Interactive evolutionary algorithms and data mining for drug design

Lameijer, E.M.W.

Interactive Evolutionary Algorithms and Data Mining for Drug Design

Of Molecules, Machines, and Men

Interactive Evolutionary Algorithms and Data Mining for Drug Design

Of Molecules, Machines, and Men

Proefschrift

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van Rector Magnificus prof. mr. P.F. Van der Heijden,
volgende besluit van het College voor Promoties
te verdedigen op donderdag 28 januari 2010
klokke 13.45 uur

door

Eric Marcel Wubbo Lameijer
geboren te Hilversum
in 1976

Promotiecommissie

Promotoren: Prof. Dr. A. P. IJzerman
Prof. Dr. J. N. Kok

Overige leden: Prof. Dr. Th. W. Bäck
Dr. A. Bender
Prof. Dr. M. Danhof
Prof. Dr. H. van Vlijmen
Dr. M. Wagener

The BioScience Initiative of the Leiden University Faculty of Mathematics and Natural Sciences is gratefully acknowledged for funding the research described in this thesis.

Cidrux Pharminformatics is gratefully acknowledged for financially supporting the printing of this thesis.

The research described in this thesis was performed at the Division of Medicinal Chemistry of the Leiden/Amsterdam Center for Drug Research, Leiden University, the Netherlands, in collaboration with the Leiden Institute for Advanced Computer Sciences at the same university.

Cover design by Mathijs Wansink.

This thesis was printed by Wöhrmann Print Service (Zutphen, the Netherlands).

Hell, there are no rules here. We're trying to accomplish something.

Thomas Alva Edison

Dedicated to the three persons who laid the foundation for this work:
my father, who imbued me with his love of knowledge
my mother, who always encouraged me to play with ideas
and my highschool teacher, Olaf Budde, who taught me the joy of chemistry

Contents

Chapter 1 Introduction: Molecules, Machines and Men 1

Chapter 2 Evolutionary Algorithms in Drug Design 11

Chapter 3 Mining a Chemical Database for Fragment Cooccurrence: Discovery of "Chemical Clichés" 81

Chapter 4 The Molecule Evoluator. An interactive evolutionary algorithm for the design of drug-like molecules 109

Chapter 5 Using Data Mining to Improve Mutation in a Tool for Molecular Evolution 131

Chapter 6 Evolutionary algorithms in *de novo* molecule design: comparing atom-based and fragment-based optimization 149

Chapter 7 Designing Active Template Molecules by Combining Computational *De Novo* Design and Human Chemist's Expertise 191

Chapter 8 Conclusions and Future Perspectives 219

Summary 237

Samenvatting 241

Curriculum Vitae 245

List of Publications 247

Epilogue 248

1 Introduction: Molecules, Machines and Men

The research described in this thesis focused on “interactive evolutionary algorithms and data mining for drug design”. That may sound impressive, but what does it mean? The purpose of this introduction is to ensure that people who do not know much about interactive evolutionary algorithms and data mining will have a pretty good idea what those are after reading the next few pages, and that people who are already familiar with the field will get a more intimate acquaintance with the problems we are trying to solve, and the perspective that we have. Interactive evolution and data mining are really just formal names and procedures for activities all of us already do in daily life: whenever you redecorate your room, you're performing a kind of interactive evolution, by asking yourself whether the room would look better if you painted it blue or soft yellow or added a large portrait of Barack Obama. Whenever you are browsing the newspaper, looking for interesting articles, you are doing a form of data mining. Science is often just common sense formalized. This thesis will discuss interactive evolutionary algorithms for drug design, as well as data mining, but these are merely sophisticated tools to achieve our goal: to find new or better drug molecules.

The research I have done was a collaboration between the department of Medicinal Chemistry, which focuses on developing biologically active molecules, and the Algorithms group of computer science, which investigates the use of computer methods to solve real-world problems. The subject of my research was therefore how to use computers (machines) to design drugs (molecules), which are discussed in the next two sections of this introduction. However, while creating the computer programs, we found out that by merely focusing on software and molecular structures we were neglecting something crucial: the scientists themselves. Creating a computer program that should be used by people required us to pay attention to how people think, and how a computer program can be made intuitive and easy to use. We also found that it was extremely useful to complement the molecule-generating capabilities of the computer with the experience and pattern-recognition ability of people. We therefore also dedicated a section to the third factor in this research, the “interactive” in interactive evolution. After these sections on molecules, machines and man, there will be an introduction to some of the terms used in this thesis. Finally, we will discuss the aims of this thesis and give an outline of the chapters to follow.

Molecules, machines and man

Molecules

The human body viewed at normal scale already seems complex. However, when one zooms in to the microscopic level of cells and proteins, it becomes even more fascinating, for only on that scale the true complexity of our existence is revealed. The human body contains about one hundred trillion cells of over 200 distinct cell types, with 20,000 genes, which can produce at least as many proteins. It also contains a large variety of hormones, fatty acids, and other small organic compounds which help the cells and organs communicate and cooperate with each other in many ways, adjusting the activity of the body to whatever is needed in the circumstances in which we live.

Next to admiring the beauty and complexity of the workings of life, and satisfying our curiosity on how things work, there is also a very practical reason to strive to understand the human body: fighting disease. If we know how the human body works when it is healthy, and what happens when it falls ill, it should be easier to find a proper remedy for a disease. And in the end, it is not the understanding, but the action, the resulting medicine, that is important. However, even if one knows what is wrong in the body, the problem may still not be easy to correct.

Except from some cases in which the “diseased part” of the body can simply be removed (surgery), the most effective way to treat diseases is by administering drugs, which contain many billions of molecules of a specific compound. These drug molecules bind to biological molecules (usually proteins), either activating them or inhibiting them. This changes the behaviour of the protein, and thereby the behaviour of the cell, ultimately affecting the organ or even the whole body. For example, aspirin works by inhibiting cyclooxygenase, an enzyme which produces prostaglandins, compounds that cause pain. When someone takes aspirin, aspirin molecules diffuse through the gastro-intestinal wall and enter the bloodstream, where they block cyclooxygenase. With a reduced number of active cyclooxygenase enzymes which create pain-causing prostaglandins, less prostaglandins are produced, and so the pain is alleviated. By targeting the right step in biological processes, drugs can “reset” the body to a healthier state, or at least alleviate the symptoms of a disease.

There are however still many diseases which cannot be treated well with current medicine, for example AIDS, many forms of cancer, and Alzheimer's disease. Finding drugs for these and other diseases is difficult for several reasons. First of all, the mechanism of a disease is often not clear, so it is not always known which protein to target. The second problem is that even if a good target protein is found, a molecule must be developed which binds to it effectively. Also, these molecules must be able to get to the right place in the body and not be metabolized or excreted before they can reach the diseased area. And finally, the molecules should not interact strongly with other biological molecules, which would cause harmful side effects. Finding a molecule that both interacts effectively with the target *and* has favourable "ADME-tox" properties (absorption, distribution, metabolism, elimination, toxicity) is a very difficult and time-consuming process: it costs on average over 800 million dollars and 12 years of development time to bring a drug to the market[1].

Our goal in this project was therefore to investigate how we could help drug discovery become faster or better.

Machines

Finding new drugs for diseases is the 'why' of this project; let us now turn to the 'how': how can we improve the drug discovery process? In the past three decades, various methods have been developed to improve or speed up the drug design process: so-called "rational design", high-throughput screening, combinatorial chemistry, and, more recently, systems biology and bioinformatics. These methods, diverse as they are, have one striking common denominator: they all use computers.

Even while computers often only do "simple things fast", they can increase efficiency in scientific research tremendously. For example, when I was a MSc student, if I wanted to find information on a certain compound, I needed to manually search multiple annual editions of the chemical abstracts service (thick books), before I could jot down the numbers of the abstracts, which had to be looked up in *another* series of heavy books. Of course, if the abstract suggested that the article would be useful, I still had to locate the attic section and/or shelf where that specific edition of the journal was located, and then go to the copier to make a copy for myself. The process could take hours. Nowadays, using internet and search machines, one can find and print articles about a particular compound or topic in seconds or minutes. Next to doing fast calculations (allowing for example fast elucidation and visualisations of protein structure), and controlling complicated machinery (such as in high-throughput screening), information storage and distribution is probably the greatest benefit of IT.

For example, electronic lab journals allow companies to find out about already performed experiments much more easily than the "classical" method of finding a synthesis in a stack of paper lab journals.

However, what could we add to the already impressive array of computational techniques for aiding drug discovery? In this research we have focused on the possibilities of two fields of computer science: evolutionary algorithms and data mining.

Evolutionary algorithms address one of the traditional problems of computers: computers can be programmed to do anything that involves any sequence of fixed actions – but sometimes it is not known which actions are necessary to achieve the desired result. Finding a molecule that binds to a certain protein is a problem of this type: the goal is known, but there is no "procedure" that will systematically and unambiguously lead to the desired molecule. In practice, intelligent trial and error is needed. Evolution works this way too. First, it produces a large number of solutions (animals/plants) to certain problems (environments). Then, the best of these solutions procreate (are copied, changed/adapted and combined) to produce even better solutions in the next generation. Inventions and machines change over the generations just like organisms, and computer programs can simulate this by changing and combining the best designs of a collection of designs. In our case, those designs are molecules.

Data mining is another powerful technique, useful in cases where the programmer does not yet know what the "rules" of a system are, for example which factors in one's diet increase or decrease the risk of heart disease. By statistically analysing large amounts of data, data mining can unravel patterns in masses of data which may be hidden for the human eye. For example, software has been developed that correctly picked out the 10 known fraudsters (and about a dozen new suspects) in a database of the online auction site eBay – totalling one million transactions and 66000 users, far too much data for a human to analyze.[2,3] Likewise, data mining could give insights in hidden patterns in databases of molecules or drugs.

Looking for ways to help drug design, we therefore wondered how we could use data mining and evolutionary algorithms to our advantage.

Man

When one develops software that will also be used by others, a third factor needs to be taken into account, next to problem knowledge and computer knowledge: people. In my research this is more important than in day-to-day science, where for many scientists and programmers the existence of people almost seems an afterthought. Scientific papers are usually written in the passive voice, ranging from the standard "10 ml NaOH (1M) was added to the mixture" to the slightly deceptive "it was hypothesized that…", as if a hypothesis objectively and unambiguously follows from certain facts or experimental results. While perhaps scientists *should* behave objectively and perfectly rational, scientists are people, and people are not completely objective or rational, even though they may try. Therefore, if something needs to be used by humans, even if those humans are scientists, it is not sufficient that it is objectively and scientifically functional. And this is also true for software. Even a potentially useful computer program may not be used if people can't find the time or courage to read 500-page manuals to learn how to navigate through cumbersome, illogical menus. It was therefore important for us to pay attention to how people think, and how we could adapt the software to make it easier to use.

On the positive side, it would be wrong to see humans merely as imperfect reasoning machines. Humans have evolved in nature, where there is usually lack of useful information combined with a huge amount of useless information that obscures the useful information there might be, where there are urgent problems with not enough time to calculate all odds and all possible ways out, and where an incredible amount of knowledge is required to achieve even the most modest results – even walking up stairs is something most programmers dread to program robots for. Humans are far superior to computers in detecting new patterns, making connections between pieces of information, and thinking "out of the box" to solve a problem. Humans can easily solve many problems which baffle the most advanced computers, for example, recognizing a face even if it is seen from the side, understanding words even if they're spoken in dialect, or walking through a house without bumping against walls or furniture.

It would therefore be ideal if we could not only use the capabilities of the computer, but let the talents of the human/scientist complement these. However, combining humans and computers is not easy to do right. The first main problem is that to be of any kind of use, software must be user-friendly – software that cannot be understood by the user will not be used, even if it has tremendous capacities. Second, what things can or should we delegate to the computer, and what things can we ask of a human user? And can we close the gap so that there can be useful collaboration?

The third issue we had to pay attention to in this research was therefore how to effectively make use of human-computer collaboration. Computers can make calculations of molecule properties quickly, while chemists have lots of experience and intuition on which molecules are drug-like and which molecules can and cannot be synthesized. Yet any cooperation between man and computer can only occur if the software is sufficiently intuitive and user-friendly. The first word processor I used, Symphony, required the user to remember the key combination of <ALT>-<F1>-<B><A><B> to put anything in boldface, but such an interface would nowadays only discourage use. The last of the three questions is therefore how to design our chemical software in such a way as to ensure it will not only be useful, but also that it will be used.

Introduction to some of the terms and concepts used in our research

A number of computer science and cheminformatics terms will occur throughout this thesis, and while most of them will be explained in more formal terms in the following chapters, it may be useful for reader comprehension to clarify some of the most important concepts here.

Interactive Evolutionary Algorithm

One of the main aspects of evolution is selection, sometimes called "survival of the fittest". In evolutionary algorithms we also want the best solutions to survive and procreate, but to do that we have to determine what we mean by "best" or "fittest". Does "fittest" mean the strongest construction? The smallest molecule? The circuit board that gets the job done with least components? Or the circuit board that consumes least energy? Sometimes the fitness of a design can be calculated easily and objectively by a so-called "fitness function" which takes the organism/solution as input and returns a number that indicates its quality. Other times, though, the quality of a solution is difficult to calculate. For example, the ideal interior design of a room will depend on the taste of the human occupant.

Cases in which there is no objective way to calculate fitness are however not impossible to solve. Evolutionary algorithms can work if there exists *any* method to assign relative quality to solutions, and it is perfectly possible to have a human being as the "fitness function". That means that a human scores solutions or selects the ones he or she considers best. An evolutionary algorithm that uses a human to evaluate solutions is called an interactive evolutionary algorithm or interactive evolutionary computation (IEC). IECs have been used in many applications, varying from face image generation to help an eye-witness reconstruct the face of her attacker,[4] geophysics in which experts can distinguish realistic from unrealistic earth layer patterns, to helping people find better settings for their hearing aid.[5] Since interactive evolutionary algorithms can use both explicit and implicit/subconscious knowledge of drug design present in human medicinal chemists, it also seemed a promising approach for our research.

Data mining

Governments, companies, universities and many other organizations nowadays have large databases which house enormous amounts of data. Such data is useful in its own right (for example, checking how much money your bank account contains), but these databases also bring the promise that one can discover patterns and laws in the data, much like Kepler discovered the laws of planetary motion from his astronomical data. However, most databases are so vast that it would be hard or impossible for a human to find laws and patterns. For that reason, many scientists are working on techniques collectively called "data mining", which means that they develop software that can automatically find relationships between data or parameters. Usually data mining is performed on database tables, for example, whether there is a correlation between the education and the income of a person, and if yes, what it is and how strong it is, but it can be applied to any collection of data. For example, data mining also can handle a "shopping basket" problem in which a supermarket wants to find out whether people usually buy product X with product Y (such as bread and peanut butter). In this thesis, our main investigation of data mining is described in chapter 3, while chapter 4 and especially chapter 5 discuss how we used data mining to improve our main evolutionary algorithm.

Docking

Docking is a term used for computer simulations of the interaction between small molecules and proteins. Small molecules such as drugs influence the behaviour of proteins by crawling into a "sensitive" place in the interior of the protein, much like a key enters a lock or a hand fits into a glove. Similar to the docking of ships in a harbour, a "docking program" will attempt to find the best fit of a small molecule into an enzyme or receptor. However, docking is a difficult problem, and many different docking programs have been developed, such as GOLD, FlexX, DOCK and Glide,[6] each having its own strengths and weaknesses. For drug design, the ideal is to predict how well a drug candidate would bind to a receptor, so one could select the most promising leads from a large library of compounds without having to perform expensive syntheses and biological testing. However, docking programs are yet far from reliable for finding such quantitative binding strengths, since the exact strengths of electrostatic interactions and hydrogen bonds between ligand atoms and the amino acids in a protein are unknown, and most docking programs cannot simulate how a protein can mould itself around a ligand to improve binding. However, docking programs can often indicate how a molecule would fit into a protein, and despite their flaws they are currently the most reliable methods to theoretically compare binding affinities of a wide variety of small molecules. We used docking for the research in chapter 6, as despite its imperfections, docking is the best simulation of a `protein like`-system currently available.

Aims of this thesis

The aim of the research described in this thesis is to use evolutionary algorithms and data mining to help find new drugs.
For this purpose, we have:
-developed an internal representation of molecules and a set of mutations aimed to reach all possible molecules in chemical space.
-created a user interface that allows chemists to give input and feedback to the evolutionary algorithm efficiently and easily.
-mined large molecule databases to find frequent and infrequent substructures that can be used to design new molecules.

We also tested out the resulting interactive evolutionary algorithm in collaboration with the medicinal chemists at our laboratory. A set of compounds generated by the evolutionary algorithm was examined by the chemists, who selected the molecules they deemed most interesting and adjusted them for ease of synthesis. Subsequently, these compounds were synthesized to assess whether the methods we developed could indeed be used to find new biologically active molecules.

Outline of this thesis

This thesis will open with a review on the applications of evolutionary algorithms in drug design (chapter 2). Chapter 3 focuses on the question on how well current chemistry covers total chemical space –what is the real diversity of compounds? The answer is perhaps somewhat sobering (the term “chemical clichés” in the title of this chapter was coined for a reason), however we also indicate ways to use the data gathered to create more novel molecule scaffolds. Chapter 4 will discuss the Molecule Evoluator, a computer program we developed that uses an interactive evolutionary algorithm to create novel chemical compounds by using both computing power and chemist's intuition. In chapter 5, we show that the results of the Molecule Evoluator can be improved by combining the evolutionary algorithm with the technique of data mining, and show how the parameters of the evolutionary algorithm can be set to reflect the results of our data mining – which is not as straightforward as it may seem! Chapter 6 tackles the question whether atom- or fragment-based approaches are preferable for evolutionary algorithms in molecule design, by using docking to approximate the fitness of the compounds generated by the Molecule Evoluator. The part dedicated to our investigations closes with chapter 7, which looks into some real-world results: creating novel biologically active compounds which have been discovered by collaboration between medicinal chemists and the Molecule Evoluator. Finally, chapter 8 closes this thesis with the conclusions and my perspectives on the future of computers in drug design.

> "We have so much time and so little to do. Strike that, reverse it."
> - Willy Wonka, *Charlie and the Chocolate Factory* (Roald Dahl)

I hope you're set for the journey. Let's get started.

References

[1] DiMasi, J.A., Hansen, R.W., and Grabowski, H.G. The price of innovation: new estimates of drug development costs. *Journal of Health Economics* **2003**, *22*, 151-185.

[2] Pandit, S., Chau, D.H., Wang, S., and Faloutsos, C. NetProbe: A Fast and Scalable System for Fraud Detection in Online Auction Networks. *WWW 2007* **2007**.

[3] Simonite, T. Network analysis spots online-auction fraudsters. *New Scientist* (online edition) **2006**, December 6th.

[4] Marks, P. How to recall the face that fits. *New Scientist* **2005**, March 19th, p24.

[5] Takagi, H. Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation. *Proceedings of the IEEE* **2001**, *89*, 1275 – 1296.

[6] Klebe, G. Virtual ligand screening: strategies, perspectives and limitations. *Drug Discovery Today* **2006**, *11*, 580-594.

2 Evolutionary Algorithms in Drug Design

Eric-Wubbo Lameijer[1], Thomas Bäck[2,3], Joost N. Kok[2] and Ad P. IJzerman[1]

[1]Division of Medicinal Chemistry, Leiden/Amsterdam Center for Drug Research, Leiden University, PO Box 9502, 2300 RA Leiden, The Netherlands
[2]Leiden Institute of Advanced Computer Science, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
[3]NuTech Solutions GmbH, Martin-Schmeisser-Weg 15, 44227 Dortmund, Germany

This chapter was first published in the Journal of Natural Computing, reference: Lameijer, E.W.; Bäck, T.; Kok, J.N.; IJzerman, A.P. Evolutionary algorithms in drug design. Natural Computing, **2005**, *4*, 177-243.

