotine phosphate, adenine phosphate, adenine ribose, and adenine are labeled.

As in the rigid case, we took 1HDX as the pivot conformation, used a maximum cluster diameter of 3.0\AA , and pruned clusters of disparate chemical type or with residue representation from fewer than half of proteins. We used the default minimum fragment size of 6 heavy atoms. Rather than specify a number of fragments for partitioning NAD, we used LigAlign's default behavior to iteratively increase the number of hinges inserted until the incremental RMSD improvement from additional hinges was less than 10%. Given the set parameters, LigAlign automatically determines a fragmentation that yields a minimal RMSD alignment against the pivot ligand conformation. The fragmentation for the NAD ligand of 1HDR is shown in Figure B.6. In some cases, such as 9LDT or 1LDN, the bound conformation of the ligand is sufficiently similar to the conformation of NAD in 1HDX, that the ligand is not fragmented. In this situation, alignment proceeds identically to the rigid case. In

PDB ID	C1	C2	C3	C4	C5	C6
1HDX	G199	G201	V203	I269	–	?385
1BMD_1	G10	G13	I15	G87	H186	?342
1DHR_3	–	A149	–	G84	–	–
1EMD_4	I97	–	I75	–	–	–
1GD1_3	–	–	–	–	R10	–
1GEU_1	–	–	–	–	–	?478
1GGA_4	–	–	–	–	R11	–
1HDG_3	–	–	–	–	R10	–
1HDR_1	–	A152	–	A86	–	–
1HLP_2	–	A34	A250	–	–	–
1LDM_1	G27	G29	V31	A96	H193	–
1LDN_1	G27	G29	V31	A96	H193	?356
1LLD_1	G14	G16	V18	A83	H180	?324
1PSD_2	G158	G160	I162	–	H292	?566
1LVL_4	–	A313	–	–	–	–
2HSD_3	V107	–	–	G89	–	–
2NAD_2	A198	G200	I202	–	H332	?409
2NPX_2	–	–	–	–	–	?900
2OHX_1	G199	G201	V203	I269	–	?477
4MDH_1	G10	G13	I15	G87	H186	?362
9LDT_1	G28	G30	V32	A98	H195	?424
Diameter	2.96	3.55	3.01	2.93	2.90	2.73

Table B.2: Merged clusters (see discussion in section B.2.3 and B.3.2) as detected after rigid superposition of fragments containing the nicotine moiety. Entries in grey are further than 4.5Å from any of their protein's nicotine fragment's atoms. Residues of type '?' in cluster C6 are conserved waters.

the proteins under consideration, the largest number of fragments required is 4. Each of these fragments can be independently aligned to the pivot. The fragment alignments induce a fragment-based superposition of the bound proteins, which can be searched for clusters of amino acids.

Once we have fragmented the ligands, we generate a protein alignment for each moiety by superimposing the fragment containing the moiety under consideration against the pivot. The consistent residues around each moiety can be determined by examining each alignment in turn. Among common natural ligands, NAD is particularly flexible and contains six distinct moieties (nicotine, nicotine ribose, nicotine phosphate, adenine phosphate, adenine ribose, and adenine). The large number of rigid fragments creates a significant amount of clustering data. Tables 5 through 22 in the supplementary material list the overlapping clusters discovered when the superposition of each moiety is independently considered. As expected, the fragment-based alignments of the adenine, adenine ribose, and phosphate moieties find clusters consistent with those discovered through rigid alignment, since a rigid alignment achieves a close superposition of these moieties Carugo and Argos (1997). A fragment-based alignment of the nicotine ribose also finds clusters consistent with the rigid alignment.

In the case of the nicotine fragment, however, our flexible results diverge from the rigid analysis. Carugo and Argos note a high variability in the orientation of this moiety and the shape of the surrounding active site pocket. Using rigid alignment, they find no consistent positions near the nicotine. Using LigAlign's fragment-based

alignment, we can avoid the disorder induced by rigid ligand alignment on the position and orientation of the nicotine moiety. After flexible alignment, LigAlign suggests a new structurally conserved group corresponding to cluster C5 in Table B.2, shown in Figure B.7. The amino acids in cluster C5 are close enough to be considered part of the active site and Carugo and Argos group the arginine residues with their S5 position. However, when the nictines are precisely aligned, a structurally and chemically consistent cluster of basic residues becomes apparent.

As demonstrated by NAD, a rigid alignment of a complete ligand gives no guarantee that constituent ligand moieties will be meaningfully aligned. In contrast, flexible ligand-based alignment does not suffer from this disadvantage. By separately aligning each moiety, additional conserved structural relationships can be identified.

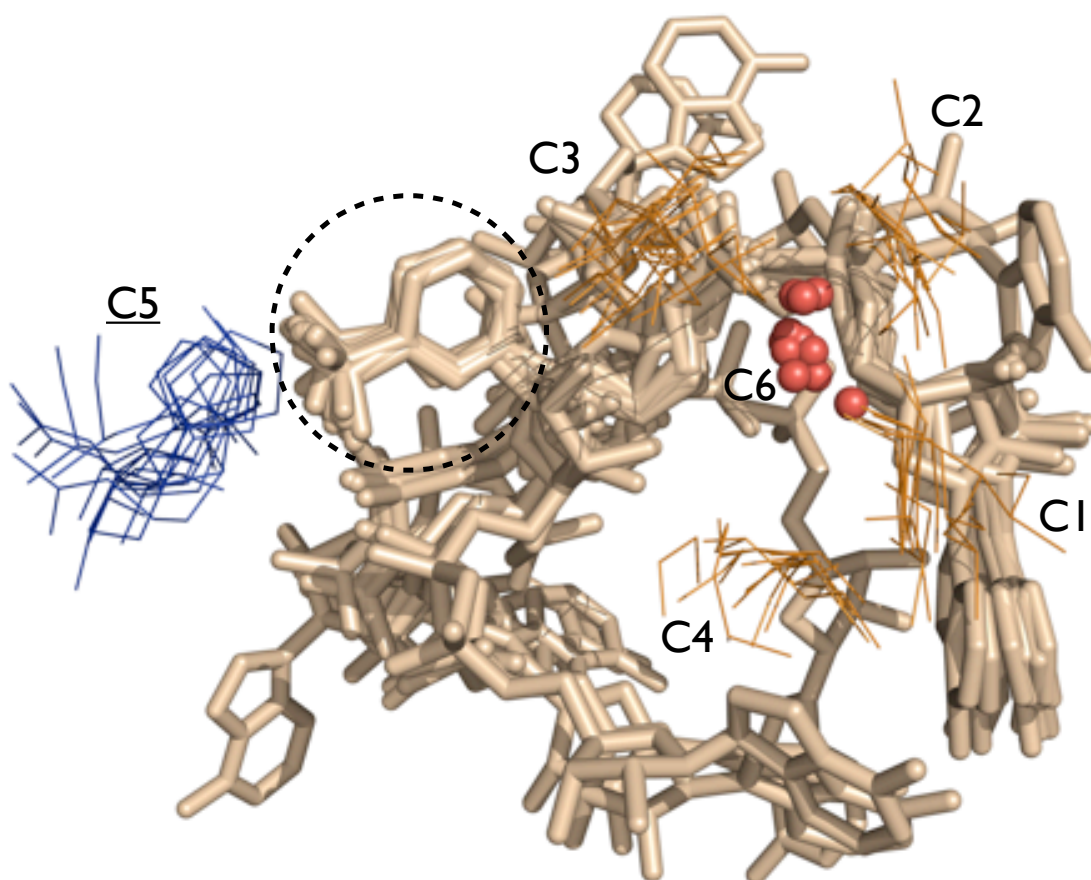


Figure B.7: A protein alignment using only the nicotine moiety of the NAD ligand (shown in a dashed circle). The shared NAD ligand is colored tan. The misalignment induced by nicotine alignment on the non-nicotine moieties is apparent. Consistent clusters of residues, as listed in Table B.2, are shown surrounding the ligand. The underlined C5 cluster was identified by flexible alignment but was not detected by rigid alignment. Hydrophobic residues (LVAGIMP) are depicted as orange lines; basic residues (RHK) are blue lines. Red spheres correspond to the location of structurally conserved water molecules.