Abstract

Designing a drug is the process of finding or creating a molecule which has a specific activity on a biological organism. Drug design is difficult since there are only few molecules that are both effective against a certain disease and exhibit other necessary physiological properties, such as absorption by the body and safety of use. The main problem of drug design is therefore how to explore the chemical space of many possible molecules to find the few suitable ones. Computational methods are increasingly being used for this purpose, among them evolutionary algorithms. This review will focus on the applications of evolutionary algorithms in drug design, in which evolutionary algorithms are used both to create new molecules and to construct methods for predicting the properties of real or yet unexisting molecules. We will also

discuss the progress and problems of application of evolutionary algorithms in this field, as well as possible developments and future perspectives.

1. Introduction

Drug design

Being healthy is usually taken for granted, but the importance of health becomes very clear when it is not present: the various illnesses can greatly diminish the quality and quantity of life, and are usually fought with all means available. One of the primary means of conserving health or improving quality of life is the administration of small molecules called drugs. These molecules can bind to specific critical components (generally proteins) of the target cells, and activating or deactivating these components leads to a change in behaviour of the entire cell. Cells of disease-causing organisms or of the patients themselves can be targeted[1], leading to destruction of the cells or modification of their behaviour. This can help to cure or at least alleviate the disease. Modern medicine has access to a large variety of compounds to fight diseases ranging from AIDS to high blood pressure, from cancer to headache, and from bacterial infection to depression.

Drugs, together with improved nutrition and hygiene, have led to a large increase in life expectancy in Western society (in 1900, life expectancy in the USA at birth was 47.3 years, which had increased to 77.0 years in 2000). However, there still exists a great need for new and better therapeutics. Current drugs can in most cases only slow cancer, not cure it. The remarkably effective treatment of HIV infection with combination therapy prevents the progression of AIDS, but the treatment itself is quite harmful to the body. And some illnesses, like Alzheimer’s disease, are still untreatable.

Unfortunately, developing a novel drug is not easy. The pharmaceutical industry is spending enormous amounts of time and effort to develop drugs that improve on existing ones or treat previously untreatable maladies. On average, development of a new drug takes 10 to 15 years and costs 400-800 million US dollars (DiMasi *et al.*, 2003). A large part of this money is spent on investigating compounds that eventually turn out to be unsuitable as drugs. Many molecules fail to become drugs because of “low bioavailability”, which means that they do not succeed in reaching the site of

[1] In the case of viruses, which have no cells themselves, the viral proteins which are present in the infected human cells are targeted, preventing or reducing proliferation of the virus.

action due to poor solubility in water/blood (Lipinski *et al.*, 1997), bad penetration of the gut wall, or being broken down by the body before they can exert their effect.

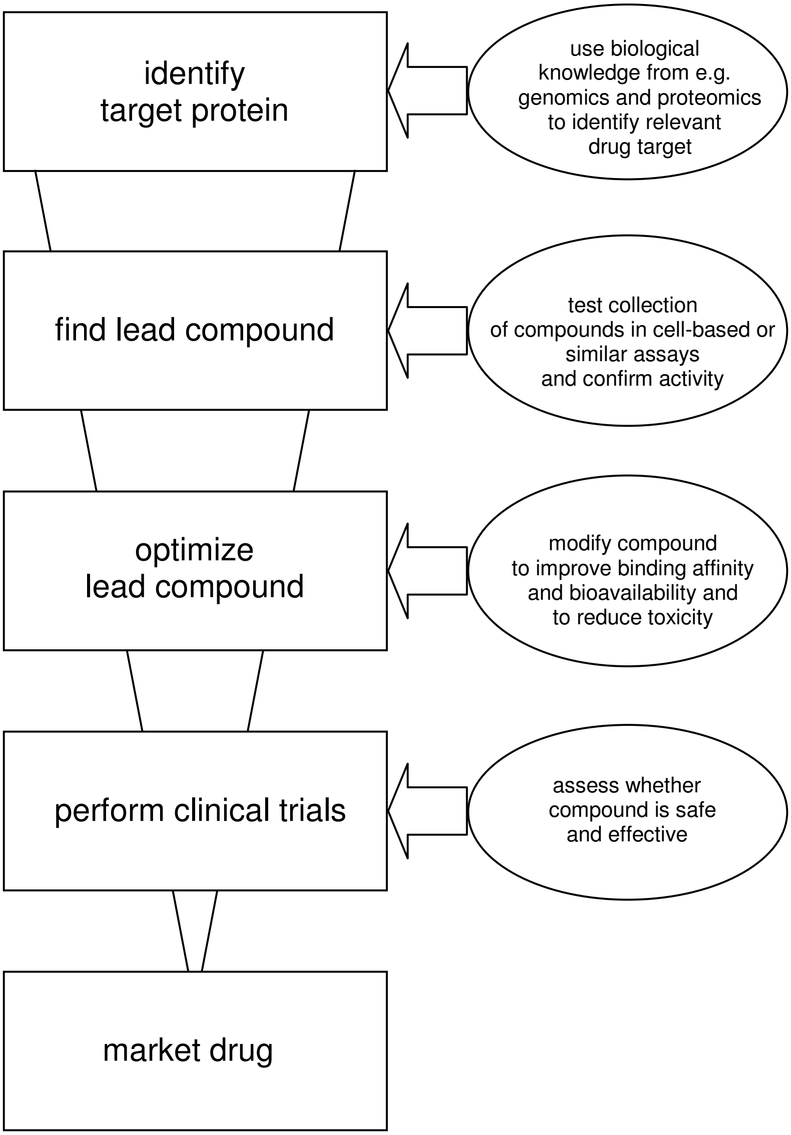


Figure 2.1: A schematic overview of the different phases of the drug development process

Additionally, the biological targets of the drug candidates may turn out not to have a significant influence on the disease, or the adverse effects outweigh the health benefits.

Due to these many independent factors that can make a drug candidate fail, it is hardly surprising that only one out of about 5000 screened drug candidates reaches the market (Rees, 2003). The drug development process (Figure 2.1) is largely an elaborate and expensive filter to eliminate the unsuitable compounds.

The largest part of time and effort of drug development is spent on trials to determine whether the drug candidate meets these criteria of bioavailability, efficacy and safety. Since it is better that a drug candidate should fail early in this process instead of late, the pharmaceutical industry generally strives for the "fail fast, fail cheap" ideal.

To fail fast and cheaply, it is essential to have fast, cheap methods of determining whether the drug candidate does or does not have suitable properties to be a drug. Computational methods are ideal for this goal, since they could replace expensive biological tests and do not even need the synthesis of the drug candidate. Additionally, computers are also applied to increase the input of the pipeline by suggesting alternative drug candidates.

One of the classes of methods used in the pharmaceutical industry for these purposes is evolutionary algorithms, which seems especially appropriate since drug design is largely survival of the fittest compound. This review will focus on the diverse evolutionary algorithms applied to the problems of drug design. We will first introduce the concept of evolutionary algorithms.

Evolutionary algorithms

Evolutionary Computation is the term for a subfield of Natural Computing that has emerged already in the 1960s from the idea to use principles of natural evolution as a paradigm for solving search and optimization problems in high-dimensional combinatorial or continuous search spaces. The algorithms within this field are commonly called evolutionary algorithms, the most widely known instances being genetic algorithms (Holland 1975, Goldberg 1989, Goldberg 2002)[2], genetic programming (Koza 1992, Koza et al., 2003), evolution strategies (Rechenberg 1973,

[2] It should be noted that many evolutionary algorithms described in this review are called "genetic algorithms" by their authors, even though they do not follow Holland's original scheme at all. This misleading nomenclature might decrease in the future, however meanwhile the reader is advised when searching literature on evolutionary algorithms in the area of drug design to supplement his database queries regarding "evolutionary algorithms" with searches for "genetic algorithms."

Rechenberg 1994, Schwefel 1977, Schwefel 1995), and evolutionary programming (Fogel *et al.* 1966, Fogel 1995). A detailed introduction to all these algorithms can be found e.g. in the Handbook of Evolutionary Computation (Bäck *et al.*, 2000).

Evolutionary Computation today is a very active field involving fundamental research as well as a variety of applications in areas ranging from data analysis and machine learning to business processes, logistics and scheduling, technical engineering, and of course drug design, the topic of this article. Across all these fields, evolutionary algorithms have convinced practicians by their results on hard optimization problems, and thus became quite popular today. This introductory section on evolutionary algorithms aims at giving the reader a first impression of their fundamental working principles, without going into details of the variety of implementations available today. The interested reader is referred to the literature for in-depth information.

The general working principle of all instances of evolutionary algorithms is based on a program loop that involves implementations of the operators mutation, recombination, selection, and fitness evaluation on a set of candidate solutions (often called a population P(t) of individuals at generation t) for a given problem. This general evolutionary loop is shown in the following algorithm.

Algorithm 2.1: Simplified abstract evolutionary algorithm.

```
t := 0;
initialize P(t);
evaluate P(t);
while not terminate(P(t)) do
   P'(t)  := select_I(P(t));
   P''(t) := recombine(P'(t));
   P'''(t) := mutate(P''(t));
   Evaluate(P'''(t));
   P(t+1) := select_II(P'''(t) ∪ P(t));
   t := t+1;
od;
return(best(P(t));
```

In this general setting, mutation corresponds to a modification of a single candidate solution, typically with a preference for small variations over large variations. Recombination (called “crossover” by some investigators) corresponds to an exchange of components between two or more candidate solutions. Selection drives the evolutionary process towards populations of increasing average fitness by preferring better candidate solutions to proliferate with higher probability to the next generation than worse candidate solutions (this can be done probabilistically like in genetic algorithms, or deterministically like in evolution strategies). Selection can be used either before recombination as a kind of sexual selection operator preffering better individuals to generate more copies before recombination occurs, or as an environmental selection operator after fitness evaluation to reduce population sizes by removing worse individuals from the population. This second selection operator can also take the original population P(t) into account, thus allowing the algorithm to always keep the best individuals in the population (which is called an elitist strategy assuring that fitness values do not get worse from one generation to the next). By evaluation, often called more specifically fitness evaluation, the calculation of a measure of goodness associated with candidate solutions is meant, i.e., the fitness function corresponds to the objective function of the optimization problem $Y = f(x_1,\dots,x_n) \rightarrow \min (\max)$ at hand (minimization and maximization are equivalent problems), where $f: M \rightarrow R$ maps candidate solutions defined over a search space M into real-valued (usually scalar) measures of goodness.

Evolutionary algorithms offer several advantages over conventional optimization methods, as they can deal with various sets of structures for the search space M, they are direct optimization methods which do not require additional information except the objective function value $f(x_1,\dots,x_n)$ (i.e., no first or second order derivatives in continuous search spaces), they can deal with multimodal optimization problems (i.e., problems where many local optima exist where the search can get trapped into a suboptimal solution), and they can also deal with additional problems such as discontinuities of the search space, noisy objective function values or dynamically changing problem characteristics.

The candidate solutions (elements of the search space M) to an optimization problem can have arbitrary datastructures. However, certain kinds of candidate solution structures are popular, such as binary or discrete valued vectors, as often associated with the concept of a genetic algorithm, real-valued vectors, as often associated with evolution strategies or evolutionary programming, or parse trees in a functional language such as LISP, as often associated with genetic programming. The differences between these representational instances of evolutionary algorithms have become blurred since 1990, however, such that state-of-the-art evolutionary algorithms often use concepts from several of the pure historical instances together in an implementation that is tailor-made for a particular application problem. Also, many mixed representations are used to solve challenging problems defined in more complex search spaces, e.g., mixed-integer nonlinear optimization problems. Expansions to new search spaces including graph-based representations naturally imply the potential application of evolutionary algorithms to drug design or molecule optimization problems.

Scope and limitations of this review

This review focuses on the stage of drug design in which the drug molecule is designed. Therefore applications of evolutionary algorithms that are also important but preliminary to this stage, such as protein folding prediction and elucidation of protein structure, are not discussed here. The interested reader is referred to other literature, such as the compilation of reviews edited by Clark (2000).

The articles discussed in this review were published in the period from 1993 to 2004. Our primary criterion for selection was diversity in application and method, not recency. However, most of the articles (44 of 54) are from the period 1998 to 2004, since the application of evolutionary algorithms in drug design only started to bloom in the mid-nineties.

Due to our focus on design of drug molecules, the distribution of literature references is skewed towards chemical literature. The three major journals discussing cheminformatics and computational chemistry contributed 38 articles, journals in medicinal chemistry and general chemistry 13 articles, and computer science-based conference proceedings only 3 articles. This is however not an exhaustive compilation of existing literature, and the interested reader will be able to find more relevant articles in the (medicinal) chemical and computer science literature.

We hope that this review will help the reader gain insight in the problems of drug design and the diverse kinds of evolutionary algorithms applied so far, and enable him or her to read or perform additional research in this area with a wider perspective and more understanding. We hope that in this way the review can contribute to the further development of computational methods that help solve the problems of drug design, and enable researchers to apply the power and processing capabilities of the computer to enhance human health.

2. Evolutionary algorithms in the design of molecule libraries

To find a lead compound for further drug design a set of compounds (called a library) can be tested for the desired biological activity. A good library should have good efficiency and good effectiveness: it should be so small that the cost of testing it is as low as possible, yet be so large that the chances of finding a suitable lead compound are sufficiently high.

Choosing the contents of the library rationally instead of randomly can enhance the efficiency and effectiveness: since compounds with similar structures usually have similar activities, a library consisting of compounds that are very dissimilar to each other will require fewer compounds to cover as much of the "biological activity" space.

Another criterion is drug-likeness: drug molecules must have certain properties to work (for example, have a weight of under 500 atomic mass units to be taken up by the body (Lipinski *et al.*, 1997)), so such constraints can also be enforced during the design of the library.

More advanced criteria can also be applied, if more information is available: if the structure of either a ligand (a compound that binds to the receptor) or of the target receptor itself is known, one could select those compounds which look like the ligand or fit into the receptor, instead of the most diverse ones; this is called *targeting*.

The most popular method of creating the compounds of the molecule libraries is combinatorial chemistry: a number of compounds of group A, which all have a certain common reactive group, is combined with a number of compounds of group B, which have another common reactive group that can react with the reactive group of A (Figure 2.2).

In this way, N+M reactants are converted into N*M products. Higher dimensions of synthesis (N+M+P reagents give N*M*P products) can also be applied. Since there are many available reactants and multiple reaction steps can be applied, the number of potential compounds is much larger than the number that is practically and economically feasible to make and test. For this reason, selection of the reagents to be used or of the products to be made is very important. This has turned out to be a promising application for evolutionary algorithms. We will now discuss a number of these applications.

The first application we would like to discuss is the program SELECT (Gillet *et al.*, 1999). SELECT has the objective to construct a general library, the compounds of which should both be diverse and druglike. Testing this idea on virtual amide

(100x100) and thiazoline-2-imide (12x99x54) libraries, the goal is to choose that sublibrary which has highest diversity, and whose molecules have a similar property distribution as known drugs (so if 15% of drug molecules have 3 rotatable bonds, 15% of library molecules should have 3 rotatable bonds too). The desired sizes of the libraries were 20x20 and 8x40x20, respectively.

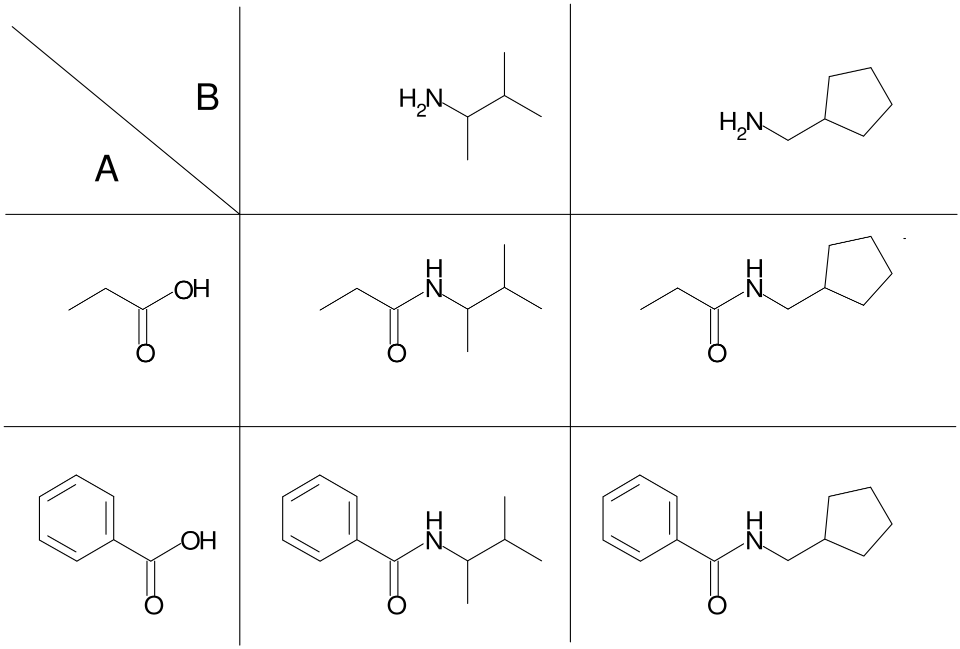


Figure 2.2: A simple combinatorial library.

The data structures representing the candidate solutions (these data structures are commonly called "chromosomes" in the field of evolutionary algorithms, see also the glossary) were vectors with as length the number of reagents for the target library, consisting of the identification numbers of the reagents used. Each set of reagents was assigned to a separate partition of the chromosome. Single point mutation and single point crossover (crossover only occurred in one randomly chosen partition) were applied. The population size was 50.

The diversity of the library was determined by first calculating a chemical fingerprint of each molecule, a vector of bits, and summing the differences between all pairs of vectors.

In the case of the amide library, with diversity as fitness criterion, convergence was reached after about 1000 iterations, with a very reproducible optimum (mean 0.595, standard deviation 0.001)- a clear improvement over the diversity of randomly constructed libraries (mean 0.508, standard deviation 0.015). However, it turned out that taking drug-likeness as additional criterion decreased the diversity, and that depending on the relative weights of the criteria, different solutions were found. This task of minimizing diversity while maximizing drug-likeness could be viewed as a multiple criteria decision making task.

Since manually adjusting the weights to create different solutions is inelegant and impractical, the authors subsequently developed an extension of SELECT, called MoSELECT (Gillet *et al.*, 2002). The goal of this program is to find a set of solutions, each solution so that no other solution in the set is equal or superior to it in all respects (the solution is nondominated, or “Pareto optimal”; see Figure 2.3).

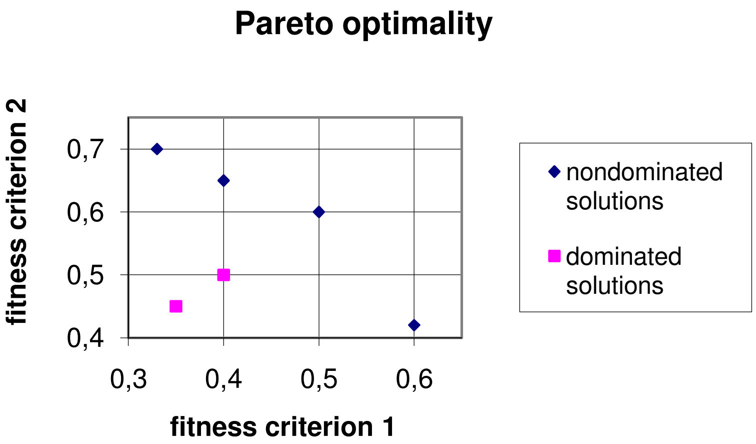


Figure 2.3: Pareto optimality. In this example, both fitness criteria are to be maximized. A solution is dominated if there exists another solution that has equal or better scores on all criteria. for example (0.5 , 0.6) dominates (0.4 , 0.5) because 0.5>0.4 and 0.6>0.5. However, (0.5 , 0.6) does not dominate (0.4 , 0.65) because 0.5>0.4 but 0.6<0.65.

This algorithm can perform multi-objective optimization by Pareto-ranking the chromosomes: nondominated chromosomes get rank 0, chromosomes which are dominated by one other chromosome get rank 1, etcetera, after which roulette wheel selection is applied, a common implementation of the “select-I” function in algorithm 1.

Information about the mechanism of this selection method can be found in the glossary. This Pareto-ranking approach results in many nondominating solutions found; using 2 fitness criteria resulted in 31 nondominated solutions (in a population of 50), while increasing the number of criteria to 5 and the population size to 200 gave 188 nondominated solutions. However, speciation was observed so niching (forbidding the algorithm to create new solutions which are similar to already found solutions) was applied to ensure diversity. This reduced the number of solutions to 24, but made them more different. (Evolutionary algorithms have also been used for finding sets of Pareto-optimal solutions in other contexts, in which they turned out to be quite efficient, one advantage of the evolutionary algorithms being that they can find a set within a single run – see Deb (2001) for an in-depth coverage of the topic).

While diversity is a very desirable characteristic in a general purpose library, libraries can also be designed to discover a lead to a specific target. Sheridan *et al.* (2000) designed a combinatorial library of molecules built out of three fragments. There were 5321 fragments possible for the first part of the molecule, 1030 fragments for the middle of the molecule and 2851 available fragments for the third part of the molecule. Since synthesizing 15 billion compounds would be prohibitively expensive and time consuming, the authors desired to design small libraries (100-125 compounds) of molecules that looked most promising. They wanted to create libraries of compounds that look like angiotensin-II antagonists (a "2D-criterion", which only uses information on which atoms are connected to which other atoms) as well as libraries of compounds that fit in the active site of the protein stromelysin-1 (a "3D-criterion", which must know and manipulate the three-dimensional structure of the molecule).

Furthermore, Sheridan tested whether evolving a 5x5x5-library yielded results as good as evolving a library of one hundred separate molecules, addressing in this way the question whether the benefit of needing fewer different reagents by the 5x5x5 library is offset by a decrease in library quality. In the experiments the 2D-criteria were as well achieved, on average, by the library-based as by the molecule-based runs, be it at much more computational cost (molecule based: <20 minutes; library based: about 120 h). 3D-Fitness evaluation took over 120 times as long as 2D evaluation, so library-based runs could not be performed using 3D-fitness criteria. However, the library created of the 5+5+5 most frequent fragments in the molecule-based optimization had a considerably lower score than the original library. While for "2D"-criteria the whole is approximately "the sum of its parts", in the more realistic 3D fitness function this approximation no longer holds. The fitness landscape is probably much more rugged, i.e. contains many more local optima in which a solution can become trapped. It is interesting to note, however, that despite this ruggedness the number of generations needed for convergence was approximately the same for 2D and 3D, namely 10-20 generations.

A method that combines targeting and diversity is to use a known molecule as a template structure. Liu *et al.* (1998) generated two sets of compounds, the first set based on a benzodiazepine template (see figure 2.4) and the second on a template derived from the (-)-huperzine A molecule.

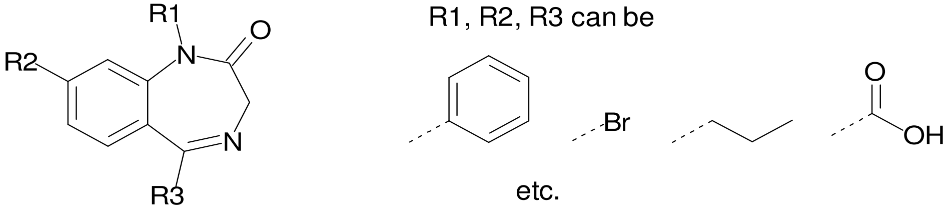


Figure 2.4: Template-based (virtual) library design.

A library of 73 fragments was used to fill the open positions on the template. A population of one hundred molecules was generated by attaching randomly chosen groups to the template molecule. After this, the diversity of the population was determined by converting the 3D-structure of the electronic field around the molecules into sets of vectors, and measuring the dissimilarity between the vectors of the different molecules. Crossover was implemented by exchanging groups of two molecules at the same template position, mutation by having fragments exchange template positions or by replacing one of the fragments. After a short run (10 generations) convergence was reached. No data were provided on the reproducibility of the run.

The (-)-huperzine A library was generated in the same way as that of the benzodiazepine analogs. Subsequently some of the proposed structures were synthesized. One of them was found to have a higher binding affinity to the target than the lead itself, showing that the method is effective.

From the foregoing it is clear that evolutionary algorithms can optimize the diversity and other properties of combinatorial libraries. However, related experiments by Bravi *et al.* (2000) have given some interesting insights into the structure of the search space. Bravi *et al.* investigated if one could not only determine the optimal library composition, but also the optimal library size. Filters were used to select the most druglike compounds from a virtual library of 13x41x59 (of which 16% turned out to be good). To synthesize all druglike molecules using a combinatorial library would require a library of 12x39x49; using this in combinatorial chemistry would however generate about 23000 compounds, of which 78% would be non-druglike. How to find a balance between efficiency (how large a part of the combinatorial library consists of desirable structures) and effectiveness (how large a part of all good structures are contained by the sublibrary)? Bravi's program PLUMS used an algorithm that evenly weighed these two factors and designed a library that still contained 86% of all good molecules, with only 37% undesirable products.

The method Bravi used was based on iterative removal of the component whose removal produced a library with an optimum score. His results were as good as those of the GA to which he compared it, as long as PLUMS followed alternative parallel paths if there was no preference for removal. This suggests that the fitness landscape is not very rugged for this problem, and that an iterative method might replace a GA in such cases. However, a simpler method (monomer frequency analysis (MFA), which assumes that the best library is built from the fragments that are most frequent in the good compounds) failed to find this optimum. Analysis of the results showed that how often a fragment occurs in a good library is less important than how often it occurs with other good fragments. However, a subsequently designed dynamic version of MFA that iteratively chooses the best compounds of each set of reactants until convergence is reached, did find the global optimum.

Does this mean that evolutionary algorithms are not needed in library design? This is not very likely, since using more advanced 3D-fitness functions seems to make the fitness landscape more rugged. A simple method like PLUMS will get stuck in a local optimum more easily, especially if the building blocks of the library must be selected among thousands instead of dozens of reactants. However, iterative methods like PLUMS and MFA are good demonstrations of the power of simple solutions appropriately applied.

Conclusion

Several experiments have been performed using evolutionary algorithms in library design, to create libraries to satisfy many different objectives such as diversity, targeting and drug-likeness. While improvement of the libraries with respect to the fitness criteria is clearly seen in these experiments, and reproducibility seems fair enough, the major current challenges lie in refining the fitness criteria to accurately reflect the demands of drug development.

The diversity in the diversity criteria themselves suggests that more systematic attention to this problem might be worthwhile, and the great computational cost of more advanced (docking) criteria of target selection are still troublesome in more refined applications. Also the drug-likeness criterion might need revision.

Libraries are designed to find lead molecules, which usually grow in size during drug development to satisfy additional criteria. In many cases this may generate molecules that are too large to be drug-like. Screening the “drug-like” larger molecules for biological activity has a lower chance of success than screening smaller molecules, since large molecules have a smaller probability to fit in the space of the active site than small molecules (Hann *et al.*, 2001). Therefore, it would be more valuable to evolve libraries with the criterion of lead-likeness. However, libraries of leads are currently not available, while libraries of drugs are. Unless calculations correct for the too high molecular weight and lipophilicity of drug-like compounds, “drug like” library design will probably produce suboptimal compounds.

A second development is the use of several conflicting criteria simultaneously in library design, of which the Pareto optimality by Gillet *et al.* (2002) and the prefiltering by Bravi *et al.* (2000) are examples. While certainly interesting, the problem of choosing the right weights by the user is now shifted to selecting the right nondominant set. Weighing must be done sooner or later. It is a good beginning, but further measures (probably based on existing knowledge of drug development and probability theory) are needed to find a better way of weighing the weights.

An application which has not been discussed in these articles is selecting compounds from a non-combinatorial library. This will become more important as proprietary compound collections of pharmaceutical companies grow and more compounds are made available by external suppliers. The disadvantages of combinatorial chemistry (generally too large and lipophilic molecules, failing reactions, etc.) could prompt using evolutionary algorithms to select a targeted or diverse test set out of tens of thousands of compounds that are available. This will be an interesting and important challenge.

Computationally, the different evolutionary algorithms can doubtlessly be improved by incorporating more domain knowledge. However, since the computational cost of most applications discussed is acceptable and performance is good, the relatively simple current algorithms may be preferred over more advanced versions. Comparisons with deterministic methods (Bravi *et al.*, 2000) indicate that evolutionary algorithms can be applied quite well to the problem of library design. Although competing methods can also satisfy the designer’s needs (Agrafiotis, 2002), evolutionary algorithms, perhaps with some small modifications, are very likely to become the standard method in library design.

3. Evolutionary algorithms in conformational analysis

A molecule is a three-dimensional entity consisting of atoms connected by bonds. Though the movement of the individual atoms is restricted by the bonds, most molecules can assume different shapes by bond stretching, by angle bending and, most importantly, by rotating parts of the molecule around single bonds (see Figure 2.5). The amount by which a bond is rotated (varying between 0 and 360 degrees) is called its torsion angle.

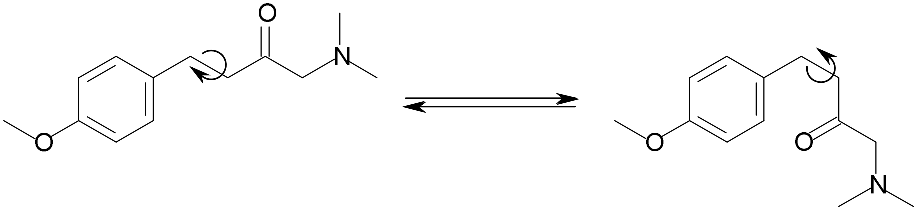


Figure 2.5: Change in conformation by rotation around a bond.

Conformational analysis, the generation and comparison of different conformations of a molecule, is an important part of medicinal chemistry. This is because the properties of a molecule are partially determined by the shape or range of shapes it can assume. Conformational analysis usually has two goals. The first and most common goal is to find the conformation of minimal energy, the “global minimum”. The energies of all other conformations (which correspond to their chance of occurring in nature) should be taken relative to the energy of this global minimum. This is especially important when a molecule is docked as a ligand into the active site of a receptor (see section 6). The increase in energy of the docked molecule relative to its minimum gives information on the true binding energy and therefore the likeliness that the docking is correct. The second goal of conformational analysis is to obtain a group of diverse yet energetically feasible conformations for virtual screening to address the issue whether the molecule or one of its good conformations fits a certain required pattern, a so-called pharmacophore.

Since bonds can be rotated over the entire range of 360 degrees the number of conformations of the molecule is in theory infinite. However, many conformations are so similar that conformational analysis usually takes a minimal step size of 15-30 degrees. Unfortunately, allowing *n* different torsion angles for *m* rotatable bonds each will give n^m possible conformations; for a flexible drug molecule like orphenadrine (which has six rotatable bonds), conformational analysis with a resolution of 15 degrees would produce 1.9 x 10^8 conformations. Systematic search is infeasible in these cases, and heuristic algorithms, among which evolutionary algorithms, are applied.

An excellent example of a genetic algorithm applied to finding the conformation of minimal energy is the work of Nair and Goodman (1998). Nair and Goodman applied the genetic algorithm to linear molecules of carbon atoms (alkanes), and took the torsion angles as genes. After random generation of the population, crossover was performed followed by mutation. Subsequently the new structures were minimized with a local optimizer and their optimized conformations written back into their genes (so-called Lamarckian evolution), and the new generation was chosen from the pool of parents+children by roulette wheel selection on their energies, which were weighted with a Bolzmann factor that determined the penalty for higher energy. This process was repeated for a fixed number of generations.

The genetic algorithm found several minima for the chains of 6, 18 and 39 carbon atoms. The next, most interesting challenge was finding the optimal energy of PM-toxin A, a long, approximately linear molecule (33 carbon atoms). This was tackled by first optimizing a 33-atom alkane, listing the several thousands of low-energy conformations found. Subsequently the branching groups were added and the resulting structures locally optimized. A minimum of less than -100 kJ/mol was found. A Monte Carlo search, using the same amount of structure optimizations, found a minimum of only –78 kJ/mol. Furthermore, the GA found 168 conformations with an energy below –70 kJ/mol, the Monte Carlo approach only two.

It is interesting to note that the more complex and flexible the molecule becomes, the more minima of approximately equal energy can be found. Since the energy of the global optimum is much more important than the conformation of the global optimum and dozens of conformations give the approximately good result, knowing the “best” answer is relatively unimportant. This makes stochastic algorithms like evolutionary algorithms even more useful in this situation.

Jin *et al.* (1999) analysed the pentapeptide [Met]-enkephalin, which has 24 torsion angles. Three different versions of their program GAP were used: GAP 1.0, GAP 2.0 and GAP 3.0. In GAP 1.0 a uniform crossover was used together with a diversity operator that mutated a child structure if more than half of its angles differed by less than 5 degrees from its parent structures. GAP 2.0 included a three-parent crossover (two parents are crossed, their product is crossed with the third parent), and GAP 3.0 has a “population splitting scheme”, which only allows crossover of individuals in different populations. The offspring was generated by crossover and subsequent mutation. After these steps, parents and offspring were taken together, the lowest half (50 conformations) was selected as the next generation, and after 1000 generations the runs were stopped. In this case, the minimum found was about 3 kcal/mol higher than the one found by a Monte Carlo method.

Since other experiences with GA/MC comparisons like those of Nair and Goodman (1998) and Tufféry *et al.* (1993) found the genetic algorithm to be superior to Monte Carlo, especially when optimizing large systems like proteins, the authors analysed their algorithm. By measuring the search space coverage it was found that, surprisingly, higher mutation rates led to *lower* coverage. This suggests that most mutations are so harmful that they are rapidly selected out by the strict fitness criterion (best half), and the next generation consists mainly of unmodified “parent” conformations, which tends to prevent departure from local minima and restricts the search space covered.

For certain purposes, not a single low-energy conformation is needed, but a set of low-energy conformations that differ as much from each other as possible. These conformations can be used for e.g. pharmacophore screening or as starting conformations for docking. Mekenyan *et al.* (1999) designed a GA for optimizing the diversity in a population of conformations. The fitness criterion was a diversity criterion that measured how bad the best possible superposition of two conformations was (in root mean square distance between corresponding atoms). The score of the individual was the average dissimilarity to the other members in the population.

Next to the traditional torsion angles Mekenyan included the flexibility of rings by allowing free ring corners (atoms that were part of only one ring) to flip, and storing the flipped/unflipped information in the chromosome too. This may be very valuable for complex molecules that often contain flexible rings.

Mutation was performed and followed by crossover. If the children were energetically inadmissible or too similar to already present conformations, they were discarded. If N_c viable children were found within a certain number of tries, the most diverse subset of size N_p was selected from the total pool of N_c+N_p conformations. The evolution was stopped if fewer than N_c viable children had been produced within the specified number of tries.

Mekenyan experimented with different settings of the population size and the number of children. The runs did not seem very reproducible and in most cases were stuck in local optima. The general conclusion was that the ratio between the number of parents and the number of children N_p/N_c is very important. If N_p/N_c is lower, convergence is reached faster and more of the search space is covered, but if it is higher, runs are more reproducible.

Thinking more theoretically about the quality of evolutionary algorithms, Wehrens *et al.* (1998) considered that only taking the value of the best individual to judge an evolutionary algorithm is somewhat limited, and proposed additional criteria: reproducibility and coverage of the search space. The authors describe the application and implementation of these criteria in the case of the conformational analysis of *N*,*N*-dimethyl-*N'*-4-phenylbutylmalonamide.

This compound has 7 rotatable bonds, the torsion angles of which form the genes of the chromosome. A population of size 50 was used for a run of 100 generations. Tournament selection was performed with tournament sizes varying from 2 to 10. Crossover rate was 0.8 with uniform crossover applied. In the experiments, several parameters were varied, mainly to investigate the influence of the “sharing” operator. If the root mean square difference between the torsion angles of the child and parent conformations is less than the sharing distance, a randomly selected torsion angle of the child will get a random twist between 0 and a fixed number of degrees called the “sharing offset”.

Coverage was measured by dividing the search space into hypercubes (hypercube size of 90 degrees, so there are 4^7 hypercubes which can be visited in the search space). About 10% of the search space was visited using a GA without sharing, 30% with sharing, 77% by random search. So while sharing increases coverage, selection pressure decreases it. A tournament size 10 instead of 2 further decreased the coverage, be it slightly.

The second criterion of coverage was how many clusters of low energy were found using different parameter settings. In this case this was 6 to 14 clusters for the genetic algorithm, 0 for random search.

Another criterion, reproducibility, was measured in two ways: the first way was to count the number of clusters in common between two runs, the second was projecting all conformations into the 7 dimensional “torsional” space and determine the principal components. The ratio of the overlap of the principal components of the different runs of one setting and those of another gave the reproducibility.

As the authors note, their criteria may also be used for other applications of genetic algorithms. Though some of their ideas seem useful, they have, considering the subsequent literature, not yet been widely applied by other researchers.

Conclusions

Evolutionary algorithms have been applied to conformational analysis with some good results. While there are some experiments that indicate that the method of "directed tweak" is slightly superior in conformational searches (Clark *et al.*, 1994) evolutionary algorithms are more versatile: they can search for sets that are diverse, as well as pursue multiple objectives. Next to seeking the most suitable mutations and crossover methods and optimizing the parameters, there are some other interesting points that could justify further research. The first question is how one could incorporate molecular mechanics such as the deformations of rings in the evolutionary algorithm. Secondly, almost all energies are now calculated for molecules in a vacuum, yet the relevant energies for biological molecules are those in solution. One should carefully compare the vacuum results with those calculated using modern force fields that include water to check whether and when this approximation is allowed. A third item, which is growing in importance, is the application of conformational analysis to larger molecules, especially proteins.

As our understanding of biology increases, molecular movement and conformations will be able to shed light on the dynamic properties of chemical and biological systems. Conformation analysis will be important to determine the "4D"-descriptors, which describe the possible changes of the molecule over space and time. Evolutionary algorithms, with their flexibility and possibilities to optimize systems in which the elements depend on each other, as is the case in conformations, will probably continue to play an interesting and important role in the development of this field.

4. Evolutionary algorithms in molecule superposition and pharmacophore detection

If two molecules bind to the same receptor, can one deduce from this information which other molecules will bind? The traditional way of solving this problem is by comparing the structures of the active molecules: one superimposes the molecules onto each other to detect the similarities. If they have the same kinds of atoms in the same relative positions, those may be important. Out of this superposition, features which might be important for activity are postulated, and their relative 3D-orientation constitutes the active pattern, or *pharmacophore*.