B.4 Chapter Conclusion and Future Work

The structural analysis of protein families relies on the ability to accurately align corresponding regions of individual structures. Ligand-based protein alignment has emerged as one solution for this alignment problem. Unfortunately, while state-of-the-art methods succeed for rigid ligands, their performance falls short for ligands which bind in multiple conformations. For flexible ligands, a rigid ligand-based protein alignment can induce an apparent disorder in the residues surrounding each ligand moiety causing clustering to fail.

In this paper, we presented LigAlign, a new software system for the structural analysis of protein active sites via ligand-based protein alignment. When performing rigid alignments, LigAlign produces automatically generated results consistent with manually annotated, biologically relevant structural motifs. When performing flexible alignments, LigAlign automatically produces biochemically reasonable ligand fragmentations and identifies conserved structural motifs that are not detected by the rigid alignment. This scenario was demonstrated for the flexible ligand NAD. The adoption of LigAlign as a tool for structural biologists could uncover similar examples of non-obvious amino acid structural conservation.

Several subproblems of flexible ligand-based protein alignment (such as ligand correspondence, unmapped ligand alignment, and cluster detection) are NP-hard and therefore likely to have a worst case exponential runtime. In LigAlign, a series of optimizations including branch-and-bound search, dynamic programming, and memoization allow the typical execution to complete quickly. Looking forward, several improvements and design alternatives may be possible. For example, LigAlign currently computes ligand correspondences using a maximal bipartite matching based on neighborhood similarity. While this technique is robust to structural and naming discrepancies, it is not guaranteed to properly map ligands that share only small fragments. A ligand mapping technique based on maximal common substructure detection could address this limitation. Unfortunately, maximal common substructure detection is NP-complete and it is unknown if new molecularly optimized algorithms Cao et al. (2008) will complete sufficiently quickly on large ligands, such as heme.

It may also be possible to improve the clustering step. LigAlign finds clusters of residue and solvent centroids based on geometric proximity and filtered by chemical similarity. These tests are currently binary, where a residue will be either accepted into a cluster or rejected as inconsistent. An alternative would be to use a soft assignment of residue membership to clusters, producing a real-valued consistency score. For example, clusters could be judged based on BLOSUM scores Kleywegt (1999) producing a non-binary measure of cluster acceptability.

We believe LigAlign is a useful tool for the structural biology community and we encourage structural biologists to use it. LigAlign runs under the PyMOL molecular viewing program. This integration allows the user to easily generate and interact with the types of figures found in this manuscript. LigAlign is supported on Apple OS X, Linux, and MS Windows. The source code is freely available under the GNU LGPL. LigAlign is available

for download at <http://compbio.cs.toronto.edu/lalign>

Bibliography

K.K. Agarwal, T.D.L. Larsen, and H.L. Gelernter. Application of chemical transforms in SYNCHEM2, a computer program for organic synthesis route discovery. *Comp. & Chemistry*, 2(2):75–84, 1978. ISSN 0097-8485. doi: DOI:10.1016/0097-8485(78)87005-3. URL <http://www.sciencedirect.com/science/article/B6TFV-44W5DTW-K0/2/acdf20f7b805b27335df83408703a4a0>.

L.V. Allis, M. van der Meulen, and H.J. van den Herik. Proof-number search. *Artificial Intelligence*, 66:91–124, 1994.

Tharun Kumar Allu and Tudor I. Oprea. Rapid evaluation of synthetic and molecular complexity for in silico chemistry. *Journal of Chemical Information and Modeling*, 45(5):1237–1243, 2005. doi: 10.1021/ci0501387. URL <http://pubs.acs.org/doi/abs/10.1021/ci0501387>.

J.C. Baber and Miklos Feher. Predicting synthetic accessibility: Application in drug discovery and development. *Mini-reviews in Medicinal Chemistry*, 4:681–692, 2004.

John M. Barnard and P. Matthew Wright. Towards in-house searching of markush structures from patents. *World Patent Information*, 31(2):97 – 103, 2009. ISSN 0172-2190. doi: DOI:10.1016/j.wpi.2008.09.012. URL <http://www.sciencedirect.com/science/article/pii/S0172219008001385>.

René Barone and Michel Chanon. A new and simple approach to chemical complexity. application to the synthesis of natural products. *J. Chem. Inf. Comput. Sci.*, 41:269–272, 2001.

Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Res*, 28:235–242, 2000.

Steven H. Bertz. The first general index of molecular complexity. *J. Am. Chem. Soc.*, 103:3599–3601, 1981.

Steven H. Bertz. On the complexity of graphs and molecules. *Bulletin of Mathematical Biology*, 45(5):849–855, 1983.

- Edward S. Blurock. Computer-aided synthesis design at RISC-Linz: automatic extraction and use of reaction classes. *Journal of Chemical Information and Computer Sciences*, 30(4):505–510, 1990.
- Krisztina Boda and A. Peter Johnson. Molecular complexity analysis of de novo designed ligands. *J. Med. Chem.*, 49:5869–5879, 2006.
- Krisztina Boda, Thomas Seidel, and Johann Gasteiger. Structure and reaction based evaluation of synthetic accessibility. *J. Comput. Aided Mol. Des.*, 21:311–325, 2007.
- Blai Bonet and Hector Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. In Enrico Giunchiglia, Nicola Muscettola, and Dana S. Nau, editors, *ICAPS*, pages 12–31. AAAI, 2003. ISBN 1-57735-187-8.
- Blai Bonet and Hector Geffner. An algorithm better than AO*? In Manuela M. Veloso and Subbarao Kambhampati, editors, *AAAI*, pages 1343–1348. AAAI Press / The MIT Press, 2005. ISBN 1-57735-236-X.
- Blai Bonet and Hector Geffner. Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings, and its application to MDPs. In *ICAPS*, pages 142–151. AAAI, 2006.
- Christopher A Bottoms and Dong Xu. Wanted: unique names for unique atom positions. PDB-wide analysis of diastereotopic atom names of small molecules containing diphosphate. *BMC Bioinformatics*, 9 Suppl 9:S16, 2008. ISSN 1471-2105 (Electronic); 1471-2105 (Linking). doi: 10.1186/1471-2105-9-S9-S16.
- Christopher A Bottoms, Paul E Smith, and John J Tanner. A structurally conserved water molecule in Rossmann dinucleotide-binding domains. *Protein Sci*, 11(9):2125–2137, Sep 2002. ISSN 0961-8368 (Print); 0961-8368 (Linking). doi: 10.1110/ps.0213502.
- Philip L. Brower, Donald E. Butler, Carl F. Deering, Tung V. Le, Alan Millar, Thomas N. Nanninga, and Bruce D. Roth. The synthesis of (4R-cis)-1,1-dimethylethyl 6-cyanomethyl-2,2-dimethyl-1,3-dioxane-4-acetate, a key intermediate for the preparation of CI-981, a highly potent, tissue selective inhibitor of HMG-CoA reductase. *Tetrahedron Letters*, 33(17):2279–2282, 1992. ISSN 0040-4039. doi: DOI:10.1016/S0040-4039(00)74189-X. URL <http://www.sciencedirect.com/science/article/B6THS-42HGSX2-6N/2/bc8c8a2735d89334846d94fb8e96820a>.
- Daniel Bryce and Subbarao Kambhampati. A tutorial on planning graph based reachability heuristics. *AI Magazine*, 28(1):47–83, 2007.
- L. Michael Cacace. Fortune Global 500 Rankings. In *Fortune*. <http://money.cnn.com/magazines/fortune/global500/2009/industries/>, July 2009.