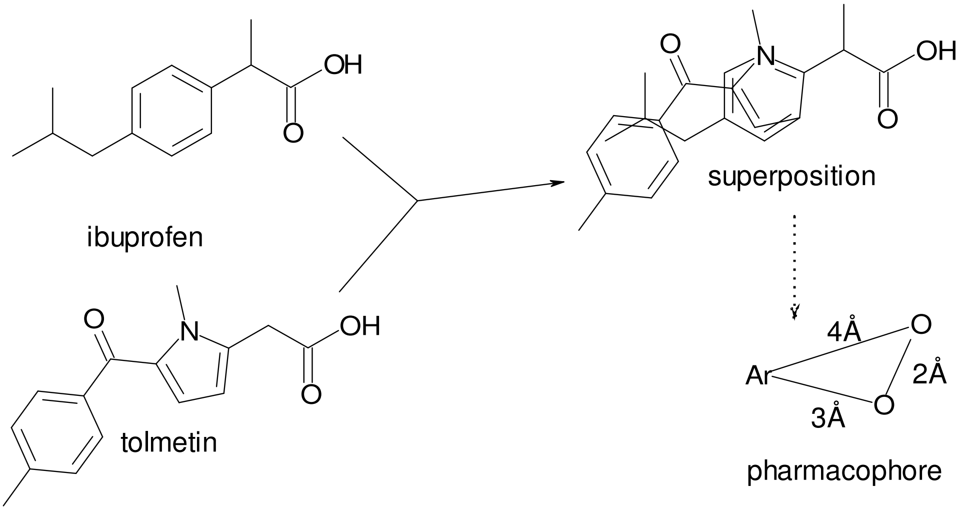


Figure 2.6: Molecule superposition and pharmacophore detection. "Ar" stands for aromatic center, 1 Å is 0.1 nm.

This entire process of superposition and assignment of pharmacophoric points is called pharmacophore detection (see figure 2.6).

There are two fundamental difficulties in molecule superposition and pharmacophore detection. The first is the definition of a good superposition. There are at least three possible criteria:

1) In a good superposition both molecules have low energies; their conformations have energies at or close to the global minimum.
2) In a good superposition the volumes of the molecules overlap optimally, which means that they fit in the receptor in about the same space of the active site.
3) In a good superposition, the most important atoms/parts should overlap best, the other parts of the molecule are relatively unimportant.

In fact, all these factors seem to play a role. Ultimately the criteria a molecule has to fulfill to be active are determined by the three-dimensional structure of the receptor, but unfortunately that structure is generally not known. Nevertheless, a method that finds high similarity of whatever kind between various active molecules and does not match inactive molecules would certainly be promising.

The second problem in molecule superposition is the combinatorial explosion: most molecules can assume thousands of conformations, so searching for the best overlap of two molecules or more by a systematic search method quickly becomes infeasible. It is no surprise that evolutionary algorithms have been applied in order to help to solve this problem.

An early example of a genetic algorithm to superpose molecules and detect pharmacophores is given by Payne and Glen (1993). The chromosomes representing the molecules are bit strings, the first elements give the 3D-coordinates for the location and the orientation of the molecule leading to 6 degrees of freedom. They are followed by genes for each bond that can be rotated and for each ring corner that can be flipped.

In some cases the fitness criterion was how well a molecule obeyed certain distance constraints, i.e. selected groups in the molecule or of different molecules should be at a certain distance from each other. Overlap constraints, i.e. overlapping another molecule as much as possible, and spherical constraints were also used. The latter constraint is defined by a sphere drawn around the molecule, the surface points of which have values representing the distance from the sphere surface point to the molecule's surface point directly beneath it or the charge on that surface point. The total fitness was a weighed sum of the several fitness functions that were appropriate for the situation. Chromosomes were represented as bit strings, the mutation was bit-flip mutation and one-point crossover was used.

Several problems were tackled with this algorithm: finding the conformation of a molecule which obeyed certain distance restraints, elucidating a pharmacophore, fitting a molecule onto itself, and fitting different molecules of a similar biological activity onto each other.

It turned out that some of the problems were relatively easy to solve using the genetic algorithm. If there is a fixed set of constraints or a rigid template molecule like morphine the evolution reaches convergence (in runs of 300 generations of 1000 molecules). If however flexible molecules have to be fitted onto each other, the "moving target" makes convergence very awkward. However, when an intermediate step was added in which the conformers were rigidly fitted onto each other the time spent by the genetic algorithm was reduced from 10 days to 9 minutes!

All in all, the program described seemed to do its job fairly well, though greater degrees of freedom clearly gave it so much trouble that optimization became difficult. A last problem is that when some regions of a molecule are important to receptor binding and others are not, a sphere model might not be a very suitable means for finding the part of the molecules that are similar. This is due to the fact that the differences in the other parts may drown out the similarities unless one has large data sets. Moreover, superpositions of the many molecules of those large data sets themselves might lead to poor convergence.

Superposition of molecules has often the goal of finding a pharmacophore. Holliday and Willett (1997) wanted to use a genetic algorithm to find a group of pharmacophore points (in their case: N and O atoms) in a 3D-arrangement present in all molecules with a certain biological activity.

Their original genetic algorithm proved to be too slow, but the authors found that performance could be improved by splitting it into two smaller genetic algorithms: one to find sets of corresponding atoms in the different molecules, a second to combine these sets into the smallest possible superset.

The first genetic algorithm uses chromosomes of length $n \times m$, where n is the number of molecules and m a user-defined number of atoms that has to be found per molecule. Crossover is performed on the border between molecules, mutation replaces an atom by another atom of the same molecule. If the atoms in the chromosome of two different molecules have the same types and approximately the same distances to each other, the second set of atoms is “fused” with the first. The evolutionary process thus results in a chromosome grouped in a few different clusters of molecules, the molecules of each cluster containing identical atoms in a common geometric pattern.

The second genetic algorithm uses the collection of patterns found by the first algorithm and attempts to find a superset which contains all of them. The chromosome here is a list of the 3D coordinates of the several points. The second algorithm can add, move or remove points in this 3D-arrangement and continues until every molecule in the set has at least m points (the value of m specified by the user) in common with the superset, within a certain tolerance range. The second genetic algorithm uses clique-finding algorithms to speed up this process.

The program was tested on five data sets of 10-19 biologically active compounds. In most cases, 3 or 4 point subsets common to all compounds were found, thus indicating the effectiveness of the method.

However, the authors add that their program should be developed further. Next to the nitrogen and oxygen atoms there may be other important elements in a pharmacophore such as a phenyl group (see also Figure 2.6). Additionally, most ligands are flexible and their active conformation is not known; therefore the genetic algorithm should either work on a good superposition (in which case it would not give much useful extra information) or take the flexibility of the molecules into account.

This issue of flexibility was addressed by Handschuh *et al.* (1998) who used a genetic algorithm to superpose flexible molecules. This superposition was again based on atom superposition, but in this method the superposed atoms did not have to be of the same type.

The authors recognized that a good superposition of molecules should satisfy conflicting demands. Although as many atoms as possible of the two molecules should be matched, matching too many atoms will result in a worse fit. For this reason Pareto optimization was used to obtain alternative solutions.

The computer program fitted only two structures simultaneously; each individual consisted of a chromosome containing the information of both molecules. The chromosome consisted of two parts, which contained the "match pairs" (which atoms of structure one were fitted onto atoms of structure two) and the torsion angles of the molecules, respectively.

A population of 100 molecule-pairs was created and subsequently evolved. Mutation and crossover in the torsional part was straightforward and mutation in the match part replaced or deleted atom matches. Crossover in the match part was implemented by choosing two match lists of equal length in the parents and appending them to the end of the other parent's match list, removing duplicate atom matches in the original parent. Interesting was the inclusion of two "knowledge augmented" operators, "creep" and "crunch", which added atom pairs to or removed them from the match list based on their distance in the current superposition. These operators improved the final results substantially, since much closer fits of 0.05-0.2Å were obtained instead of root mean square scores of 0.6-1.0Å.

Another innovation somewhat similar to the speedup described by Payne and Glen (1993) was the use of the directed tweak method to adapt the torsion angles of the match after each individual was generated. This was however not Lamarckian since the genes were not changed and the matching procedure was only used to determine the fitness value. Instead restricted tournament selection was used. Here one solution competed against the solution most similar to itself from a random subset of the population. The winner was copied into the next population. This selection method was chosen in order to conserve diversity.

Handschuh *et al.* applied the genetic algorithm to overlaying several angiotensin II antagonists, with good results in that overlays of 10-20 atoms were reached with low root mean square values (<1Å). Additionally, a known angiotensin II pharmacophore was found. These results indicate that the method is quite promising. However, some problems of pharmacophore finding remain difficult to solve, even with a method as advanced as this one. A true choice about whether molecules A and B overlap best in overlap 1 or 2 can only be made if it can be determined whether the identity of the atoms really matters (Figure 2.7). In some cases it will, in others it won't, such that there may be other objectives to add to the Pareto fitness.

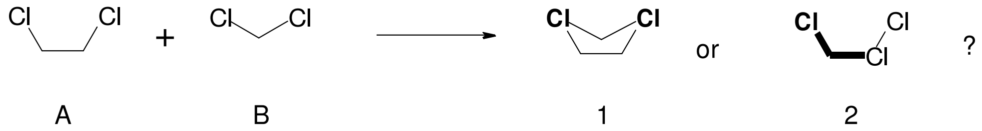


Figure 2.7: Which superposition is best?

Conclusion

There are several different kinds of molecule superposition. Superposing the shape and charge fields of two dissimilar molecules, superposing the most important atoms, or superposing all atoms are all options, but which one is "correct" or "better"? Probably much depends on the protein and the set of ligands. The existence of different criteria seems to indicate that superposing molecules is a multi-objective problem, with the different weights reflecting the one true objective of how well the superposed molecules occupy the "superposed" space when binding to the receptor. Comparison with experimental data such as crystal structures would greatly help to test, validate and optimize the different methods. Pending that, extra calculations of for example the energy of the ligands may help to make a choice between different superpositions.

Also, it would be worthwhile to extend the pharmacophore models with known inactive compounds that are similar in structure to the active molecules and study if these fit or not. This may yield information on criteria for internal energy of the superimposed conformation or information about the "excluded volume", the parts of the molecule that the receptor cannot accommodate.

Thirdly, there is the problem of superposing larger sets of compounds. The extra information gained by including more compounds is probably useful, but an optimal multiple superposition is much more difficult to find. Overlaying two molecules is quite standard, but what to do if there are more? While Handschuh et al. (1998) found that the order of superposition of their four compounds did not influence the results, it seems likely that a naive evolutionary algorithm would fail if it would attempt to overlay more than ten structures simultaneously. Sequential overlap of many compounds will probably yield local minima, especially since there may be different "best" superpositions according to the Pareto optimality criteria. Handling large datasets, especially truly large data sets on which one can apply statistics, seems to become possible (Chen *et al.*, 1999). It is still unclear yet whether this will be the final answer due to the necessarily limited number of conformations and pharmacophoric points used by such methods.

Lastly, in several cases there may be more than one active site on the receptor, or the binding site is so large that not all molecules will necessarily share the same volume. Discovering that there are several different pharmacophores in this case will be a challenging test for any superposition method.

All in all, evolutionary algorithms have led to valuable software for molecule superposition and pharmacophore detection. Still the field of molecule superposition does not have the answers yet for handling more than two molecules and choosing between different superpositions. While there are also non-evolutionary methods for pharmacophore detection (Chen *et al.*, 1999; Ting *et al.*, 2000), it is very likely that evolutionary algorithms will continue to be applied.

5. Evolutionary algorithms and quantitative structure-activity relationships

In drug design and development one of the prime views is that the biological activity of a given compound is determined by its physico-chemical characteristics. Already in the 19th century it was postulated by Crum Brown and Fraser (cited in Parascondola, 1980) that "there can be no reasonable doubt that a relation exists between the physiological action of a substance and its chemical composition and constitution". In more recent days Hansch and coworkers (Parascondola, 1980) were the first to suggest that such a relationship can also be expressed in quantitative terms, as in the following equation:

$$\text{Biological activity}=a_0+a_1.\text{descriptor}_1+a_2.\text{descriptor}_2+\ldots.+a_n.\text{descriptor}_n$$

This is called a quantitative structure-activity relationship, or QSAR. In the above formula the biological activity is a numerical value such as the logarithm of the concentration at which a compound exhibits half of its maximal biological activity. The descriptors are numerical values of the properties of either the entire molecule (like the molecular weight) or of a specific part of the molecule. In the latter case, the equation needs to be derived from a set of highly similar compounds.

The major use of a QSAR formula is the prediction of the biological activity of a compound that has not yet been tested or has even not been synthesized yet. This can be done with models consisting of descriptors that can be calculated theoretically. In essence, the structure of the molecule, which is a graph, is converted into a vector of numbers, which can hopefully be related to the biological activity by a (simple) function. In theory QSAR can thus greatly increase the speed and reduce the cost of drug design by eliminating the synthesis and testing of compounds with low activity.

However, the major problem regarding QSAR is that scientists can now choose among many hundreds of descriptors, such as experiment-based descriptors, graph-theoretical descriptors, quantum mechanical descriptors and others. Additionally, researchers are more and more realizing that QSAR does not have to be a weighted sum of simple descriptors. Cross-products and polynomials (Lučić *et al.*, 2003), splines (Rogers and Hopfinger, 1994) and even more exotic functions can be used to forge new descriptors out of the old ones, enlarging the set of available descriptors even more. Since a specific biological activity is commonly only measured in dozens of compounds, the hundreds to thousands of descriptors available will lead to overfitting if fitting procedures are used without proper caution. Since there are no 'golden rules' to govern the choice, selection of the 'right' descriptors is probably the most problematic step in the whole process.

Currently, matrix techniques such as principal component analysis (PCA) (Hemmateenejad *et al.*, 2003) and partial least squares (PLS) (Geladi and Kowalski, 1986) are applied to reduce the number of descriptors used. However, the resulting convoluted descriptors are often difficult to interpret, and the design of more active compounds is cumbersome for a medicinal chemist if the QSAR formula cannot be easily understood.

The more straightforward descriptors can lead to a model that is more easily interpreted, and are therefore still used by many researchers. The traditional way to choose the best descriptors for the model from the wide variety available is called forward stepping. This is a local search process, in which first one-descriptor models are built, of which the best is chosen. Subsequently, one by one those descriptors are added that improve the quality of the model most. Since it is possible however that there are descriptors that are separately not very informative but extremely valuable when combined, global optimization techniques are increasingly being used, among which evolutionary algorithms.

A typical example of an evolutionary algorithm to select the descriptors in QSAR analysis is the work of Kimura *et al.* (1998). In their research, CoMFA (comparative molecular field analysis, a technique that compares molecules by looking at their shapes and the electrostatic fields created by the charges of the atoms) was applied to a set of 20 polychlorinated benzofurans. The molecules were superposed and a grid of 17x15x5 was laid over the superposition. For each molecule the strength of the electric and steric field at each grid point was calculated and used as a descriptor. This resulted in 2550 descriptors, which, using conventional CoMFA with partial least squares, gave a r^2 value of 0.96 and a leave-one-out cross-validated q^2 value of 0.89.

Subsequently, the authors applied their genetic algorithm, GARGS (GA-based region selection). First a coarse grid was defined (8x6x5) that divided the molecules into larger regions. The chromosomes were bit-strings, each bit encoding the inclusion/exclusion of a specific region in the model; a population of 240-bit individuals was created. The fitness of each chromosome was calculated by the q^2 value and the best 90% of the population was selected as basis for the next generation. Uniform crossover on 10 pairs of chromosomes and bit-flip mutation on all chromosomes was applied. Elitism conserved the chromosomes which had the highest q^2 values among the chromosomes which had as many or fewer parameters (Pareto optimality, see section 2).

The genetic algorithm resulted in a model with only 8 regions (43 parameters) which by partial least squares analysis gave a model with r^2=0.97 and q^2=0.95. Thus descriptor selection not only reduced overfitting but also slightly improved the fit of the training set, possibly by removing clutter which prevented the partial least squares analysis from finding the optimum. External validation on a prediction set showed indeed improvement over conventional CoMFA, the root mean square error decreasing from 2.63 to 0.99. GARGS was later used by the same authors in a 3D-QSAR study of acetylcholinesterase inhibitors (Hasegawa et al., 1999).

An addition to conventional parameter selection was presented by Cho and Hermsmeyer (2002). Their algorithm GAS (genetic algorithm guided selection) could be used for two purposes. Next to the binary vector indicating use/non-use of descriptors, each individual also contained a vector of numbers which divided the compounds into several classes. The size of each chromosome was thus equal to the sum of the number of descriptors and the number of compounds. However, only one part of the chromosome was optimized per run, so compound classification was separate from descriptor selection. The fitness decreased with increasing size of the errors in the prediction and increasing number of variables, to prevent overfitting. Roulette wheel selection was used to select the parents for one-point crossover or mutation. In the case of crossover, the offspring replaced the worst parent if it was better.

The data set of Selwood and coworkers (1990) was used as a test for descriptor selection. The set consists of 31 compounds with their biological activity against disease-causing nematodes, measured *in vitro*. GAS selected the same descriptors in its best models as other researchers including Selwood *et al.* (1990) and Rogers and Hopfinger (1994). Subset selection was tested on the XLOGP data set, which contained 1831 compounds. Here each molecule of the test set was assigned to the set which contained the molecule most similar to it, where similarity was measured as the Euclidian distance between the descriptor vectors of different molecules. Subset selection apparently worked, increasing the r^2 for the test set from 0.80 to 0.84. Remarkable is that in the XLOGP experiments the r^2 of the training set was systematically lower than that of the external validation set (such as 0.76 vs. 0.80). Perhaps this has something to do with the relatively small size or higher homogeneity of the external validation set relative to the training set (19 drugs vs 1831 more general organic compounds).

In conclusion, the authors demonstrated that their genetic algorithm did work for both variable and subset selection, though subset selection may be less applicable for the smaller data sets that characterize QSAR.

Descriptors often do not correlate linearly to the biological activity. Therefore, Rogers and Hopfinger (1994) developed an evolutionary algorithm called GFA (genetic function approximation). Its main feature is that it creates individuals that are lists of descriptors on which diverse functions are applied, like splines, squaring, or squared splines. As an example the descriptor HOMO was used to design the novel descriptor $<\text{-9.545-HOMO}>^2$, which was combined linearly with the other (derived) descriptors. A typical individual thus may look like $\{C_4,<2.301\text{-}U_t>,(U_t\text{-}2.966)^2,<\text{-9.631-HOMO}>^2\}$. One-point crossover is applied, mutations either add a descriptor or change the number in the spline function. If a duplicate of the new model does not already exist in the population, it replaces the worst individual. The run is completed if the score of the models stops improving.

The Selwood data set was mined with only basic descriptors (no splines or polynomials). GFA indeed found a better descriptor combination than Selwood had found (r^2=0.72 vs 0.55). In an acetylcholinesterase inhibitor data set of 17 compounds and 3 descriptors, linear as well as spline, quadratic and spline quadratic terms were used. The best resulting models had r^2-values of 0.85. The population of GFA provides the user with multiple models, which are often very similar in quality although they contain different descriptors. This allows users to choose the model they intuitively regard as the best. This is a very interesting point in QSAR analysis, yet choosing the 'right model' is even more poorly defined than choosing 'the best descriptors.' The GFA method has been implemented in commercially available software, such as Cerius^2, and has led to a number of publications by users of that software. Shi and coworkers (1998) selected 112 ellipticine analogues from the compound database maintained by the National Cancer Institute (NCI). They were able to derive meaningful QSAR models with the GFA method after the users had subdivided the ellipticine data set manually into structurally homogeneous classes. GFA using splines yielded cross-validated r^2 values that were consistently about 0.3 units higher than those derived by stepwise linear regression.

Lučić *et al.* (2003) used GFA and other approaches for descriptor selection on 4 different data sets and were somewhat less impressed by the method. This may have to do with the fact that they did not allow GFA to use splines, and that they did not use stepwise selection but another genetic algorithm to select the descriptors for the multiple linear regression, to which they compared GFA.

Of course, a QSAR relationship does not have to be the weighed average of a number of descriptors. Linear models are commonly preferred due to their simplicity and smaller risk of overfitting. However, many investigators are tempted to experiment with different types of relationships. After all, many processes in nature are inherently nonlinear. Yasri and Hartsough (2001) elaborated on the combination of a genetic algorithm and a neural network, which also allows non-linear relationships to be found. The authors used a conventional descriptor-selecting genetic algorithm (single point crossover, bit flips, offspring replaced the parents if it was better) to select 6 descriptors out of the 404 available for a data set of 54 benzodiazepine derivatives. They found that the q^2 was enhanced by the GA/NN combination with respect to multiple linear regression with stepwise descriptor selection (0.90 vs 0.80). It is not clear in this case whether the improved q^2 is due to the incorporation of non-linearity by the neural network or due to the superior descriptor selection by the genetic algorithm.

Neural networks were also used by So and Karplus (1996) who found that the evolutionary programming employed gave a more robust optimization of the descriptor set than the GFA-based genetic algorithm. The Selwood data set was analyzed and an r^2-value of 0.76 was found. Additionally, the authors performed exhaustive enumeration over all three-parameter sets and found that the EP-based algorithm found all of the best 100 solutions with the exception of the 95th, the GFA-based one found only a few. The most likely reason for this is that the EP only replaced parents if the children were better, the GFA replaced all parents regardless of the quality of the children.

Finally, evolutionary algorithms have also inspired researchers to seek beyond the standard descriptor used/not used bit strings. One of these methods is FRED (fast random elimination of descriptors) by Waller and Bradley (1999). Data sets were preprocessed by eliminating zero variance descriptors and descriptors that were collinear to other descriptors. Subsequently FRED started with a population of models composed of either a fixed or variable number of randomly selected descriptors. To prevent overfitting, the rule of thumb was used that there should be at least five compounds per descriptor. The maximum chromosome length was thus set to 6 for the Selwood data set. A progeny factor was used to ensure that the population did always contain enough individuals to include each descriptor on average “progeny factor” times. The user specifies a “kill factor”, which divides the population in a part of higher and lower fitness. Those descriptors occurring only in the low-fitness part are considered deleterious and are eliminated from the descriptor pool using a tabu-like process. After every generation, a new population is generated from the remaining descriptors.

As mentioned, FRED was tested on the Selwood data set. The original 53 descriptors were reduced to 23 in the preprocessing step, and FRED was applied with a kill-factor of 5% and progeny factor of 30. The authors concluded that their algorithm performed efficiently and quite similar to alternative algorithms, yielding the same ‘optimal’ solutions (r^2 of 0.83, q^2 of 0.69).

A good in-depth review of the somewhat older literature on evolutionary algorithms in QSAR is given by So (2000).

Conclusion

Quantitative structure activity relationships form a terrain in which evolutionary algorithms have been applied many times. The most likely reasons for this are that the presence and/or absence of descriptors is readily encoded using a standard genetic algorithm, and that the fitness of individuals can easily be calculated using available statistical techniques.

Evolutionary algorithms indeed seem to be valuable to the QSAR process since they are able to find better combinations of descriptors than the traditional local search processes, as stepwise addition or elimination, can.

Nevertheless, there remain some caveats when applying evolutionary algorithms to QSAR.

The first of these is that the data sets should be picked carefully. It is encouraging that a standard data set seems to have been chosen to enable comparison between the different methods, yet this Selwood data set is a somewhat unfortunate choice from a biological point of view. It is relatively small, only 31 compounds, yet the measured biological activity, the killing of the nematodes, is a complex function of the membrane penetration of the compound, its cellular metabolism and its interaction with the target receptor. This multitude of biological processes makes it unlikely that the activity of the Selwood set can be truly explained by using only six descriptors. Direct measurements of receptor or enzyme affinities would be more valuable since these would include fewer intervening factors.

A second point is that biological measurements tend to have quite large margins of error (about 0.5 log units). It therefore remains to be seen how much a slightly improved r^2 value really means since the inaccuracy of biological data does not allow us to choose between models which differ only slightly in performance.

From a point of view of a medicinal chemist, it seems that researchers in the QSAR-field have been introducing more descriptors, and more complicated, nonlinear, methods over the last few years. This development may have been prompted by the need to improve the predictiveness of the models. Though these developments offer opportunities for improved modeling –and even more opportunities for overfitting-there are some practical problems in interpreting and using the results.

Neural networks in particular are difficult to interpret and do not readily suggest to the chemist how a structure can be improved. Another computational method, like a evolutionary algorithm, may be necessary to perform “inverse QSAR” to find better structures in such a case. The structures can then be optimized using the predicted biological activity as a fitness function. However, the problem remains that results researchers do not understand are often used reluctantly, if at all.

A striking observation is that most QSAR techniques find a wide range of models that differ only minimally in their fitness, yet contain entirely different descriptors. This makes one wonder about how the “quality” space looks, and whether the descriptor sets do not rather describe similarity between compounds of similar activity instead of producing formulas that can be truly extrapolated beyond the measured activity range of the tested compounds. For instance, logP, a measure for lipophilicity and therefore membrane penetration, is almost always significant in a “Selwood”-

QSAR. This implies that a factor that is really important in such a system does surface consistently; the other descriptors however may be more “classifying”, rather than “causing” the activity.

Such classification would however not help much in achieving the real purpose of QSAR, which is to find a formula which predicts biological activity with such accuracy that one can use it to design new compounds with higher biological activity than the compounds of the training set. Unfortunately, none of the articles reviewed here contains this extrapolation step that would be crucial for validating the usefulness of the models.

In conclusion, evolutionary algorithms seem to improve the parameter selection of quantitative structure-activity relationships, especially since they can be applied to other than linear models. The main problem for further application of evolutionary algorithms is not so much in improving the quality of the models, but in testing whether the models can extrapolate reliably. Leaving the most active compounds out of the training set and using them as validation set might provide such a check. This has not been done in the articles discussed and is generally neglected in other QSAR publications. The reason for this may be that QSAR is known not to be very well suited for extrapolation; results such as q^2 values are likely to be much worse if the omitted data points have to be extrapolated. The failure of QSAR in these cases is rather a weakness of the current implementations of the QSAR paradigm than of the evolutionary algorithms used to optimize the parameter choice. A second opportunity for application of evolutionary algorithms would be to increase the availability of models. Next to the traditional linear models and neural networks, genetic programming might be applicable to find novel ways to combine existing descriptors. Also there is still an avenue less explored by evolutionary algorithms, i.e. to use the QSAR models for reverse engineering of compounds. Synthesizing and testing these compounds will truly test the validity of the QSAR methods employed and the value of evolutionary algorithms therein.

6. Evolutionary Algorithms in Ligand Docking

Ligand docking, generally simply called “docking” in the medicinal chemistry community, places a small molecule, called the ligand, into a protein in the same way that nature does. Docking could be compared to a 3D jigsaw puzzle, in which the pieces can be turned in more than four ways and can also change shape.

Docking is a very important tool in medicinal chemistry. If one can reliably predict how a molecule will bind to a protein, visual inspection of the fit may give information to the drug designer at which positions the molecule fits well, at which positions there is a worse fit, etc. Based on this information a molecule can be designed that binds more strongly. Also, if the original molecule would not be suitable as a drug due to its toxicity or other undesirable properties, one can dock other molecules and select those that seem to bind well for further testing; this can be much cheaper and faster than measuring the binding strengths experimentally.

In the ideal case, a docking program would give the medicinal chemist a list of alternative docking options of the ligand into the receptor, and assign to each docking an energy value indicating the binding strength. If the docking procedure is really perfect, there will be a sizeable energy difference between the best and second-best conformations, which indicates a large chance that the best docking is also the true docking.

However, so far no program has reached this ideal. There are two main reasons for this:

1) The energy function is often not very accurate. This means that there may be docking options of the molecule that are in reality higher in energy than the real docking, but are indicated by the energy function as lower in energy (the lower the energy is, the better the docking). Some interactions between the molecule and the receptor, such as hydrophobic interactions and entropic effects, are notoriously difficult to model.
2) The search space is often very large. The ligand has three translational and three rotational degrees of freedom, as well as one degree of freedom for each rotatable bond. Additionally, the hydrogen atoms in the receptor, which are usually not visible on the X-ray crystallographic structure, also must be in the right orientation for good binding between protein and the ligand. All these degrees of freedom result in a search space of about 10^{20} to 10^{30} possible docking options, which cannot be searched fully.

Although docking is certainly not easy, much effort has been and still is being spent improving existing methods and developing new ones. After all, the structure of the protein target itself will give much more information for drug design than any QSAR model based on just the ligands can. A perfect docking program would be incredibly valuable, one of the holy grails of drug design.

Many docking techniques have been developed so far. The following part of the review will discuss the role of evolutionary algorithms in the more recent applications.

A well known and often used example of evolutionary algorithms in docking is GOLD (Genetic Optimisation for Ligand Docking) developed by Jones *et al.* (1997). This is a genetic algorithm that uses chromosomes encoding the internal torsion angles of the ligand, as well as two integer strings representing hydrogen bonds between the ligand and the receptor. The latter replace the more conventional "location and orientation" parameters, since the location and orientation of the ligand is determined by least-squares fitting of the ligand's hydrogen donors and acceptors onto those of the protein.

Each individual is evaluated by first performing the least-squares fit of the hydrogen acceptors and donors of the ligand onto the hydrogen donors and acceptors of the protein. Subsequently the internal energy of the ligand and the ligand-protein interaction energy are calculated, the sum of which determines the fitness of the docking. The population is divided into 5 subpopulations with 100 individuals each. The operators are crossover, mutation and migration, which are applied as alternatives rather than sequentially. Mutation and crossover differ for the binary (torsional) and the integer (matching) part: bit flip mutation and one-point crossover are used for the binary string, mutation to a random valid value and two-point crossover is applied on the integer string. Additionally, a niching technique is used, which makes new individuals replace the worst individual of a similar subgroup instead of the worst individual of the entire subpopulation.

The genetic algorithm was tested on 100 protein-ligand complexes, and run 20 times on each. The resulting conformation of lowest energy was compared with the corresponding crystal structure. In 71 out of 100 cases GOLD found acceptable solutions, in which all or most parts of the ligand bound to the right place in the receptor. Also the genetic algorithm generally did not need 20 runs, in 49 out of 100 cases 2 runs were enough. However, errors in the scoring function found a false minimum in at least seven of the 100 cases and regarded it as superior to the crystal structure, which is biologically spoken not probable. Other problems encountered involved ligands that had too few hydrogen donors and acceptors, which made the least squares fitting work poorly, and inaccurate protein structures. If the protein structure had a resolution more accurate than 2.5Å, GOLD succeeded in 77% of test cases, else it succeeded in only 52%. Finally, in some cases the structure of the ligand was distorted by the protein, therefore docking using the normal ligand failed.

All in all, GOLD seems an interesting computer program that can dock ligands over a wide range of test systems. The authors indicate that incorporating protein flexibility in the algorithm would be a useful addition, though probably not necessary in all systems.

Morris *et al.* (1998) also used a genetic algorithm, but added Lamarckian evolution (Autodock). In every generation 6% of the population was optimized using local search, and the improved parameters were written back to the genes. The chromosome here is a string of real-valued genes. The first three values are the Cartesian coordinates of the ligand, the four following values define the orientation. Usually three are sufficient for orientation, but then the so-called "gimbal lock" problem may occur, in which an unfortunate rotation can make two rotational axes of the object point into the same direction. The last values represent the internal torsion angles. Crossover is two-point and always takes place between genes. After crossover the mutation is performed, in which the values are mutated using a Cauchy distribution. A population of 50 individuals was used, and a maximum of 27000 generations or 1.5×10^6 energy evaluations.

Seven protein-ligand complexes were docked with the Lamarckian genetic algorithm, and for comparison purposes also with simulated annealing and a normal (non-Lamarckian) genetic algorithm. Ten runs were performed per method per complex. It turned out that the Lamarckian genetic algorithm clearly outperformed simulated annealing, which had a large root mean square distance of the fitted relative to the crystal docking (>3Å) in 2 out of 7 cases. The root mean square distance between fitted and real ligand was quite small in both the Lamarckian genetic algorithm and the genetic algorithm (under 1.5Å). The energies in the Lamarckian algorithm were slightly lower, though at the cost of more energy evaluations. Additionally, the Lamarckian genetic algorithm found the minimum conformation in 78% of the runs, the genetic algorithm and simulated annealing reaching 40% and 24%, respectively. As a validation, the binding energies returned by the fitness functions were compared to the experimental binding energies. The prediction error ranged from –3.89 to +9.93 kcal/mol, which means that predicted binding affinities vary by a factor 1000, which is the difference between very good and quite bad ligands. The largest deviation, 9.93 kcal/mol for the streptavidin/biotin complex suggests that the protein flexibility might be too important to neglect in this case.

Several attempts have been made to improve upon this Lamarckian algorithm. Hart *et al.* (2000) performed experiments with different settings of the local search and found that improvement was possible by taking another local search algorithm, a pattern search method, instead of the previously used Solis-Wets algorithm, and increase the number of steps in the local search procedure. Other experiments of the same author used self-adaptive evolutionary programs and evolutionary pattern search algorithms (Hart *et al.*, 1999). The evolutionary programs could adapt the step size of all search parameters, while the evolutionary pattern search algorithms used only one step size which was slowly decreased over the course of the evolution. The evolutionary pattern search algorithm was configured in such a way that theory guaranteed that it would converge to a stationary point. While both methods performed decently, they were still outperformed by the optimized local search method. According to the authors, this indicates that the local search had a more extensive effect on the evolution than just performing localized step length adaptation.

Thormann and Pons (2001) parallellized the Autodock algorithm for use on multi-processor machines, and called the result EGA/LS (Enhanced Genetic Algorithm with Local Search). Dividing the population between the processors resulted in a natural island model, which proved to be superior to a single-population model. For the more difficult test cases the island model was more effective than a pooled population (the minimum found in 76% vs 66% of docking options). Migration between the populations was taken care of by one individual called the “king”. The king could be overwritten by the fittest individual with a certain chance, and the king itself occasionally overwrote some individuals in the subpopulations. In each run, the subpopulations were randomly initialized three times, but after the first and second round the king was kept, seeding the populations slightly so that convergence could be reached faster.

Three test cases were taken, which took on average about 9 seconds to dock. Unfortunately, no root mean square distance data was given by the authors. Hence one cannot know whether the crystal structures were reproduced. One hundred runs were made with eight subpopulations of size 25. In general, since docking within one run is not assured, the authors advise to use at least three test runs for each complex. The main problem encountered was that when many degrees of freedom were taken into account (all torsion angles of the ligand and some of the protein as well), the optimization got stuck in local minima. However, the local minima that EGA/LS found were lower in energy than those discovered by GA/LS, indicating that splitting the docking populations at least offers enhanced possibilities to escape from local minima.

More recent work based on AutoDock has been described by Thomsen (2003). The author did experiments to optimize the evolution parameters of the evolutionary algorithm used by AutoDock. The optimal settings were found to be a population size of 100, and mutation which was a slowly annealed Gaussian. Arithmetic crossover, which creates offspring out of a weighted combination of the genes of the parents with in this case a random weight for each gene, was found to be superior to the traditional single point, two-point and uniform crossover. Strikingly, the new evolutionary algorithm did not profit from adding the Lamarckian local optimization. While the improved evolutionary algorithm showed no significant improvement over the Lamarckian genetic algorithm in the three simplest test cases (7-11 dimensional search space), it improved upon its predecessor in two of the three more complex test cases (12-18 dimensional search space). Ironically, though the docking energies found were generally lower, the root mean square distances from the crystal structures were slightly increased from those found by the Lamarckian genetic algorithm. The efficiency was increased however, the "DockEA" needed only 50,000-150,000 fitness evaluations, while the Lamarckian genetic algorithm needed over 250,000 evaluations to obtain accurate and reliable results.

From these experiments the author concluded that the energy function needs some improvement to make the lower docking energies also correspond most closely to the crystal structure, but the deterioration of docking quality in one of the more complicated test cases relative to the Lamarckian algorithm indicated that the balance between exploration and exploitation is sensitive to the protein structure, and testing on more complexes would be required to refine the docking algorithm.

Among the evolutionary algorithms, genetic algorithms have been most prominent in docking. Yang and Kao (2000) however created a docking method called FCEA (family competition evolutionary algorithm), which is more similar to evolution strategies. Next to the vector of real numbers encoding the location and orientation (6 numbers) and the torsional angles of the ligand, each individual contains three additional vectors. They are of the same size as the data vector, encoding the parameters for a decreasing-based Gaussian mutation, self-adapting Gaussian mutation and self-adaptive Cauchy mutation, respectively. Thus, each gene in each individual has three self-adaptive mutation parameters. The mutation step consists of subsequent application of the three mutations (decreasing, Gaussian, Cauchy). In each of these submutation steps each member of the population generates *l* children by mutation or recombination with another member of the parent population. The fittest of the children survives. In most cases, from each pair of father-child the best survives into the next generation, sometimes however from the entire population of *n* parents and *n* children the *n* best solutions are selected. For further details on the rather intricate procedures and many parameters used in this algorithm the reader is referred to the publication itself.

The resulting program was subsequently tested on one protein, the enzyme dihydrofolate reductase, with three different ligands. Population size was 50, the maximal number of generations 250. The results were compared to those of DOCK and other docking programs. While DOCK found the best fit to the crystal structure (RMSD 0.6Å vs 0.67Å), the average fit (over 20 runs) by FCEA was better (1.37Å vs 2.4Å). However, using only one protein structure for comparison seems a bit meager for a conclusion on the general competitiveness of this method.

Since evolutionary algorithms are by far not the only computational methods used for docking, Vieth *et al.* (1998) made a comparison between three common methods: molecular dynamics, Monte Carlo and a genetic algorithm. The genetic algorithm was kept relatively simple. The population of 90 individuals was split over five subpopulations with an elitism of 2 per subpopulation. In each generation, the individuals were modified by using single-point crossover, mutation or migration. In migration, two individuals were exchanged between subpopulations. The search was performed in two stages for each algorithm, in which the parameters in the second stage were adapted to fine-tune the solutions found in the first phase.

Five ligand-receptor complexes were used as the test set. It turned out that the genetic algorithm worked best for small search spaces in which the ligand was located within 3Å of its actual binding site and the molecular dynamics performed best for larger search spaces, within 11Å of the binding site. While the genetic algorithm gave the highest fraction of runs that found a good docking, the molecular dynamics algorithm returned the conformations with the lowest energy and closest fit.

Combination of the different computational techniques is also possible. An example of this is the Mining Minima optimizer (David *et al.*, 2001), which uses a combination of the so-called global underestimator method, genetic algorithms and the poling method of Smellie *et al.* (1995). However, it is possible to regard it as an elitist genetic algorithm with some twists. First a large population of individuals is created within a certain search region. The individual with the lowest energy is used as the center around which the next generation of docking options is created. After each generation the width of the search region is narrowed down. To prevent the rediscovery of energy minima, exclusion zones are placed around previously found minima. There is a crossover-like operator, which combines a newly designed individual by partially copying information of a previously found minimum into it. The modified new individual is then placed in the next generation.

The authors tested their method on 27 complexes and compared their method with the genetic algorithm, simulated annealing and tabu search of PRO_LEADS, as well with AutoDock, FlexX and MCDOCK. Also nine of the “difficult cases” of GOLD were tackled with the Mining Minima optimizer.

The median docking time of the Mining Minima optimizer turned out to be about 1.2 minutes. The results of the comparison with the other programs indicate that the Mining Minima method is comparable to PRO_LEADS and the other programs; occasionally it scores higher, sometimes lower. The authors point out that in some cases the fit was good, yet the "objective" root mean square distance criterion indicated low quality. In some of these cases the solvent accessible parts of the ligand, which are relatively free to move, cause the main part of the error. This contribution is however not very relevant since these parts do not influence the quality of the docking, which is defined by ligand-protein interactions. Six out of nine of the docking options that were difficult for GOLD were solved. Three remained problematical: in one case the crystal structure itself was suspect, in the other two cases the global minimum was found at another place than the binding site.