- R. S. Cahn, Christopher Ingold, and V. Prelog. Specification of Molecular Chirality. *Angewandte Chemie International Edition in English*, 5(4):385–415, April 1966. ISSN 0570-0833. doi: 10.1002/anie.196603851. URL <http://dx.doi.org/10.1002/anie.196603851>.
- M. Campbell. Knowledge Discovery in DEEP BLUE. *Communications of the ACM*, 42(11):65–67, 1999.
- Yiqun Cao, Tao Jiang, and Thomas Girke. A maximum common substructure-based algorithm for searching and predicting drug-like compounds. *Bioinformatics*, 24(13):i366–74, Jul 2008. ISSN 1367-4811 (Electronic); 1367-4811 (Linking). doi: 10.1093/bioinformatics/btn186.
- John S. Carey, David Laffan, Colin Thomson, and Mike T. Williams. Analysis of the reactions used for the preparation of drug candidate molecules. *Org. Biomol. Chem.*, 4:2337–2347, 2006. doi: 10.1039/B602413K. URL <http://dx.doi.org/10.1039/B602413K>.
- Oliviero Carugo and Patrick Argos. NADP-dependent enzymes. I: Conserved stereochemistry of cofactor binding. *Proteins*, 28(1):10–28, May 1997. ISSN 0887-3585 (Print); 0887-3585 (Linking).
- ChemAxon. JChem 5.7.1. <http://www.chemaxon.com>, 2011.
- Chao Chen, Andy Liaw, and Leo Breiman. Using random forest to learn imbalanced data. Technical Report 666, Department of Statistics, UC Berkeley, July 2004.
- Ying Chen, Scott Spangler, Jeffrey Kreulen, Stephen Boyer, Thomas D. Griffin, Alfredo Alba, Amit Behal, Bin He, Linda Kato, Ana Lelescu, Cheryl Kieliszewski, Xian Wu, and Li Zhang. Simple: A strategic information mining platform for licensing and execution. *Data Mining Workshops, International Conference on*, 0:270–275, 2009. doi: <http://doi.ieeecomputersociety.org/10.1109/ICDMW.2009.36>.
- Clara D. Christ, Matthias Zentgraf, and Jan M. Kriegl. Mining electronic laboratory notebooks: Analysis, retrosynthesis, and reaction based enumeration. *Journal of Chemical Information and Modeling*, 52(7):1745–1756, 2012. doi: 10.1021/ci300116p. URL <http://pubs.acs.org/doi/abs/10.1021/ci300116p>.
- Jonathan Clayden, Nick Greeves, Stuart Warren, and Peter Wothers. *Organic Chemistry*. Oxford University Press, 2000.
- Anthony Cook, A. Peter Johnson, James Law, Mahdi Mirzazadeh, Orr Ravitz, and Aniko Simon. Computer-aided synthesis design: 40 years on. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2(1):79–107, 2012. ISSN 1759-0884. doi: 10.1002/wcms.61. URL <http://dx.doi.org/10.1002/wcms.61>.
- E. J. Corey. General methods for the construction of complex molecules. *Pure Appl. Chem.*, 14(1):19–38, 1967.

- E. J. Corey and W. Todd Wipke. Computer-Assisted Design of Complex Organic Syntheses. *Science*, 166(3902): 178–192, 1969. doi: 10.1126/science.166.3902.178. URL <http://www.sciencemag.org/content/166/3902/178.short>.
- E. J. Corey, Alan K. Long, Theodora W. Greene, and John W. Miller. Computer-assisted synthetic analysis. selection of protective groups for multistep organic syntheses. *The Journal of Organic Chemistry*, 50(11): 1920–1927, 1985. doi: 10.1021/jo00211a027. URL <http://pubs.acs.org/doi/abs/10.1021/jo00211a027>.
- Elias J. Corey. The Logic of Chemical Synthesis: Multistep Synthesis of Complex Carbogenic Molecules (Nobel Lecture). *Angewandte Chemie International Edition in English*, 30(5):455–465, December 1991. ISSN 1521-3773. doi: 10.1002/anie.199104553. URL <http://dx.doi.org/10.1002/anie.199104553>.
- Elias J. Corey and Xue-Min Cheng. *The Logic of Chemical Synthesis*. Wiley, 1995.
- Daylight. Daylight Chemical Information Systems, Inc. http://www.daylight.com/daycgi_tutorials/smirks_examples.cgi, 2008.
- Paula de Matos, Rafael Alcántara, Adriano Dekker, Marcus Ennis, Janna Hastings, Kenneth Haug, Inmaculada Spiteri, Steve Turner, and Christoph Steinbeck. Chemical entities of biological interest: an update. *Nucleic Acids Research*, 38(suppl 1):D249–D254, 2010.
- Kirill Degtyarenko, Paula de Matos, Marcus Ennis, Janna Hastings, Martin Zbinden, Alan McNaught, Rafael Alcántara, Michael Darsow, Mickaël Guedj, and Michael Ashburner. Chebi: a database and ontology for chemical entities of biological interest. *Nucleic Acids Research*, 36(suppl 1):D344–D350, 2008. doi: 10.1093/nar/gkm791. URL http://nar.oxfordjournals.org/content/36/suppl_1/D344.abstract.
- W.L. DeLano. The PyMOL Molecular Graphics System. <http://www.pymol.org>, 2002.
- K A Denessiouk, J V Lehtonen, and M S Johnson. Enzyme-monomonucleotide interactions: three different folds share common structural elements for ATP recognition. *Protein Sci*, 7(8):1768–1771, Aug 1998. ISSN 0961-8368 (Print); 0961-8368 (Linking). doi: 10.1002/pro.5560070811.
- K A Denessiouk, V V Rantanen, and M S Johnson. Adenine recognition: a motif present in ATP-, CoA-, NAD-, NADP-, and FAD-dependent proteins. *Proteins*, 44(3):282–291, Aug 2001. ISSN 0887-3585 (Print); 0887-3585 (Linking).
- Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997. ISSN 0885-6125. doi: 10.1023/A:1007413511361. URL <http://dx.doi.org/10.1023/A%3A1007413511361>.

- Robert C Edgar. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, 5:113, Aug 2004. ISSN 1471-2105 (Electronic); 1471-2105 (Linking). doi: 10.1186/1471-2105-5-113.
- J. Eiblmaier, H. Kraut, H. Saller, H. Matuszczyk, , and P. Loew. Reaction classification, an enduring success story. In *50 Years of Computers in Organic Chemistry: Symposium in Honor of James B. Hendrickson*, National Meeting Anaheim Spring 2011. American Chemical Society, 2011.
- Jon A Erickson, Mehran Jalaie, Daniel H Robertson, Richard A Lewis, and Michal Vieth. Lessons in molecular recognition: the effects of ligand and protein flexibility on molecular docking accuracy. *J Med Chem*, 47(1): 45–55, Jan 2004. ISSN 0022-2623 (Print); 0022-2623 (Linking). doi: 10.1021/jm030209y.
- Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Cheminformatics*, 1(8), 2009.
- Richard Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artif. Intell.*, 2(3/4):189–208, 1971.
- Igor V. Filippov and Marc C. Nicklaus. Optical structure recognition software to recover chemical information: Osra, an open source solution. *Journal of Chemical Information and Modeling*, 49(3):740–743, 2009. doi: 10.1021/ci800067r. URL <http://pubs.acs.org/doi/abs/10.1021/ci800067r>.
- Paul Fleming, Geraldine C. B. Harriman, Zhan Shi, and Shaowu Chen. Ccr9 inhibitors and methods of use thereof. *U.S. Patent Trademark Office*, US 7,238,717, 2007.
- Robert W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, 1962. ISSN 0001-0782. doi: <http://doi.acm.org/10.1145/367766.368168>.
- Jeff Forcier, Paul Bissex, and Wesley Chun. *Python Web Development with Django*. Addison-Wesley Professional, 1 edition, 2008. ISBN 0132356139, 9780132356138.
- Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990. ISBN 0716710455.
- H.L. Gelernter, A. F. Sanders, D. L. Larsen, K.K. Agarwal, R. H. Boivie, G. A. Spritzer, and J. E. Searleman. Empirical explorations of SYNCHEM. *Science*, 197(4308):1041–1049, September 1977.
- Michael R. Genesereth, Nathaniel Love, and Barney Pell. General game playing: Overview of the AAAI competition. *AI Magazine*, 26(2):62–72, 2005. URL <http://games.stanford.edu/competition/misc/aaai.pdf>.

GGASoftware. Indigo 1.1.8. <http://ggasoftware.com>, 2012.

Thomas D. Griffin, Stephen K. Boyer, and Isaac G. Council. Annotating patents with medline mesh codes via citation mapping. In Hamid R. Arabnia, editor, *Advances in Computational Biology*, volume 680 of *Advances in Experimental Medicine and Biology*, pages 737–744. Springer New York, 2011. ISBN 978-1-4419-5913-3. URL http://dx.doi.org/10.1007/978-1-4419-5913-3_82. 10.1007/978-1-4419-5913-3_82.

Rajarshi Guha, Michael T. Howard, Geoffrey R. Hutchison, Peter Murray-Rust, Henry Rzepa, Christoph Steinbeck, Jörg Wegner, and Egon L. Willighagen. The blue obelisk – interoperability in chemical informatics. *Journal of Chemical Information and Modeling*, 46(3):991–998, 2006. doi: 10.1021/ci050400b. URL <http://pubs.acs.org/doi/abs/10.1021/ci050400b>.

Eric A. Hansen and Shlomo Zilberstein. LAO*: A heuristic search algorithm that finds solutions with loops. *Artif. Intell.*, 129(1-2):35–62, 2001.

P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2):100–107, 1968.

Kazunari Hattori, Hiroaki Wakabayashi, and Kenta Tamaki. Predicting key example compounds in competitors’ patent applications using structural information alone. *Journal of Chemical Information and Modeling*, 48(1):135–142, 2008. doi: 10.1021/ci7002686. URL <http://pubs.acs.org/doi/abs/10.1021/ci7002686>. PMID: 18177028.

Abraham Heifets and Igor Jurisica. Diversity as a first-class ranking measure in automated bioisosteric replacement. In *Abstracts of Papers, 242nd ACS National Meeting & Exposition, Denver, Colorado, United States*, August 28-September 1 2011.

Abraham Heifets and Igor Jurisica. Construction of new medicines via game proof search. In Jörg Hoffmann and Bart Selman, editors, *AAAI*. AAAI Press, 2012a.

Abraham Heifets and Igor Jurisica. Scripdb: a portal for easy access to syntheses, chemicals and reactions in patents. *Nucleic Acids Research*, 40(Database-Issue):428–433, 2012b.

Abraham Heifets and Ryan H. Lilien. Lalign: Flexible ligand-based active site alignment and analysis. *Journal of Molecular Graphics and Modelling*, 29(1):93 – 101, 2010. ISSN 1093-3263. doi: 10.1016/j.jm gm.2010.05.005. URL <http://www.sciencedirect.com/science/article/pii/S1093326310000756>.

Malte Helmert, Minh Do, and Ioannis Refanidis. International planning competition. <http://ipc.informatik.uni-freiburg.de/Results>, 2008.

- James B. Hendrickson, Ping Huang, and A. Glenn Toczko. Molecular complexity: a simplified formula adapted to individual atoms. *Journal of Chemical Information and Computer Sciences*, 27(2):63–67, 1987. doi: 10.1021/ci00054a004. URL <http://pubs.acs.org/doi/abs/10.1021/ci00054a004>.
- William Hersh and Ellen Voorhees. Trec genomics special issue overview. *Information Retrieval*, 12:1–15, 2009. ISSN 1386-4564. URL <http://dx.doi.org/10.1007/s10791-008-9076-6>. 10.1007/s10791-008-9076-6.
- Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res. (JAIR)*, 14(253-302), 2001.
- L.S. Homem de Mello and A.C. Sanderson. AND/OR graph representation of assembly plans. *IEEE Transactions on Robotics and Automation*, 6(2):188–199, Apr 1990. ISSN 1042-296X. doi: 10.1109/70.54734.
- Qi Huang, Lin-Li Li, and Sheng-Yong Yang. Rasa: A rapid retrosynthesis-based scoring method for the assessment of synthetic accessibility of drug-like molecules. *Journal of Chemical Information and Modeling*, 51(10):2768–2777, 2011. doi: 10.1021/ci100216g. URL <http://pubs.acs.org/doi/abs/10.1021/ci100216g>.
- Tomas Hudlicky and Josephine W. Reed. *The Way of Synthesis: Evolution of Design and Methods for Natural Products*. Wiley-VCH, 2007.
- Nicolas Hulo, Amos Bairoch, Virginie Bulliard, Lorenzo Cerutti, Beatrice A Cuche, Edouard de Castro, Corinne Lachaize, Petra S Langendijk-Genevaux, and Christian J A Sigrist. The 20 years of PROSITE. *Nucleic Acids Res*, 36(Database issue):D245–9, Jan 2008. ISSN 1362-4962 (Electronic); 1362-4962 (Linking). doi: 10.1093/nar/gkm977.
- David Jessop, Sam Adams, and Peter Murray-Rust. Mining chemical information from open patents. *Journal of Cheminformatics*, 3(1):40, 2011. ISSN 1758-2946. doi: 10.1186/1758-2946-3-40. URL <http://www.jcheminf.com/content/3/1/40>.
- A. Peter Johnson, Chris Marshall, and Philip N. Judson. Starting material oriented retrosynthetic analysis in the LHASA program. 1. general description. *Journal of Chemical Information and Computer Sciences*, 32(5):411–417, 1992. doi: 10.1021/ci00009a003. URL <http://pubs.acs.org/doi/abs/10.1021/ci00009a003>.
- W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Cryst Section A*, 34(5):827–828, Sep 1978. doi: 10.1107/S0567739478001680. URL <http://dx.doi.org/10.1107/S0567739478001680>.