If the elaborate comparisons in this article make one thing clear, it is that the different algorithms are more or less suitable for different complexes, since no method is superior over all other methods in all investigated complexes. Moreover, some complexes seem much more difficult to solve than other complexes, whichever method is used.

For a broad overview of the many different methods (genetic, simulated annealing, fragment-based methods, etc.) of docking and the many programs using these, the review of Taylor *et al.* (2002) is recommended.

Conclusions

Several evolutionary algorithms have been developed for docking ligands into the active site of proteins, and all obtain reasonable to good results, quite like the other heuristic algorithms. It is so far doubtful whether evolutionary algorithms have inherent advantages that make them more efficient for docking than for example simulated annealing. Since the coordinates of all atoms depend on the location and orientation and many of the torsion angles, this high coupling would make it very unlikely that there are small simple building blocks that can be recombined with each other into larger, high-quality building blocks. This view seems to be supported by findings such as that of Thomsen (2003) that arithmetic crossover outperforms the more conventional uniform and one- or two-point crossovers. Several promising techniques have been found, such as introduction of subpopulations, employing local optimization next to the normal genetic algorithm, and using different crossover and mutation methods. Further investigation is necessary, however, to conclude whether combining these will further improve docking efficiency and efficacy.

In any case, several authors have displayed great ingenuity in introducing novel and complex operators, most notably Yang and Kao (2000). Yet the arguments for this complexity are lacking, with the possible exception that the other methods do not work perfectly. If the complexity of the evolutionary algorithms is enhanced, it should be done either carefully and on the basis of solid experimentation, that is, study of many complexes, or it should be based on knowledge of the chemical and biological reality. Otherwise such methods might “overfit” their docking options due to an overdose of adjustable parameters and a paucity of test cases.

However, comparison of the diverse methods employed is extremely difficult since the three separate components of the docking procedure (fitness function, search method and test data) are generally different per article. A desirable development for this field would be the introduction of a library of standard search algorithms, fitness evaluators and test data sets. Only then a new algorithm can be truly compared to existing ones.

Another development would be the incorporation of protein flexibility. Proteins can dock different small molecules in their active site. Most method developers, understandably, have docked the ligands of known complexes and compared those to the crystal structures. But is a crystal structure of the protein in which ligand A is docked also suitable for docking ligand B? Or would subtle differences in the protein conformations prevent finding the real docking? Such extrapolation is of vital importance for predicting the binding of series of molecules, and may necessitate extending the algorithms with some protein flexibility.

Further advances are also needed in the area of fitness functions. Since some discovered docking modes have lower energy (according to the computer) than those of the crystal structures, the energy evaluation procedures should be improved. Training the force fields that evaluate the fits by finding the right parameters and formulas might by itself also be an interesting field for applying evolutionary algorithms (an application of an evolutionary algorithm in descriptor selection for such a model is the work of Deng *et al.* (2004), see section 8).

The ultimate goal of docking algorithms, taking a protein structure and a ligand structure and calculating both the position in which the ligand will be bound and the affinity of the ligand for the receptor, will probably need the following extensions of existing docking algorithms:

1) Addition of protein flexibility to accommodate the binding of different ligands and correct for errors in the crystal structure.

2) Addition of water molecules to the active site; these can influence binding as well.
3) Calculation of the changes of entropy on the protein, the ligand and the water molecules during the binding process.

In conclusion, much remains to be done in the field of ligand docking. It is not certain yet whether the docking algorithm of the future is a pure evolutionary algorithm, basic simulated annealing, or one of the other methods currently applied. Most likely the ultimate docking method will incorporate the most suitable properties of existing search methods combined with chemical and biological heuristics. There is still a long way to go before ligands can be docked automatically, accurately and with good binding energy estimations into their receptors, but the end result will undoubtedly be extremely worthwhile.

7. Evolutionary algorithms in *de novo* design

To find molecules with a specific biological activity, compound libraries are commonly screened. However, “only” about 10^{10} structures have been synthesized by chemists thus far, while the number of all possible drug-like molecules is estimated to be at least 10^{60} (Gillet, 2000). Clearly, designing new molecules may be required to cover more of this “chemical space” and to find a molecule that would be a more suitable drug against a certain disease than any currently known molecule. This process of designing new molecules is called *de novo* design.

Applying computer programs to design molecules seems an obvious choice, especially since computers can create virtual molecules much faster than humans can. However, the set of all possible molecules is difficult to search systematically. One of the reasons for this is that the number of possible mutations rises with the size of the molecule. If one defines a mutation as a single step in chemical space (changing/ adding/removing an atom or bond), the number of orthogonal steps/dimensions increases with the number of atoms in the molecule. So a “normal” drug molecule, which may contain e.g. 20 non-hydrogen atoms, can have over one hundred possible one-atom mutations. This results in a very high-dimensional search space, and the dimensionality will only increase when larger molecules are allowed. This makes a systematic search of all possible molecules to find those with the desired properties quite difficult. Also, a molecule is a graph and therefore is difficult to represent by the traditional vector notation of a genetic algorithm. Additionally, the rules of chemistry limit the number of possible molecules by demanding that e.g. every oxygen atom has two bonds, and every carbon atom four. Therefore, mutation from a carbon atom to an oxygen atom will always involve some additional modification of the molecule, like removing hydrogen atoms. This is sometimes possible, sometimes not, depending on the rest of the molecule. Lastly, developing a proper fitness function is probably the most challenging problem of all. Since experimental fitness evaluation is slow and expensive, the search goes on for computational methods that predict the properties of a molecule reliably.

To find promising molecules in the vast chemical space, several different evolutionary algorithms have been developed and applied to a variety of *de novo* design problems. A good review on some of the older work has been written by Gillet (2000). This review will only briefly discuss the earlier work and mainly cover the work performed in the last few years.

One of the first and best known applications of evolutionary algorithms in *de novo* design is the work of Glen and Payne (1995). Since a molecule is a graph that can contain cycles, a traditional linear chromosome with bit-flip mutations could not be used. Therefore a graph representation of the molecule itself was used as genotype, in conjunction with linear chromosomes which indicate the position and orientation of the molecule and the torsion angles, similar to ligand docking.

Mutation of the orientation, position and torsion angles was performed using an approximated Gaussian function. The structure of the molecule could be altered by a set of eight mutations, which included adding and deleting atoms or groups of atoms, forming and breaking rings, and changing atom types. Crossover could be 1-point or 2-point between single, non-ring bonds that occupied approximately the same 3D-coordinates. The fitness function consisted of a weighed combination of scalar properties of the molecule such as molecular weight, surface properties and the fit of the molecule on a predefined grid. Selection was done by the roulette wheel method.

The authors performed two experiments. One experiment was aimed to design molecules that resemble ribose, the other to design molecules that fit the active site of the bacterial enzyme dihydrofolate reductase (DHFR). Population sizes of 50-100 were used, since lower sizes such as 10 were found to be too erratic due to premature convergence. Evolution indeed improved the fitness scores from 100 to –30 for the ribose analogs. Also, the average score of the best four molecules in the initial population of the DHFR experiment was 26.3, but converged after 32 generations to –32.1.

The authors envisioned two extensions to their program. The addition of metal atoms and transition states might be useful to mimic enzymes better. Another important improvement would be a fitness function that gives a more biologically relevant value, such as binding strength. This would also eliminate the need to set the relative weights of the many fitness criteria manually, which is far from objective. Nevertheless, the diverse mutations and the 3D-representation of the molecule were designed very well, and as of yet few *de novo* design programs have improved on Glen and Payne's work in these respects.

Worth mentioning as another pioneering study of evolutionary algorithms in *de novo* design is the work of Westhead *et al.* (1995). The authors first generated and superposed an initial population of molecules, which formed the input for the evolutionary algorithm. Similar to Glen's work, molecules can be crossed only if they have single bonds that lie near each other in the superposition. However, mutation is limited to rotation of torsion angles, and the fitness function is less sophisticated than Glen's, being the number of functional groups that overlap the functional groups of a known molecule in a superposition. Analogs of the molecules distamycin and methothrexate were nevertheless found and scored higher than the initial population of molecules.

However, though the molecule itself is a graph, the genotype of the molecule does not have to be a graph. Other representations might have advantages for an evolutionary algorithm.

Nachbar (1998) developed a evolutionary algorithm that converted the molecule graph into a tree, in which cycles are represented by special ring nodes. Mutation involves changing the atom types or bond orders, but crossover is responsible for the major part of structural change. The crossovers are very much like those of genetic programming, though subtrees containing an open ring bond are not exchangeable. The fitness function was a graph descriptor-based QSAR which predicted toxicity of the compound in tadpoles. After the population of 50 individuals had been evolved for 50 generations, 30% of the molecules were in the desired activity range. In this case it is somewhat difficult to establish the efficiency of evolution since no data were collected on the evolution of the population's fitness during a run (R.B. Nachbar, personal communication).

The algorithm did have some small problems, such as that many molecules in the final generation were identical, which is not very useful for a chemist who wants as many alternative solutions as possible. Checking for duplicates will probably be important in any *de novo* design method. A problem caused by the tree-like representation was that ring manipulation was difficult. The author would have liked to be able to expand, contract and break rings at other positions than the ring closure bond, but this was not easy to implement.

The ring opening problem was solved in subsequent work of the same author (Nachbar, 2000) by inverting/re-rooting subtrees. The fitness function changed to molecular similarity, and several test molecules were recreated by the evolutionary algorithm, with the exception of a large polycyclic molecule, which turned out to be difficult to generate due to the surrounding local optima.

Douguet *et al.* (2000) used the chemical SMILES-notation (Weininger, 1988) to represent the molecules. SMILES is also tree-like, yet contains fewer brackets since hydrogen atoms are not explicitly stated and the superfluous brackets in the linear parts of the molecule are omitted. Two crossover operators, one-point and two-point, were implemented, as were thirteen mutation operators (though the article, oddly, describes only eight). These were quite similar to Glen's, though ring breaking was absent. This may be due to similar problems as Nachbar encountered with tree representations. Fitness was calculated as a weighed sum of a few physicochemical criteria, such as the solvent accessible surface and the dipole moment of the molecule, which had to be in a certain range, and roulette wheel selection was used. As test cases, the target criteria were set to the properties of retinal and salicylic acid. The evolutionary algorithm did indeed find mimics of these molecules. The structures of some of the molecules were adapted by medicinal chemists to make them easier to synthesize. In contrast to the work of Nachbar, the authors considered crossover to be very much like a macromutation due to the tree-like representation, and it was applied much less frequently than mutation.

Globus *et al.* (1999) handled rings more elegantly by using "genetic graphs". These have the advantage that they look very similar to real molecules. A crossover operator was implemented which could easily cross over rings, which had not been done yet by other authors. However, no mutation operator was used. This had the unfortunate result that if a generation happened to contain only rings, no chains could be generated, and vice versa. The fitness function was graph similarity to a specific molecule. Globus demonstrated that his algorithm can indeed recreate complex molecules, even those which have different atom types and a complex structure, like the five rings-containing morphine. The authors acknowledge, however, that rediscovering known molecules is not very useful, and that a fitness function that gives biological activity should be implemented.

Simplifying the representation of the molecules can work, but tends to restrict the possible mutations. This may make an escape from local minima more difficult. Another way to apply the evolutionary algorithms more easily is to adapt the problem domain and only consider subsets of molecules which have a relatively simple structure.

Schneider *et al.* (1998) used experimental data on the biological activity of peptides to train a neural network to predict activity from structure. Subsequently an evolutionary algorithm was applied that chose the best individual from the initial population. Since peptides are linear chains of amino acids, a linear chromosome can be used. Mutation can then be performed by picking a position and substituting the amino acid there by another amino acid. The best peptide filled the next generation together with its mutants, after which the new best peptide was selected. Unfortunately the neural network made quite inaccurate predictions, which was aggravated by the errors in the biological data used to train it. Nevertheless a peptide with comparable activity to the seed peptide but a very different sequence was found.

Related work was performed by Patel *et al.* (1998) who focused on bactericidal peptides. A training set of 29 peptides with measured biological activities was used. Using this set, 29 multi-layer perceptron neural networks were created, each based on 28 peptides. The fitness value was taken to be the average of these 29 models. The genetic algorithm used was somewhat more conventional than that of Schneider *et al.*, having a population of size 100, elitism that conserved the best 25, probability of crossover (two-point) 0.6 and probability of mutation 0.033.

The genetic algorithm was shown to be much more efficient than Monte Carlo or random search in finding peptides with high predicted activity, since only 0.008% of randomly generated peptides were in the desired activity range, 0.5% of those generated by Monte Carlo but 7.2% of those made by the genetic algorithm. Of the more than 400 candidate peptides that were generated by the genetic algorithm, the 5 most diverse were synthesized and were shown to have high-ranking bactericidal activity.

With the traditional fixed-size chromosomes the length of the peptide cannot be modified; this may however be important for optimizing activity. Kamphausen *et al.* (2002) solved this problem by implementing $n \times m$ crossover. This technique selects a group of parent peptides and aligns the sequences. It enables the shorter sequences to align with the longer sequences by filling the empty space at the end of the shorter sequences by repeating the first part of that sequence until the maximum length is reached. The length of the child peptide is then determined by averaging the length of the best parent peptide and the average of the other parent peptides. Subsequently, the child is assembled by taking one value per column in as many aligned columns necessary to reach the target length. The implemented version of this mutation also allows the sequence to "shift", which can lengthen and shorten the sequence at both ends.

The program was used to find a peptide that optimally inhibited the blood clotting protein thrombin. The population contained 123 peptides of lengths 6 to 12, whose fitness was determined experimentally. Four cycles of design and testing were performed. With each generation the average activity increased, and in the fourth generation a very active inhibitor was discovered. It was more potent than known peptide inhibitors of thrombin, and this experiment can thus be considered to be successful.

Peptides, however, are currently only rarely used as drugs since they generally have unfavourable physico-chemical properties. Conventional drugs are much smaller molecules, which can be absorbed more easily by the body. Schneider acknowledged this and also created a evolutionary algorithm for small molecule design, TOPAS (Schneider *et al.*, 2000a,b). This program again uses Schneider's method in which only the fittest individual survives and procreates, but uses molecule fragments instead of amino acids. A subset of about 3 of approximately 25,000 fragments is converted by the algorithm into a real molecule. The fragments also contain data on the connections they can form, which should allow the constructed molecule to be easily synthesized in the laboratory. Mutation is implemented by replacing a fragment by a similar fragment with the same type of attachment point. The fitness function calculates the similarity of the constructed molecule to a known ligand.

In the test case, TOPAS identified a ligand chemically not very similar to the original molecule, but with receptor affinity, be it a 1000-fold less potent. While one could not yet argue that this evolutionary algorithm develops structures that improve the affinity of a known ligand, it can find compounds with a similar kind of activity in a very different class of chemical structures.

The fragment-based approach was also used by Pegg *et al.* (2001). However, in their algorithm runs use far fewer different fragments (in the order of dozens). Acyclic graphs containing maximally 16 fragments are constructed. Crossover is performed by exchanging subtrees between individuals, mutation by changing one of the fragments in an individual or by connecting a fragment to another fragment in the same individual as long as this does not introduce a cycle. The fitness is determined by docking the resulting molecule into the active site of the target protein.

Three test cases were taken: design of cathepsin analogs, inhibitors of dihydrofolate reductase and inhibitors of HIV-1 reverse transcriptase. The results of the evolutionary algorithm were compared to the experimental data available. Two major problems were discovered. First, fitness evaluations took much time: a run of 100 generations of a population of 20 molecules took 5 hours of processor time. Second and worse, not all good inhibitors were judged as good by the fitness function. So while the evolutionary algorithm designed many molecules with higher fitness values than the compounds that turned out best in the experiments, it remains to be seen if those molecules actually bind better. Like Schneider's program, the fitness function is ill-equipped for optimizing the activity, yet the generated libraries do find general trends, i.e. substructures that seem to work. It is likely that the libraries generated by Pegg's program are better than randomly designed libraries in binding to the target site. However, since no experimental validation was performed, definitive conclusions on the effectiveness of this method cannot be drawn.

The SYNOPSIS program by Vinkers and coworkers (2003) can be considered a synthesis of the good points of both Pegg and Schneider with some additional clever ideas. The database constructed by the authors contains about 32000 molecules, which can be transformed and combined using 70 different reactions. A chromosome represents a sequence of molecules and reactions, which is transformed by the program into the actual molecule. Mutations consist of adding reactions or changing reactants. The fitness function is the docking score of the molecule binding to the enzyme HIV-reverse transcriptase.

A good point of this program is that it automatically suggests a synthesis route for the molecules. For 8 out of 28 molecules the synthesis route was followed and succeeded, while for only 3 molecules the suggested route was tried and failed. In the other cases a different method was taken or a compound differing from the original suggestion was made. Therefore, depending on the definition of success, 29% to 64% of syntheses succeeded.

Similar to Pegg, finding good inhibitors proved to be more difficult. The docking function was extremely slow (1 processor-hour per compound) which probably only allowed small populations and a low number of generations, although the article gives no numbers on these. Also the docking function was quite inaccurate. For all suggested ligands a high binding strength was calculated, but a low binding strength was found in experiments. Similar to Schneider's approach, the evolutionary algorithm acts not so much as an optimizer of biological activity, more as an idea generator of molecules that are on average much more active than one would get from a random library screening. In that respect SYNOPSIS is a success.

Conclusion

A wide variety of evolutionary algorithms has been applied to *de novo* design. Their applications and results highlight both their successes and their current shortcomings.

The two main challenges, i.e. representation of the molecular structure and the fitness function, have been addressed by the authors with varying success. The many applications of evolutionary algorithms in "simplified" chemical domains make it clear that representing the molecule remains difficult, and that mutation and crossover are not straightforward to implement. However, the work of Glen and Payne (1995) has clearly shown that evolutionary algorithms can be applied very well by using the molecule as its own representation. Implementing mutations and crossover will remain amenable to tweaking and discussion, but basically this problem has been solved.

There are currently two major elements in automated *de novo* design to focus on. The first is that the molecules suggested are not always easily synthesized. The fragment-based approach by Vinkers *et al.* (2003) to use available molecules and known reactions is promising. However, it also calls attention to the fact that due to problems in reaction prediction only few of the thousands of available chemical reactions can be used by the program. And even those few "robust reactions" fail quite often. Additionally, limiting the reactions and the building blocks will undoubtedly confine the parts of chemical space that can be explored by a 'fragment and reaction'-based algorithm. Also the fragment-reaction like structure of the chromosomes makes mutation awkward: fine-tuning a molecular structure that is almost right is extremely complicated and therefore not very likely to happen. On the other hand, atom-based mutations like those of Glen and Douguet allow more refined exploration of the chemical space and relatively easy fine-tuning of the molecular structures. Yet they have the disadvantage that synthetic feasibility of the resulting molecules is doubtful. Perhaps the ideal algorithm will use a combination of these two approaches.

For the second weak point of current *de novo* drug design algorithms, i.e. the fitness function, good solutions seem even harder to find. Docking, which in principle yields the best affinities for a broad diversity of molecules, is extremely slow and moreover gives results that are too inaccurate for optimization. This suggests, as in section 6, that the most important contributions to this area by evolutionary algorithms would be in deriving proper binding functions from quantum mechanical and experimental data. Additionally, there is the problem that many important proteins are membrane-bound, and that their crystal structures are therefore extremely difficult to determine. This means that docking is currently not applicable to a large portion of interesting drug targets. Experimental fitness determinations are for now the only alternative, yet it may well be that a evolutionary algorithm used interactively by medicinal chemists would need fewer syntheses to achieve optimization than the traditional methods.