- Abdullah Kahraman, Richard J Morris, Roman A Laskowski, and Janet M Thornton. Shape variation in protein binding pockets and their ligands. *J Mol Biol*, 368(1):283–301, Apr 2007. ISSN 0022-2836 (Print); 0022-2836 (Linking). doi: 10.1016/j.jmb.2007.01.086.
- Y. Kawano. Using similar positions to search game trees. In *Games of No Chance*, volume 29. MSRI Publications, 1996.
- Akihiro Kishimoto. *Correct and Efficient Search Algorithms in the Presence of Repetitions*. PhD thesis, University of Alberta, March 2005.
- Akihiro Kishimoto. Dealing with infinite loops, underestimation, and overestimation of depth-first proof-number search. *AAAI*, 2010.
- Akihiro Kishimoto and Martin Müller. Df-pn in go: An application to the one-eye problem. In H. Jaap van den Herik, Hiroyuki Iida, and Ernst A. Heinz, editors, *ACG*, volume 263 of *IFIP*, pages 125–142. Kluwer, 2003. ISBN 1-4020-7709-2.
- Akihiro Kishimoto and Martin Müller. A general solution to the graph history interaction problem. *AAAI*, pages 644–649, 2004.
- Akihiro Kishimoto and Martin Müller. About the completeness of depth-first proof-number search. In H. Jaap van den Herik, Xinhe Xu, Zongmin Ma, and Mark H. M. Winands, editors, *Computers and Games*, volume 5131 of *Lecture Notes in Computer Science*, pages 146–156. Springer, 2008. ISBN 978-3-540-87607-6.
- G J Kleywegt. Recognition of spatial motifs in protein structures. *J Mol Biol*, 285(4):1887–1897, Jan 1999. ISSN 0022-2836 (Print); 0022-2836 (Linking). doi: 10.1006/jmbi.1998.2393.
- N Kobayashi and N Go. ATP binding proteins with different folds share a common ATP-binding structural motif. *Nat Struct Biol*, 4(1):6–7, Jan 1997. ISSN 1072-8368 (Print); 1072-8368 (Linking).
- R T Koehler, H O Villar, K E Bauer, and D L Higgins. Ligand-based protein alignment and isozyme specificity of glutathione S-transferase inhibitors. *Proteins*, 28(2):202–216, Jun 1997. ISSN 0887-3585 (Print); 0887-3585 (Linking).
- Yehuda Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, April 2010. ISSN 0001-0782. doi: 10.1145/1721654.1721677. URL <http://doi.acm.org/10.1145/1721654.1721677>.
- Richard E. Korf. Iterative-deepening A*: An optimal admissible tree search. In *IJCAI*, pages 1034–1036, 1985.

Richard E. Korf. Linear-space best-first search: Summary of results. *AAAI*, 1992.

Dexter Kozen. *Automata and computability*. Springer, New York, 1997. ISBN 0387949070 (hardcover : alk. paper). URL <http://www.loc.gov/catdir/enhancements/fy0815/96037409-d.html>.

Daren Krebsbach, Herbert L. Gelernter, and Scott McN. Sieburth. Distributed heuristic synthesis search. *Journal of Chemical Information and Computer Sciences*, 38(4):595–604, 1998.

Peter S. Kutchukian, Nadya Y. Vasilyeva, Jordan Xu, Mika K. Lindvall, Michael P. Dillon, Meir Glick, John D. Coley, and Natasja Brooijmans. Inside the mind of a medicinal chemist: The role of human bias in compound prioritization during drug discovery. *PLoS ONE*, 7(11):e48476, 11 2012. doi: 10.1371/journal.pone.0048476. URL <http://dx.doi.org/10.1371%2Fjournal.pone.0048476>.

Yosef Y Kuttner, Vladimir Sobolev, Alexander Raskind, and Marvin Edelman. A consensus-binding structure for adenine at the atomic level permits searching for the ligand site in a wide spectrum of adenine-containing complexes. *Proteins*, 52(3):400–411, Aug 2003. ISSN 1097-0134 (Electronic); 1097-0134 (Linking). doi: 10.1002/prot.10422.

Michael S. Lajiness, Gerald M. Maggiora, and Veerabahu Shanmugasundaram. Assessment of the consistency of medicinal chemists in reviewing sets of compounds. *Journal of Medicinal Chemistry*, 47(20):4891–4896, 2004. doi: 10.1021/jm049740z. URL <http://pubs.acs.org/doi/abs/10.1021/jm049740z>.

Greg Landrum. Rdkit: Open-source cheminformatics. <http://www.rdkit.org>, 2006.

Sarah R. Langdon, Peter Ertl, and Nathan Brown. Bioisosteric replacement and scaffold hopping in lead generation and optimization. *Molecular Informatics*, 29(5):366–385, 2010. ISSN 1868-1751. doi: 10.1002/minf.201000019. URL <http://dx.doi.org/10.1002/minf.201000019>.

James Law, Zsolt Zsoldos, Aniko Simon, Darryl Reid, Yang Liu, Sing Yoong Khew, A. Peter Johnson, Sarah Major, Robert A. Wade, and Howard Y. Ando. Route designer: A retrosynthetic analysis tool utilizing automated retrosynthetic rule generation. *J. Chem. Inf. Model*, 49(3):593–602, 2009. doi: 10.1021/ci800228y. URL <http://pubs.acs.org/doi/abs/10.1021/ci800228y>.

Alexander J. Lawson and Hartmut Kallies. Multistep reactions: the RABBIT approach. *Journal of Chemical Information and Computer Sciences*, 30(4):426–430, 1990. doi: 10.1021/ci00068a013. URL <http://pubs.acs.org/doi/abs/10.1021/ci00068a013>.