In conclusion, while the quality and applicability of the discussed evolutionary algorithms for *de novo* design varies, they do show promise. Even at this moment the applied evolutionary algorithms with their crude fitness functions give inspiration for unconventional analogs of known ligands, which opens up alleys otherwise closed off by patents or unfavourable physiological properties of the original ligands. As fitness functions become faster and more accurate, the future of evolutionary algorithms in *de novo* design looks very bright indeed.

8. Other applications of evolutionary algorithms in drug design

The scope of application of evolutionary algorithms in drug design is wide. Whereas in the previous sections the more prominent uses were discussed, this section will focus on some less mainstream work. The publications discussed here may give an impression of other areas that have been tentatively trodden, a brief glimpse of areas that may become more important in the near future, and inspiration for application of evolutionary algorithms to other problems related to drug design.

If a large database of molecules has to be screened for biological activity, most drug developers would prefer to test only the most promising compounds. If these have the much sought after but ill-defined “drug-likeness” property, they will have a larger chance of being a good drug. While one could argue about the merits of selecting for drug-likeness versus selecting for lead-likeness (see section 2), the search for drug-likeness criteria has inspired some interesting research, amongst others that of Gillet *et al.* (1998). Gillet *et al.* attempted to estimate drug-likeness by taking two databases of molecules, the World Drug Index, which contains about 30,000 drug molecules, and the SPRESI database, which contains 1.7 million molecules, in vast majority nondrugs. Of each molecule in the databases, six simple properties were calculated, such as the number of rotatable bonds and the number of hydrogen bond donors. The value range for each property was divided into 20 bins. Using statistics on a subset of 1000 WDI-molecules and about 17,000 SPRESI-molecules, for each bin the chance was determined that a molecule having its property within the value range of the bin was a drug molecule. The total database was then sorted to see if the drug molecules indeed ranked higher.

It turned out that this method gave some information on drug-likeness. For example, ranking the molecules by only taking into account the number of hydrogen donors resulted in finding 4.6-fold times as many drugs in the top 1000 molecules as would be expected by chance. However, combining descriptors worsened this enhancement, probably because the descriptors were not truly independent.

Subsequently, experiments were performed to see if setting the bin weights by using a genetic algorithm instead of statistics would improve the score. The genetic algorithm used vectors of length 6x20=120 as chromosomes. Mutation changed the value of one bin to a new permitted value, crossover could be one-point, two-point or uniform. Two fitness measures were compared: the number of drug molecules in the top 1000 and the average rank of drug molecules in the drug-likeness list. The average rank resulted in much better scoring over the entire population. The enhancement factor here was 3.0; so to find 50% of all drug molecules only the top 17% of all compounds had to be considered. Subsequently, experiments were performed to distinguish specific classes of drugs from inactive molecules, either by using the generic binning weights or weights specifically optimized by comparing the particular drug class with SPRESI. The discriminative power of the method depended heavily on the therapeutic class. For example, retrieval of anticancer compounds was enhanced 4.9-fold with the generic binning method and 6.8-fold with specific training, while for psychiatric drugs enrichment was only 1.3-fold with the generic method and 2.0 after training. The authors suggest that these differences may be due to the fact that there are relatively few psychotropics and that the class is structurally quite diverse.

The results of this investigation can certainly be considered interesting. Drugs can be somewhat discriminated from non-drugs, even by a simple method such as this one, and the structure of the chromosomes might yield interesting insights on what makes a compound drug-like. However, some problems are not addressed by the authors. The SPRESI-database is not 17, but 54 times as large as the WDI. This means that even with a factor 3 enhancement, only one in twenty of tested molecules in the first third is a drug, which is not a very good score. Additionally, the binning weights are trained on structures that already occur in drugs, so compounds which work via diverse mechanisms like the psychotropics are not readily found. Therefore using this method to prioritize the lead screening for a novel receptor would probably not be very advantageous.

In addition to the question of whether a compound has biological activity, another important question is *which* kind of biological activity it possesses. Xue and Bajorath (2000) used a descriptor based classification method. By placing the compounds into descriptor space, one should be able to discover clusters of compounds with the same biological activity and discover which part of descriptor space corresponds to a specific biological activity. Since the authors could use over 100 different descriptors, they aimed at simplifying descriptor space by using principal components. Dividing the principal component space into square boxes, compounds were grouped per box. If the compounds in one box had the same biological activity, the set of compounds in the box was called a pure class. If the box contained compounds of several biological activities, it was counted as a mixed class. Finally, if there was only one compound, the box was said to represent a singleton class.

The genetic algorithm was designed to solve three optimization problems simultaneously: 1) which descriptors should be used, 2) how many principal components should be used, and 3) into how many bins should every principal component be divided. The chromosome was a vector of 141 bits, 111 bits representing the use/non-use of particular descriptors, 15 bits to encode the number of principal components used, and 15 further bits to encode the number of boxes into which each principal component axis is divided. The fitness function increased with the number of pure classes and decreased with the number of mixed classes and singletons. This particular fitness measure might have been somewhat disadvantageous, since one ideally would want to reward a minimum of classes. It seems more desirable to have 7 pure classes than 700.

The best result had 4 principal components with 5 bins each and found 60 pure classes, 27 singletons and 2 mixed classes. The classification method therefore worked, though comparison with other classification methods and assessing quality via a separate validation set would have been valuable additions to this work.

Whereas ‘drug-likeness’ is a somewhat nebulous concept, there are also more sharply defined properties that are important for candidate drugs. One of these is water-solubility: if a compound does not dissolve in water, it cannot be transported by the blood to its desired site of action.

Wegner and Zell (2003) derived a quantitative structure-property relationship to predict water solubility of a molecule from its structure. The authors calculated 230 descriptors from these structures. Since there were 1016 molecules in the training set, using all descriptors would probably have led to overfitting. Therefore, the authors wanted to reduce the number of descriptors. Principal component analysis was regarded to yield non-intuitive results, so a genetic algorithm was created. The initial population of this genetic algorithm was seeded with maximally diverse individuals as selected by Shannon entropy measures and clique detection algorithms. The genetic algorithm for descriptor selection was very similar to the ones used in QSAR (see section 5): the chromosome was a vector of bits, each bit indicating the presence of a descriptor. The chromosomes could undergo one- and two-point crossover and bit-flip mutation. A neural network was trained using the descriptors that were indicated by the chromosomes, and the fitness of each chromosome was r^2 for the test set. The final best model had a validation set r^2 value of 0.82, which was comparable to the results of other neural networks trained on similar data sets that gave r^2 values between 0.79 and 0.91 for their validation sets.

A third factor to consider when designing drugs is metabolism, the breakdown of drugs by the body. Some of these breakdown products are toxic. For example, the drug paracetamol itself is harmless, but when taken in huge quantities it is partially transformed into a toxic product that causes liver damage. Drug designers therefore want to know the possible breakdown products of a compound, preferably before synthesis. Rules exist to predict metabolism, and computerized rule bases, like META (Klopman *et al.*, 1997) can be applied to automatically predict probable metabolites. However, a molecule can often be broken down in many different ways, and it is not clear which of those ways are preferred by the body. Assigning priorities to the diverse transforms is traditionally done by experts. Klopman *et al.* however investigated whether a genetic algorithm could do this automatically. This would be advantageous since it would eliminate the need to manually recalibrate all weights after adding new data.

The chromosomes contained the priorities of all reactions, coded as a vector of binary numbers. Crossover was one-point and only took place between the genes, and mutation was performed by flipping bits. Fitness was defined as the number of correct predictions minus the number of incorrect predictions, the false positives and false negatives. Bolzmann tournament selection outperformed normal tournament selection and roulette wheel selection, and was therefore chosen as selection method.

For both training and validation sets, the genetic algorithm found a better solution than the experts (table 2.1). Clearly, genetic algorithms can combine large amounts of data reliably into a rule-base, an exciting prospect.

Table 2.1: Comparison of the results of the genetic algorithm-set priorities versus the expert-set priorities.

	Test set			Validation set		
	True Positive	False Negative	False Positive	True Positive	False Negative	False Positive
Expert	103	45	28	66	9	56
GA	134	14	18	75	0	21

Biological activity of a compound can be predicted by computationally docking the molecule into the receptor. Often, however, the receptor structure is unknown. The common alternative is churning out high numbers of descriptors and using neural networks or multiple linear regression to find quantitative structure-activity relationships. However, this method does not use any of the knowledge available on receptors and their properties. A different possibility is making a model of the active site based on the ligand data. Walters and Hinds (1994) used this approach. Their computer program GERM (Genetically Evolved Receptor Models) uses a superposition of ligands, around which a collection of atoms, typically 50 to 60, is placed. These atoms represent the protein atoms of the active site. The interaction energy between the proposed active site and the ligands is calculated with a force field. The chromosome is the list of the atom types of the atoms in the reconstructed active site. One-point crossover and a mutation that randomly changes an atom type into a random other atom type were implemented. Crossover was the most important operator but allowing some mutation was found to improve the convergence.

When populations of 500-2000 chromosomes were allowed to evolve over up to 10000 generations, models with r^2 values of 0.90-0.99 were found. The average error for the compounds of the training set was 0.06, but for those in the validation set it was 0.40. This clearly points to overfitting. However, scrambling the bioactivity values indicates that there is also some real relationship behind the numbers: the mean r^2 value of the scrambled sets was only 0.34.

GERM has some drawbacks, however. First of all, the ligands of a receptor must be superimposed, and as has been discussed in section 4, there is no unambiguous method to do that. Overfitting is quite understandable in this system: after all, there are 50 to 60 different atoms involved for explaining the bioactivity of about 10 compounds. Interestingly, the atoms at some positions had identical types in all of the fittest individuals, at other positions there was much more variation. This would suggest some biological rationale behind the model, and it would certainly be interesting to make runs on superpositions of known ligands docked into the binding sites to see whether the conserved residues in the proposed active site correspond to the important groups in the real active site.

Some other programs used a methodology very similar to that of GERM. An example is PARM (Pseudo Atomic Receptor Model) by Chen *et al.* (1998). The main difference with GERM is that heuristics were used to initialize the chromosomes. When an atom type had to be assigned to a certain grid point, the heuristics increased the chance of choosing a negatively charged atom type if the ligand atoms near the grid point were positively charged, and vice versa.. Two training sets, 21 and 12 compounds, were used with validation sets whose sizes were about half as large. Crossvalidated r^2 values of 0.83-0.93 were reached. The compounds of the validation set were predicted with an average absolute error of 0.52; CoMFA analysis (section 5) yielded 0.61. This suggests that PARM can be somewhat better than conventional methods.

Vedani *et al.* (1998a,b) created another pseudoreceptor modeling method. The main deviations from Walters' method were the different and smaller set of pseudoreceptor atoms and incorporation of receptor flexibility. The latter means that the position of each pseudoreceptor atom is adapted for optimal interaction with each individual ligand. Therefore, if there are *n* small molecules in the training set, there are also *n* conformations of the pseudoreceptor. The authors considered this to be necessary to allow for changes in receptor conformation upon binding, especially regarding the direction of hydrogen bonds. However, movement of the pseudoreceptor atoms from their average position is penalized by decreasing the calculated binding energy.

Six different series of ligands were used, varying from compounds binding to the cannabinoid receptor to the β_2-adrenergic receptor and the sweet-taste receptor. Each series of ligands was split into a training set and a test set. The values of r^2 were smaller than those of Walters and Hinds (1994), ranging from 0.55 to 0.96. Root mean square errors of the training set were approximately 0.4, and for the test set 0.7.

The cause of the difference with Walters' research may be due to the reduced number of atom types available, the different force fields, the different test sets, other superpositions of the ligand, or receptor flexibility. These multiple changes make direct comparison between the models difficult. It is therefore unclear whether the pseudoreceptor models of the future will follow either of the two methods or will make use of new methodology, inspired by advances in superposition procedures and improvements in the entropy corrections and force fields of docking.

A final interesting application is the use of evolutionary algorithms to help create a good energy function for docking. Deng *et al.* (2004) used two sets of crystal structures of sizes 61 and 105 (of which external validation sets of size 10 resp. 6 were taken), and correlated the experimental binding energies with the presence of specific atom pairs. Since the authors distinguished 17 atom types and 5 relevant distance bins (1 Å wide between 1 and 6 Å) there were 5×17×17=1445 potentially relevant descriptors. In a first stage non-changing descriptors, highly correlated descriptors and 4-sigma outliers were removed deterministically, subsequently a genetic algorithm was used to select the best subset of descriptors. By reducing the number of descriptors used in the 105 compound data set from 456 to 20, the PLS-regression r^2 of the test set was increased from 0.43 to 0.60, and the r^2 of the prediction of the external test set even reached 0.64.

These results of Deng *et al.* can be compared to those of Morris *et al.* (1998), who developed a more traditional empirical free binding energy function using physicochemical knowledge, traditional force fields, and linear regression without feature selection. Since Morris *et al.* reached r^2 values of about 0.95 versus Deng's 0.64, it is clear that Deng's knowledge-based approach could profit from the physical and chemical knowledge that has been collected by experimental scientists. Incorporation of free energy loss due to loss of flexibility upon binding and the influence of direction upon the binding strength of hydrogen bonds would be obvious candidates to test for usefulness. Nevertheless, the increasing availability of crystal structures will make "knowledge-based" approaches more and more attractive to help refine the standard force field approaches.

Conclusion

The discussed publications make clear that there are probably many alternative areas in drug design in which evolutionary algorithms can be applied. Discovery of new, promising applications will most likely depend on the steady spread of knowledge and usage of evolutionary algorithms in the community of drug designers. Doubtlessly there are still many problems in drug design that can be at least partially solved with evolutionary algorithms, and many interesting applications may yet follow.

9. Conclusion: Evolutionary algorithms in drug design. Considering past, present and future.

Evolutionary algorithms have been applied in the field of drug design for over 10 years. In this review we have discussed their role in helping solve some of the problems of this field. Let us summarize and consider the findings so far.

The most important observation is that evolutionary algorithms are useful for drug design. This is, of course, necessarily a biased view since few authors would publish methods that do not work for their particular problem. However, the wide range of applications in which evolutionary algorithms found optimal or satisfactory solutions suggest that evolutionary algorithms are quite suitable for application to a wide range of problems in drug design, varying from conformational analysis to finding quantitative structure-activity relationships and performing *de novo* design.

This success has led to many applications of evolutionary algorithms, several of which have been incorporated into commercial packages. Some of the examples mentioned in this review are GFA (Genetic Function Approximation) which is now part of the molecular modeling package Cerius2, and the commercialized docking program GOLD. However, there is also other software that uses evolutionary algorithms, like the computer program Spartan that has procedures for evolutionary structure optimization. Doubtlessly there are several other software packages for drug design on the market in which evolutionary algorithms are a major or minor component. One could say that evolutionary algorithms have proven their worth and either already possess or at least approach the status of one of the standard optimization methods in drug design.

While over time evolutionary algorithms have been applied to more and more areas of drug design, one could also ask whether their performance in the diverse areas has also improved.

Looking at the different areas of application it is not clear whether the more recent implementations of evolutionary algorithms are more effective or efficient than the older versions. If any trends can be discerned, it is towards more complex evolutionary algorithms. Unfortunately it cannot be concluded with confidence that this increased complexity leads to improved performance due to the dissimilarity in test data sets, fitness functions and quality criteria used by the different authors.

The progress in the different fields can be summarized as follows:

-library design: Multiobjective fitness functions have been introduced. Calculations are getting more intricate and biologically relevant (2D/3D). Objective weighing of the

conflicting objectives, especially diversity and focusing, remains problematical.

-*conformational analysis:* The evolutionary algorithms have largely been superseded by the directed tweak algorithm, which is more specialized and seems somewhat more efficient than the current evolutionary algorithms.

-*quantitative structure-activity relationships:* The evolutionary algorithms seem to have grown more complex over the years. While some innovations have been introduced, notably the use of more complex functions of the descriptors, the novelty of most newer publications that involve evolutionary algorithms lies mainly in novel types of descriptors or the addition of other descriptor selection methods, not in the evolutionary algorithms themselves.

-*docking:* The evolutionary algorithms in this field have grown more elaborate and complex, however due to the absence of good test sets it is not clear whether this increased complexity has led to true progress. Experiments have indicated that Lamarckian evolutionary algorithms and island models are useful.

-*de novo design:* Evolutionary algorithms for structure manipulation have not seen significant advances since the work of Glen and Payne in 1995, however the fitness functions have improved from manually weighted parameters to docking. Also, the concept of "ease of synthesis" has been introduced, which is very important to ensure that the designed molecules can also be created in the laboratory.

Overviewing the past few years of application of evolutionary algorithms in drug discovery, one can conclude that a wide variety of chromosome representations, fitness functions and mutation operators have been developed for the different problems. The basic principles of evolutionary algorithms, however, still remain at the core of all these variants, and have proven themselves to be quite a robust and easily applied base of design for a range of optimization problems in this field. As the articles reviewed in this paper demonstrate, there are obviously a number of cases where evolutionary algorithms do not offer clear benefits over other methods such as Monte Carlo search, simulated annealing or deterministic optimization methods. There are also some cases where evolutionary algorithms achieve clearly satisfactory results and improvements over results that have been available so far. Obviously, a clear general conclusion cannot be drawn at this point, as there are no elaborate systematic comparisons of the different search methods available yet on the subareas of drug design covered in this paper.

It is interesting to note that almost all of the applications of evolutionary algorithms in drug design today use rather basic genetic algorithms, and thus fail to use self-adaptation capabilities. These are usually associated with evolution strategies and evolutionary programming (see e.g. (Bäck, 1996) for an in-depth discussion of this topic) but can also be used in genetic algorithms. Other developments in the field of evolutionary algorithms, such as estimation of distribution algorithms (used in bioinformatics by Saeys *et al.* (2004)) also have so far found no or only extremely sparse application in drug design. It would clearly be an interesting issue for future research to check whether these algorithmic techniques can deliver more convincing improvements over classical methods. While the success of novel techniques for optimization would clearly depend on the particular problem studied, the computer programs discussed in this review generally use quite basic algorithms, so chances are good that adding advanced techniques can improve their performance.

While optimization remains important, the current bottleneck in computational methods for drug design seems to be the fitness function, since this is often still either somewhat arbitrary, like manually weighing different measures of molecular similarity, useless, like rediscovering a known molecule, or inaccurate and too slow for extensive optimization, like docking. The biggest problem in current drug design seems to be calculating/predicting the relevant properties of a molecule, not finding more efficient optimization algorithms.

Where does that leave evolutionary algorithms? Should computer scientists be content by having added evolutionary algorithms to the standard toolbox of medicinal chemistry and move on, or is it still possible to do innovative and useful research in this area?

One reason to continue applying evolutionary algorithms to medicinal chemistry is that the collaboration between computer scientists and medicinal chemists itself can be fruitful. Medicinal chemists can profit from the knowledge of optimization methods and the experience in validation methods that computer scientists possess. Computer scientists may learn from the ideas and paradigms of medicinal chemists, which have resulted in diverse and ingenious forms of chromosomes and variation methods. These inventions could themselves be interesting concepts to be studied, improved, and possibly used for other problems by the computer scientists.

However, there are also reasons to believe that continued application and development of evolutionary algorithms could also be useful for the development of new and better computational methods for drug designers.

First, prediction methods in drug design are improving each year, therefore better fitness functions will become available. When they do, the existing evolutionary algorithms can be reapplied with greater success than before.