Andrew G. Leach, Huw D. Jones, David A. Cosgrove, Peter W. Kenny, Linette Ruston, Philip MacFaul, J. Matthew Wood, Nicola Colclough, and Brian Law. Matched molecular pairs as a guide in the opti-

- mization of pharmaceutical properties; a study of aqueous solubility, plasma protein binding and oral exposure. *Journal of Medicinal Chemistry*, 49(23):6672–6682, 2006. doi: 10.1021/jm0605233. URL <http://pubs.acs.org/doi/abs/10.1021/jm0605233>.
- Weizhong Li, Hamish McWilliam, Ana Richart de la Torre, Adam Grodowski, Irina Benediktovich, Mickael Goujon, Stephane Nauche, and Rodrigo Lopez. Non-redundant patent sequence databases with value-added annotations at two levels. *Nucleic Acids Research*, 38(suppl 1):D52–D56, 2010. doi: 10.1093/nar/gkp960. URL http://nar.oxfordjournals.org/content/38/suppl_1/D52.abstract.
- Qianhui Althea Liang and Stanley Y. W. Su. AND/OR graph and search algorithm for discovering composite web services. *Int. J. Web Service Res.*, 2(4):48–67, 2005.
- Daniel M. Lowe, Peter T. Corbett, Peter Murray-Rust, and Robert C. Glen. Chemical name to structure: Opsin, an open source solution. *Journal of Chemical Information and Modeling*, 51(3):739–753, 2011. doi: 10.1021/ci100384d. URL <http://pubs.acs.org/doi/abs/10.1021/ci100384d>.
- C.D.A. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. Mit Press, 1999. ISBN 9780262133609. URL <http://books.google.ca/books?id=YiFDxbEX3SUC>.
- Alberto Martelli and Ugo Montanari. Additive AND/OR graphs. In *IJCAI*, pages 1–11, 1973.
- Arman Masoumi and Mikhail Soutchanski. Reasoning about chemical reactions using the situation calculus. In *AAAI Fall Symposium Series*, 2012. URL <https://aaai.org/ocs/index.php/FSS/FSS12/paper/view/5635>.
- Nicholas A. Meanwell. Synopsis of some recent tactical application of bioisosteres in drug design. *Journal of Medicinal Chemistry*, 54(8):2529–2591, 2011. doi: 10.1021/jm1013693. URL <http://pubs.acs.org/doi/abs/10.1021/jm1013693>.
- MIT. MIT OpenCourseWare. <http://ocw.mit.edu>, 2003.
- H. L. Morgan. The generation of a unique machine description for chemical structures—a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, 5(2):107–113, 1965. doi: 10.1021/c160017a018. URL <http://pubs.acs.org/doi/abs/10.1021/c160017a018>.
- Martin Müller. Proof-set search. In Jonathan Schaeffer, Martin Müller, and Yngvi Björnsson, editors, *Computers and Games*, volume 2883 of *Lecture Notes in Computer Science*, pages 88–107. Springer, 2002. ISBN 3-540-20545-4.

- Martin Müller. Proof-set search. In *Computers and Games*, volume 2883, pages 88–107. Lecture Notes in Computer Science, 2003.
- J. Munkres. Algorithms for the Assignment and Transportation Problems. *J Soc Ind and App Math*, 5(1):32–38, March 1957.
- Ayumu Nagai. A new AND/OR Tree Search Algorithm Using Proof Number and Disproof Number. In Ian Frank, Hitoshi Matsubara, Morihiko Tajima, Atsushi Yoshikawa, Reijer Grimbergen, and Martin Müller, editors, *Complex Games Lab Workshop*, pages 40–45. Electrotechnical Laboratory, Machine Inference Group, Tsukuba, Japan, 1998. URL <http://www.cs.ualberta.ca/~mmueller/cgo/cg98-workshop/Nagai.pdf>.
- Ayumu Nagai. *Df-pn Algorithm for Searching AND/OR Trees and Its Applications*. PhD thesis, University of Tokyo, 2002.
- Rafael Najmanovich, Natalja Kurbatova, and Janet Thornton. Detection of 3D atomic similarities and their use in the discrimination of small molecule protein-binding sites. *Bioinformatics*, 24(16):i105–11, Aug 2008. ISSN 1367-4811 (Electronic); 1367-4811 (Linking). doi: 10.1093/bioinformatics/btn263.
- Takashi Nakayama. Computer-assisted knowledge acquisition system for synthesis planning. *Journal of Chemical Information and Computer Sciences*, 31(4):495–503, 1991. doi: 10.1021/ci00004a011. URL <http://pubs.acs.org/doi/abs/10.1021/ci00004a011>.
- K. M. Nalin de Silva and Jonathan M. Goodman. What is the smallest saturated acyclic alkane that cannot be made? *Journal of Chemical Information and Modeling*, 45(1):81–87, 2005. doi: 10.1021/ci0497657. URL <http://pubs.acs.org/doi/abs/10.1021/ci0497657>.
- Dana S. Nau, Yue Cao, Amnon Lotem, and Héctor Muñoz-Avila. SHOP: Simple hierarchical ordered planner. In Thomas Dean, editor, *IJCAI*, pages 968–975. Morgan Kaufmann, 1999. ISBN 1-55860-613-0.
- Jean-Christophe Nebel. Modelling of P450 active site based on consensus 3D structures. *Int Conf Biomed Eng*, February 2005.
- Jean-Christophe Nebel. Generation of 3D templates of active sites of proteins with rigid prosthetic groups. *Bioinformatics*, 22(10):1183–1189, May 2006. ISSN 1367-4803 (Print); 1367-4803 (Linking). doi: 10.1093/bioinformatics/btl040.
- Jean-Christophe Nebel, Pawel Herzyk, and David R Gilbert. Automatic generation of 3D motifs for classification of protein binding sites. *BMC Bioinformatics*, 8:321, 2007. ISSN 1471-2105 (Electronic); 1471-2105 (Linking). doi: 10.1186/1471-2105-8-321.

NextMove Software. ChemDraw CDX File Parser v2.0. <http://nextmovesoftware.com/>, 2010.

K. C. Nicolaou and E. J. Sorensen. *Classics in Total Synthesis: Targets, Strategies, Methods*. Wiley-VCH, 1996.

Nils J. Nilsson. *Principles of Artificial Intelligence*. Morgan Kaufmann, 1980.

Noel O'Boyle, Chris Morley, and Geoffrey Hutchison. Pybel: a python wrapper for the openbabel cheminformatics toolkit. *Chemistry Central Journal*, 2(1):5, 2008. ISSN 1752-153X. doi: 10.1186/1752-153X-2-5. URL <http://journal.chemistrycentral.com/content/2/1/5>.

Noel O'Boyle, Michael Banck, Craig James, Chris Morley, Tim Vandermeersch, and Geoffrey Hutchison. Open babel: An open chemical toolbox. *Journal of Cheminformatics*, 3(1):33, 2011a. ISSN 1758-2946. doi: 10.1186/1758-2946-3-33. URL <http://www.jcheminf.com/content/3/1/33>.

Noel O'Boyle, Rajarshi Guha, Egon Willighagen, Samuel Adams, Jonathan Alvarsson, Jean-Claude Bradley, Igor Filippov, Robert Hanson, Marcus Hanwell, Geoffrey Hutchison, Craig James, Nina Jeliazkova, Andrew Lang, Karol Langner, David Lonie, Daniel Lowe, Jerome Pansanel, Dmitry Pavlov, Ola Spjuth, Christoph Steinbeck, Adam Tenderholt, Kevin Theisen, and Peter Murray-Rust. Open data, open source and open standards in chemistry: The blue obelisk five years on. *Journal of Cheminformatics*, 3(1):37, 2011b. ISSN 1758-2946. doi: 10.1186/1758-2946-3-37. URL <http://www.jcheminf.com/content/3/1/37>.

Martin A. Ott and Jan H. Noordik. Long-range strategies in the LHASA program: The quinone Diels-Alder transform. *Journal of Chemical Information and Computer Sciences*, 37(1):98–108, 1997. doi: 10.1021/ci9600972. URL <http://pubs.acs.org/doi/abs/10.1021/ci9600972>.

George A. Patani and Edmond J. LaVoie. Bioisosterism: A rational approach in drug design. *Chemical Reviews*, 96(8):3147–3176, 1996. doi: 10.1021/cr950066q. URL <http://pubs.acs.org/doi/abs/10.1021/cr950066q>.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

György Pirok, Nóra Máté, Jenő Varga, József Szegezdi, Miklós Vargyas, Szilárd Dóránt, and Ferenc Csizmadia. Making “real” molecules in virtual space. *Journal of Chemical Information and Modeling*, 46(2):563–568, 2006. doi: 10.1021/ci050373p. URL <http://pubs.acs.org/doi/abs/10.1021/ci050373p>. PMID: 16562984.

- Aske Plaat, Jonathan Schaeffer, Wim Pijls, and Arie de Bruin. Best-first fixed-depth minimax algorithms. *Artif. Intell.*, 87(1-2):255–293, 1996.
- Yevgeniy Podolyan, Michael A. Walters, and George Karypis. Assessing synthetic accessibility of chemical compounds using machine learning methods. *Journal of Chemical Information and Modeling*, 50(6):979–991, 2010. doi: 10.1021/ci900301v. URL <http://pubs.acs.org/doi/abs/10.1021/ci900301v>.
- E. L. Post. Recursive unsolvability of a problem of Thue. *J. Symbolic Logic*, 12:1–11, 1947.
- Syed Rahman, Matthew Bashton, Gemma Holliday, Rainer Schrader, and Janet Thornton. Small molecule subgraph detector (smsd) toolkit. *Journal of Cheminformatics*, 1(1):12, 2009. ISSN 1758-2946. doi: 10.1186/1758-2946-1-12. URL <http://www.jcheminf.com/content/1/1/12>.
- Milan Randić and Dejan Plavić. On the concept of molecular complexity. *Croatica Chemica Acta*, 75(1):107–116, 2002.
- Panolil Raveendranath, Joseph Zeldis, Galina Vid, John R. Potoski, Jianxin Ren, and Silvio Iera. Aryloxy-alkyl-dialkylamines. *U.S. Patent Trademark Office*, US 6,268,504, 1999.
- Jussi Rintanen. Complexity of planning with partial observability. In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *ICAPS*, pages 345–354. AAAI, 2004. ISBN 1-57735-200-9.
- David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of Chemical Information and Modeling*, 50(5):742–754, 2010. doi: 10.1021/ci100050t. URL <http://pubs.acs.org/doi/abs/10.1021/ci100050t>. PMID: 20426451.
- Bruce D. Roth. The discovery and development of atorvastatin, a potent novel hypolipidemic agent. In F.D. King, A.W. Oxford, A. B. Reitz, and S. L. Dax, editors, *Progress in Medicinal Chemistry*, volume 40, pages 1 – 22. Elsevier, 2002. doi: DOI:10.1016/S0079-6468(08)70080-8. URL <http://www.sciencedirect.com/science/article/B7CV8-4SGTFHJ-5/2/98df40b9f329367cc9b3a6ada0de2e0b>.
- Stephen D. Roughley and Allan M. Jordan. The medicinal chemist’s toolbox: An analysis of reactions used in the pursuit of drug candidates. *Journal of Medicinal Chemistry*, 54(10):3451–3479, 2011. doi: 10.1021/jm200187y. URL <http://pubs.acs.org/doi/abs/10.1021/jm200187y>.
- Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 1995.
- Koji Satoh and Kimito Funatsu. A novel approach to retrosynthetic analysis using knowledge bases derived from reaction databases. *Journal of Chemical Information and Computer Sciences*, 39(2):316–325, 1999.

Jonathan Schaeffer. The history heuristic and alpha-beta search enhancements in practice. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(11):1203–1212, 1989.

Jonathan Schaeffer. *One Jump Ahead: Computer Perfection at Checkers*. Springer-Verlag, 2009.

Jonathan Schaeffer, Neil Burch, Yngvi Bjornsson, Akihiro Kishimoto, Martin Muller, Robert Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *Science*, 317(5844):1518–1522, July 2007. doi: 10.1126/science.1144079. URL <http://dx.doi.org/10.1126/science.1144079>.

Martin Schijf. Proof-number search and transpositions. Master's thesis, University of Leiden, 1993.

Dina Schneidman-Duhovny, Oranit Dror, Yuval Inbar, Ruth Nussinov, and Haim J Wolfson. Deterministic pharmacophore detection via multiple flexible alignment of drug-like molecules. *J Comput Biol*, 15(7):737–754, Sep 2008. ISSN 1557-8666 (Electronic); 1557-8666 (Linking). doi: 10.1089/cmb.2007.0130.

Matthew Segall, Ed Champness, Chris Leeding, Ryan Lilien, Ramgopal Mettu, and Brian Stevens. Applying medicinal chemistry transformations and multiparameter optimization to guide the search for high-quality leads and candidates. *Journal of Chemical Information and Modeling*, 51(11):2967–2976, 2011. doi: 10.1021/ci2003208. URL <http://pubs.acs.org/doi/abs/10.1021/ci2003208>.

Masahiro Seo, Hiroyuki Iida, and Jos W. H. M. Uiterwijk. The PN^{*}-search algorithm: Application to tsume-shogi. *Artif. Intell.*, 129(1-2):253–277, 2001.

Maxim Shatsky, Alexandra Shulman-Peleg, Ruth Nussinov, and Haim J Wolfson. The multiple common point set problem and its application to molecule binding pattern detection. *J Comput Biol*, 13(2):407–428, Mar 2006. ISSN 1066-5277 (Print); 1066-5277 (Linking). doi: 10.1089/cmb.2006.13.407.

Robert P. Sheridan. The most common chemical replacements in drug-like compounds. *Journal of Chemical Information and Computer Sciences*, 42(1):103–108, 2002. doi: 10.1021/ci0100806. URL <http://pubs.acs.org/doi/abs/10.1021/ci0100806>.

Robert P. Sheridan. Time-split cross-validation as a method for estimating the goodness of prospective prediction. *Journal of Chemical Information and Modeling*, 53(4):783–790, 2013. doi: 10.1021/ci400084k. URL <http://pubs.acs.org/doi/abs/10.1021/ci400084k>.

Noel T. Southall and Ajay. Kinase patent space visualization using chemical replacements. *Journal of Medicinal Chemistry*, 49(6):2103–2109, 2006. doi: 10.1021/jm051201m. URL <http://pubs.acs.org/doi/abs/10.1021/jm051201m>.

Christoph Steinbeck, Yongquan Han, Stefan Kuhn, Oliver Horlacher, Edgar Luttmann, and Egon Willighagen.

The chemistry development kit (cdk): An open-source java library for chemo- and bioinformatics. *Journal of Chemical Information and Computer Sciences*, 43(2):493–500, 02 2003. doi: 10.1021/ci025584y. URL <http://dx.doi.org/10.1021/ci025584y>.

Kent D. Stewart, Melisa Shiroda, and Craig A. James. Drug guru: A computer software program for drug design using medicinal chemistry rules. *Bioorganic and Medicinal Chemistry*, 14(20):7011 – 7022, 2006. ISSN 0968-0896. doi: DOI:10.1016/j.bmc.2006.06.024. URL <http://www.sciencedirect.com/science/article/pii/S0968089606004895>.

Gareth R Stockwell and Janet M Thornton. Conformational diversity of ligands bound to proteins. *J Mol Biol*, 356(4):928–944, Mar 2006. ISSN 0022-2836 (Print); 0022-2836 (Linking). doi: 10.1016/j.jmb.2005.12.012.

Edward M. Suh and Yoshito Kishi. Synthesis of palytoxin from palytoxin carboxylic acid. *Journal of the American Chemical Society*, 116(24):11205–11206, 1994. doi: 10.1021/ja00103a065. URL <http://pubs.acs.org/doi/abs/10.1021/ja00103a065>.

M. Takahashi, I. Dogane, M. Yoshida, H. Yamachika, T. Takabatake, and Malcolm Bersohn. The performance of a noninteractive synthesis program. *Journal of Chemical Information and Computer Sciences*, 30(4):436–441, 1990.

Yuji Takaoka, Yutaka Endo, Susumu Yamanobe, Hiroyuki Kakinuma, Taketoshi Okubo, Youichi Shimazaki, Tomomi Ota, Shigeyuki Sumiya, and Kensei Yoshikawa. Development of a method for evaluating drug-likeness and ease of synthesis using a data set in which compounds are assigned scores based on chemists' intuition. *Journal of Chemical Information and Computer Sciences*, 43(4):1269–1275, 2003. doi: 10.1021/ci034043l. URL <http://pubs.acs.org/doi/abs/10.1021/ci034043l>.

Akio Tanaka, Hideho Okamoto, and Malcolm Bersohn. Construction of functional group reactivity database under various reaction conditions automatically extracted from reaction database in a synthesis design system. *Journal of Chemical Information and Modeling*, 50(3):327–338, 2010. doi: 10.1021/ci9004332. URL <http://pubs.acs.org/doi/abs/10.1021/ci9004332>. PMID: 20187659.

Taffee T Tanimoto. An elementary mathematical theory of classification and prediction. Technical report, IBM, 1958.

Harry Thangaraj. Information from patent office could aid replication. *Nature*, 447(7145):638–638, 06 2007. URL <http://dx.doi.org/10.1038/447638c>.

- Andrew Thurkauf, Xiao shu He, He Zhao, John Peterson, Xiaoyan Zhang, Robbin Brodbeck, James Krause, George Maynard, and Alan Hutchison. High affinity small molecule c5a receptor modulators. *U.S. Patent Trademark Office*, US 6,884,815, 2003.
- Matthew H. Todd. Computer-aided organic synthesis. *Chem. Soc. Rev.*, 34:247–266, 2005. doi: 10.1039/B104620A. URL <http://dx.doi.org/10.1039/B104620A>.
- Shin-Shyong Tseng. Computer-assisted reaction searching directed toward the synthesis of target molecules. *J. Chem. Inf. Comput. Sci.*, 37:1138–1145, 1997.
- Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.
- Aniko T. Valko and A. Peter Johnson. Clide pro: The latest generation of clide, a tool for optical chemical structure recognition. *Journal of Chemical Information and Modeling*, 49(4):780–787, 2009. doi: 10.1021/ci800449t. URL <http://pubs.acs.org/doi/abs/10.1021/ci800449t>.
- Karin Verspoor, K Bretonnel Cohen, and Lawrence Hunter. The textual characteristics of traditional and open access scientific journals are similar. *BMC Bioinformatics*, 10(1):183, 2009. ISSN 1471-2105. doi: 10.1186/1471-2105-10-183. URL <http://www.biomedcentral.com/1471-2105/10/183>.
- Markus Wagener and Jos P. M. Lommerse. The quest for bioisosteric replacements. *Journal of Chemical Information and Modeling*, 46(2):677–685, 2006. doi: 10.1021/ci0503964. URL <http://pubs.acs.org/doi/abs/10.1021/ci0503964>.
- R. Waldinger. Achieving several goals simultaneously. In E. W. Elcock and D. Michie, editors, *Machine Intelligence*, volume 8, pages 94–136. Wiley, 1977.
- Ke Wang, Lisha Wang, Qiong Yuan, Shiwei Luo, Jianhua Yao, Shengang Yuan, Chongzhi Zheng, and Josef Brandt. Construction of a generic reaction knowledge base by reaction data mining. *Journal of Molecular Graphics and Modelling*, 19(5):427 – 433, 2001. ISSN 1093-3263. doi: DOI:10.1016/S1093-3263(00)00102-9. URL <http://www.sciencedirect.com/science/article/B6TGP-43CBCSP-5/2/790f9148962606d8ff2ec7941cf23066>.
- Yanli Wang, Jewen Xiao, Tugba O. Suzek, Jian Zhang, Jiyao Wang, and Stephen H. Bryant. PubChem: a public information system for analyzing bioactivities of small molecules. *Nucleic Acids Research*, 37(suppl 2):W623–W633, 2009. doi: 10.1093/nar/gkp456. URL http://nar.oxfordjournals.org/content/37/suppl_2/W623.abstract.

- Wendy Warr. ChEMBL: an interview with John Overington, team leader, chemogenomics at the European Bioinformatics Institute, Outstation of the European Molecular Biology Laboratory (EMBL-EBI). *Journal of Computer-Aided Molecular Design*, 23:195–198, 2009. ISSN 0920-654X. URL <http://dx.doi.org/10.1007/s10822-009-9260-9>. 10.1007/s10822-009-9260-9.
- David Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28(1):31–36, 1988. doi: 10.1021/ci00057a005. URL <http://pubs.acs.org/doi/abs/10.1021/ci00057a005>.
- Daniel S. Weld. An introduction to least commitment planning. *AI Magazine*, 15(4):27–61, 1994.
- Martina Wernerova and Tomas Hudlicky. On the practical limits of determining isolated product yields and ratios of stereoisomers: Reflections, analysis, and redemption. *Synlett*, 2010(18):2701–2707, 2010. doi: 10.1055/s-0030-1259018.
- Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2 edition, 2000. ISBN 0130144002.
- Craig S. Wilcox and Robert A. Levinson. *A Self-Organized Knowledge Base for Recall, Design, and Discovery in Organic Chemistry*, volume 306 of *ACS Symposium Series*, chapter 18, pages 209–230. American Chemical Society, 1986. doi: 10.1021/bk-1986-0306.ch018. URL <http://pubs.acs.org/doi/abs/10.1021/bk-1986-0306.ch018>.
- W. Todd Wipke, Glenn I. Ouchi, and S. Krishnan. Simulation and evaluation of chemical synthesis—SECS: An application of artificial intelligence techniques. *Artificial Intelligence*, 11(1-2):173 – 193, 1978. ISSN 0004-3702. doi: DOI:10.1016/0004-3702(78)90016-4. URL <http://www.sciencedirect.com/science/article/B6TYF-48VPC54-J/2/1ab50ab3e4bb5c558cf567f625d80767>. Applications to the Sciences and Medicine.
- Bianca Zadrozny, John Langford, and Naoki Abe. Cost-sensitive learning by cost-proportionate example weighting. In *ICDM*, pages 435–. IEEE Computer Society, 2003.