Second, there is the possibility of finding new application domains. Some of these might be known problems in medicinal chemistry or drug development that have not been tackled yet with evolutionary algorithms. Others would be the newly emerging fields, for example the genomic, proteomic and transcriptomic data which are becoming available and have to be processed, combined and modeled. Evolutionary algorithms may be useful in this process in some capacity. Though the generic nature of evolutionary algorithms makes them vulnerable to later replacement by hybrid algorithms or specialized optimization algorithms, like directed tweak, the simplicity and wide effectiveness of evolutionary algorithms makes them very suitable for pioneering new areas. If optimization yields clear benefits yet takes unacceptable amounts of computer power, the evolutionary algorithms may be replaced by more specialized methods.

Finally, the evolutionary algorithms that are currently used for drug design can be improved. Though the fitness functions are more important and time-critical, evolutionary algorithms that display more efficient convergence while searching as much of the search space would be very valuable. If an evolutionary algorithm needs fewer fitness evaluations for a good optimization, one can process and suggest more molecules in the same amount of computer time. Even with an inaccurate fitness function this collection will still give more “hits” than randomly screening, which would be very valuable to drug developers.

Yet how to achieve such progress? The key to this would lie in closer collaboration between medicinal chemists and computer scientists. Procedures that have become quite common in computer science, like having standard test sets on which an algorithm should work, should be used much more extensively in medicinal chemistry to help compare different methods and improvements in a method more objectively. The size and diversity of these sets should be sufficient to draw reliable and statistically significant conclusions when comparing different methods, the one to three test cases which have been used in some articles probably do not adequately reflect the diversity of cases in a particular domain. Publicly available reference fitness functions would also be necessary to compare different algorithms in a fair way.

The main contribution of medicinal chemists in this process would be the development and testing of heuristics. There are no shapeless, featureless problems in drug design; each problem has its own inherent, natural constraints. A generic evolutionary algorithm that is applicable in all cases and fails to take the information provided by the specific problem into account can be improved by including heuristics which make it less generally applicable, but more powerful for that specific application. Developing these heuristics and testing them critically would represent an advance in quality, which would be more reliable and systematic than the current independent tweaks of poorly compared algorithms which rarely take the biochemical nature of the problem space into account. As far as we understand the systems, we can develop heuristics. And as for the parts of the system we do not understand, we can observe, make hypotheses, test methods and learn from their results. This would result in speciation to optimally fill the diverse niches in drug design, and represent a true evolution of evolutionary algorithms in this field.

Looking over the past and current research, the challenges to create computational methods to predict reliably the biological properties of molecules are great indeed, and will take much time and intellectual effort to resolve. The current problems in developing new drugs indicate that the drug design process as well as the drugs themselves can only remain affordable if we can find ways to intelligently combine the growing available biological information with the possibilities of quickly and effectively searching the huge collection of drug-like molecules. This is a major challenge, but one in which evolving evolutionary algorithms can play an important role.

Glossary

ACTIVE SITE: the part of a protein which binds the messenger molecules or catalyzes the biochemical reactions.

CHROMOSOME: named after the strings of DNA which contain the genetic information of biological organisms, chromosomes in evolutionary algorithms are the data structures which contain the genetic information/genotype of one individual candidate solution. Often, especially in genetic algorithms, a chromosome is a vector of bits or numbers.

CROSSOVER: another name for the recombine-function of evolutionary algorithms.

HYDROGEN ACCEPTOR: oxygen or nitrogen atom in a molecule with a free electron pair that can bind to a hydrogen atom of a hydrogen donor.

HYDROGEN BOND: the attractive force between a hydrogen acceptor and the hydrogen atom of a hydrogen donor. Is generally the predominant binding force between a ligand and its receptor.

HYDROGEN DONOR: oxygen or nitrogen atom in a molecule that is bonded to a hydrogen atom. This hydrogen atom can bind to a free electron pair of a hydrogen acceptor.

LAMARCKIAN EVOLUTION: an individual's *phenotype* is optimized by a local search. The information of the phenotype is subsequently written back to the genotype, which then undergoes normal mutation/crossover.

LEAD (COMPOUND): a compound that seems to have a desirable biological activity and may be developed further into a drug.

LIGAND: a molecule that binds to a large biological molecule (usually a protein).

MOLECULE: a collection of atoms which are connected by bonds. On a simple level a molecule can thus be considered to be a graph in which the nodes are the atoms and the edges are the bonds. The specific physical and chemical restrictions on this graph are that each atom type has a maximum number of bonds, generally ranging from 1 to 4, and that the length of the bonds, the preferences for certain bond angles and finally the interplay of the attraction and repulsion between the atoms cause each molecule to assume a distinctive range of three-dimensional structures, called conformations.

POLING: optimization method developed by Smellie *et al.* (1995) that ensures diversity of the individuals in the population by modifying the fitness function in such a way that similarity to other individuals is penalized.

q^2: a measure of statistical significance. It is determined by leaving a subset of the data (often of size one) out of the training set, training a model with the remainder of the training set, and predicting the dependent variable of the subset. This is done for all items in the training set, the r^2 value of the resulting predictions is called the q^2. It is considered to be less sensitive to overfitting than r^2 and therefore a better measurement of the quality of a statistical model.

r^2: a measure of the statistical significance of a model. Its values are between 0 (no linear correlation between the independent and dependent variables) and 1 (a perfecty linear correlation between the independent and dependent variables). It can be calculated by comparing the values that a model gives (f_i) to the observed values (y_i) by the following formula: $r^2 \equiv 1 - \sum_i (y_i - f_i)^2 / \sum_i (y_i - \bar{y})^2$, where $\bar{y}$ is the means of the observed values.

REAGENT: a molecule that is used in a process in which it will react with another molecule, forming one or more new molecules, is called a reagent (derived from Latin: "something that must react").

ROULETTE WHEEL SELECTION: A method to select good individuals with higher probability than bad individuals as parents of the next generation. All members of the population are assigned a segment on a wheel, usually in proportion to their relative fitness. Subsequently random points on the wheel are selected and the corresponding population members become the parents of the next generation (Parrill, 2000).

SPECIATION: A large part of the population of individuals is very homogeneous: although there are officially many solutions, it is in reality just one solution with small variations. Usually one wants to prevent this and develop several solutions which differ significantly.

SPLINE FUNCTION: commonly written as f(*x*)=<*a*-*x*>. This function returns 0 if *x* is greater than *a*, and *a*-*x* if *x* is smaller than or equal to *a*.

TARGET (RECEPTOR): the biological macromolecule to which a drug or drug candidate should bind.

TORSION ANGLE: Angle indicating how much one end of a single bond is rotated with respect to the other end. For four bonded atoms A-B-C-D, the torsion angle of the bond B-C is defined as the angle which the C-D bond makes with the plane in which A, B and C lie.

TOURNAMENT SELECTION: A method to select the parents for the next generation of the evolution in an evolutionary algorithm. Tournament selection works by randomly picking a certain number of individuals out of the population and letting the best of them become a parent, repeating this process as often as is required (Parrill, 2000).

TRANSITION STATE: When a molecule is broken down by an enzyme, the enzyme first twists it into a strained conformation to make the subsequent reaction(s) easier. This strained state is called the transition state, since it is the phase a reacting molecule must go through in order to form the product.

VIRTUAL LIBRARY: A database of molecule structures.

References

Agrafiotis DK (2002) Multiobjective optimization of combinatorial libraries. Journal of Computer-Aided Molecular Design 16: 335-356.

Bäck T (1996) Evolutionary Algorithms in Theory and Practice, Oxford University Press, NY.

Bäck T, Fogel DB and Michalewicz Z (2000) Handbook of Evolutionary Computation, Vol. 1 and 2. Institute of Physics Publishing, Bristol, UK.

Bravi G, Green DVS, Hann MM and Leach AR (2000) PLUMS: a Program for the Rapid Optimization of Focused Libraries. Journal of Chemical Information and Computer Sciences 40: 1441-1448.

Chen H, Zhou J and Xie G (1998) PARM: A Genetic Evolved Algorithm To Predict Bioactivity. Journal of Chemical Information and Computer Sciences 38: 243-250.

Chen X, Rusinko A, Tropsha A and Young SS (1999) Automated Pharmacophore Identification for Large Chemical Data Sets. Journal of Chemical Informatics and Computer Sciences 39: 887-896.

Cho SJ and Hermsmeier MA (2002) Genetic Algorithm Guided Selection: Variable Selection and Subset Selection. Journal of Chemical Information and Computer Sciences 42: 927-936.

Clark DE, Jones G and Willett P (1994) Pharmacophoric Pattern Matching in Files of Three-Dimensional Chemical Structures: Comparison of Comformational-Searching Algorithms for Flexible Searching. Journal of Chemical Information and Computer Sciences 34: 197-206.

Clarck DE (ed) (2000) Evolutionary Algorithms in Molecular Design. Wiley-VCH, Weinheim.

David L, Luo R and Gilson MK (2001) Ligand-receptor docking with the Mining Minima optimizer. Journal of Computer-Aided Molecular Design 15: 157-171.

Deb K (2001) Multi-objective optimization using evolutionary algorithms, Wiley, New York.

Deng W, Breneman C and Embrechts MJ (2004) Predicting Protein-Ligand Binding Affinities Using Novel Geometrical Descriptors and Machine-Learning Methods, Journal of Chemical Information and Computer Sciences 44: 699-703.

DiMasi JA, Hansen RW and Grabowski HG (2003) The price of innovation: new estimates of drug development costs. Journal of Health Economics 22: 151-185.

Douguet D, Thoreau E and Grassy G (2000) A genetic algorithm for the automated generation of small organic molecules: Drug design using an evolutionary algorithm. Journal of Computer-Aided Molecular Design 14: 449-466.

Fogel LJ, Owens A, and Walsh M (1966) Artificial Intelligence through Simulated Evolution. Wiley, New York.

Fogel DB (1995): Evolutionary Computation – Toward a New Philosophy of Machine Intelligence. Addison-Wesley, Reading, MA.

Geladi P and Kowalski BR (1986) Partial least squares regression: a tutorial. Analytica Chimica Acta 185: 1-17.

Gillet VJ, Willett P and Bradshaw J (1998) Identification of Biological Activity Profiles Using Substructural Analysis and Genetic Algorithms. Journal of Chemical Information and Computer Sciences 38: 165-179.

Gillet VJ, Willett P, Bradshaw J and Green DVS (1999) Selecting Combinatorial Libraries to Optimize Diversity and Physical Properties. Journal of Chemical Information and Computer Sciences 39: 169-177.

Gillet VJ (2000) *De Novo* Molecular Design. In: Clark DE (ed) Evolutionary Algorithms in Molecular Design, pp. 49-69. Wiley-VCH, Weinheim.

Gillet VJ, Khatib W, Willett P, Fleming PJ and Green DVS (2002) Combinatorial Library Design Using a Multiobjective Genetic Algorithm. Journal of Chemical Information and Computer Sciences 42: 375-385.

Glen RC and Payne AWR (1995) A genetic algorithm for the automated generation of molecules within constraints. Journal of Computer-Aided Molecular Design 9: 181-202.

Globus A, Lawton J and Wipke T (1999) Automated molecular design using evolutionary techniques. Nanotechnology 10: 290-299.

Goldberg DE (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison Wesley.

Goldberg DE (2002) The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Kluwer, Boston.

Handschuh S, Wagener M and Gasteiger J (1998) Superposition of Three-Dimensional Chemical Structures Allowing for Conformational Flexibility by a Hybrid Method. Journal of Chemical Informatics and Computer Sciences 38: 220-232.

Hann MM, Leach AR and Harper G (2001) Molecular Complexity and Its Impact on the Probability of Finding Leads for Drug Discovery. Journal of Chemical Information and Computer Sciences 41: 856-864.

Hart WE (1999) Comparing Evolutionary Programs and Evolutionary Pattern Search Algorithms: A Drug Docking Application. In: Banzhaf W, Daida J, Eiben AE, Garzon MH, Honavar V, Jakiela M and Smith RE (eds) Proceedings of the Genetic and Evolutionary Computation Conference, Orlando, Florida, USA, July 13-17 1999. Morgan Kaufmann, pp. 855-862.

Hart WE, Rosin C, Belew RK and Morris GM (2000) Improved Evolutionary Hybrids for Flexible Ligand Docking in Autodock. In: Floudas CA and Pardalos PM (eds) Optimization in Computational Chemistry and Molecular Biology, Pinceton, USA May 7-9 1999. In: Nonconvex Optimization and its Applications (vol 40) Kluwer Academic Publishers, Dordrecht, the Netherlands, pp 209-230.

Hasegawa K, Kimura T and Funatsu K (1999) GA Strategy for Variable Selection in QSAR Studies: GA-Based Region Selection to a 3D-QSAR Study of Acetylcholinesterase Inhibitors. Journal of Chemical Information and Computer Sciences 39: 112-120.

Hemmateenejad B, Akhond M, Miri R and Shamsipur M (2003) Genetic Algorithm Applied to the Selection of Factors in Principal Component-Artificial Neural Networks: Application to SAR Study of Calcium Channel Antagonist Activity of 1,4-Dihydropyridines (Nifedipine Analogous). Journal of Chemical Information and Computer Sciences 43: 1328-1334.

Holland JH (1975) Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor.

Holliday JD and Willett P (1997) Using a genetic algorithm to identify common structural features in sets of ligands. Journal of Molecule Graphics and Modelling 15: 221-232.

Jin AY, Leung FY and Weaver DF (1999) Three Variations of Genetic Algorithm for Searching Biomolecular Conformation Space: Comparison of GAP 1.0, 2.0 and 3.0. Journal of Computational Chemistry 20: 1329-1342.

Jones G, Willett P, Glen RC, Leach AR and Taylor R (1997) Development and Validation of a Genetic Algorithm for Flexible Docking. Journal of Molecular Biology 267: 727-748.

Kamphausen S, Höltge N, Wirsching F, Morys-Wortmann C, Riester D, Goetz R, Thürk M and Schwienhorst A (2002) Genetic algorithm for the design of molecules with desired properties. Journal of Computer-Aided Molecular Design 16: 551-567.

Kimura T, Hasegawa K and Funatsu K (1998) GA Strategy for Variable Selection in QSAR Studies: GA-Based Region Selection for CoMFA Modeling. Journal of Chemical Information and Computer Sciences 38: 276-282.

Klopman G, Tu M and Talafous J (1997) META. 3. A Genetic Algorithm for Metabolic Transform Priorities Optimization. Journal of Chemical Information and Computer Sciences 37: 329-334.

Koza JR (1992) Genetic Programming: On the Programming of Computers by Natural Selection. MIT Press, Cambridge, MA.

Koza JR, Keane MA, Streeter MJ, Mydlowac W, Yu J and Lanza G (2003) Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Kluwer, Boston.

Lipinski CA, Lombardo F, Dominy BW and Feeney PJ (1997) Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. Advanced Drug Delivery Reviews 23: 3-25.

Liu DX, Jiang HL, Chen KX and Ji RY (1998) A New Approach to Design Virtual Combinatorial Library with Genetic Algorithm Based on 3D Grid Property. Journal of Chemical Information and Computer Sciences 38: 233-242.

Lučić B, Nadramija D, Bašic I and Trinajstić N (2003) Toward Generating Simpler QSAR Models: Nonlinear Multivariate Regression versus Several Neural Network Ensembles and Some Related Methods. Journal of Chemical Information and Computer Sciences 43: 1094-1102.

Mekenyan O, Dimitrov D, Nikolova N and Karabunarliev S (1999) Conformational Coverage by a Genetic Algorithm. Journal of Chemical Information and Computer Sciences 39: 997-1016.

Morris GM, Goodsell DS, Halliday RS, Huey R, Hart WE, Belew RK and Olson AJ (1998) Automated Docking Using a Lamarckian Genetic Algorithm and an Empirical Binding Free Energy Function. Journal of Computational Chemistry 19: 1639-1662.

Nachbar RB (1998) Molecular Evolution: A Hierarchical Representation for Chemical Topology and Its Automated Manipulation. In: Koza JR, Banzhaf W, Chellapilla K, Deb K, Dorigo M, Fogel DB, Garzon MH, Goldberg DE., Iba H, and Riolo RL (eds) Genetic Programming 1998: Proceedings of the Third Annual Conference, University of Wisconsin, Madison, Wisconsin. 22-25 July 1998, San Francisco, CA: Morgan Kaufmann, pp. 246-253.

Nachbar RB (2000) Molecular Evolution: Automated Manipulation of Hierarchical Chemical Topology and Its Application to Average Molecular Structures. Genetic Programming and Evolvable Machines 1: 57-94.

Nair N and Goodman JM (1998) Genetic Algorithms in Conformational Analysis. Journal of Chemical Information and Computer Sciences 38: 317-320.

Parascondola J (1980) Early efforts to relate structure and activity. Trends in Pharmacological Sciences 1: 417-419.

Parrill AL (2000) Introduction to Evolutionary Algorithms. In: Clark DE (ed) Evolutionary Algorithms in Molecular Design, pp. 49-69. Wiley-VCH, Weinheim.

Patel S, Stott IP, Bhakoo M and Elliott P (1998) Patenting computer-designed peptides. Journal of Computer-Aided Molecular Design 12: 543-556.

Payne AWR and Glen RC (1993) Molecule recognition using a binary genetic search algorithm. Journal of Molecule Graphics 11: 74-91.

Pegg SC-H, Haresco JJ and Kuntz ID (2001) A genetic algorithm for structure-based de novo design. Journal of Computer-Aided Molecular Design 15: 911-933.

Pritchard JF, Jurima-Romet M, Reimer MLJ, Mortimer E, Rolfe B and Cayen MN (2003) Making better drugs: decision gates in non-clinical drug development. Nature Reviews Drug Discovery 2: 542-553.

Rechenberg I (1973) Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann-Holzboog, Stuttgart.

Rechenberg I (1994) Evolutionsstrategie ´94. Frommann-Holzboog, Stuttgart.

Rees P (2003) Big pharma learns how to love IT. Scientific Computing World : 16-18.

Rogers D and Hopfinger AJ (1994) Application of Genetic Function Approximation to Quantitative Structure-Activity Relationships and Quantitative Structure-Property Relationships. Journal of Chemical Information and Computer Sciences 34: 854-866.

Saeys Y, Degroeve S, Aeyels D, Rouzé P and Van de Peer Y (2004) Feature selection for splice site prediction: A new method using EDA-based feature ranking. BMC Bioinformatics 5: 64.

Schneider G, Schrödl W, Wallukat G, Müller J, Nissen E, Rönspeck W, Wrede P and Kunze R (1998) Peptide design by artificial neural networks and computer-based evolutionary search. Proceedings of the National Academy of Sciences of the United States of America 95: 12179-12184.

Schneider G, Lee M-L, Stahl M and Schneider P (2000a) De novo design of molecular architectures by evolutionary assemly of drug-derived building blocks. Journal of Computer-Aided Molecular Design 14: 487-494.

Schneider G, Clément-Chomienne O, Hilfiger L, Schneider P, Kirsch S, Böhm H-J and Neidhart W (2000b) Virtual Screening for Bioactive Molecules by Evolutionary De Novo Design. Angewandte Chemie International Edition 39: 4130-4133.

Schwefel H-P (1977) Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie, volume 26 of Interdisciplinary Systems Research. Birkhäuser, Basel.

Schwefel H-P (1995) Evolution and Optimum Seeking. Wiley, New York.

Selwood DL, Livingstone DJ, Comley JCW, O'Dowd AB, Hudson AT, Jackson P, Jandu KS, Rose VS and Stables JN (1990) Structure-Activity relationships of Antifilarial Antimycin Analogues: A Multivariate Pattern Recognition Study. Journal of Medicinal Chemistry 33: 136-142.

Sheridan RP, SanFeliciano SG and Kearsley SK (2000) Designing targeted libraries with genetic algorithms. Journal of Molecular Graphics and Modelling 18: 320-334.

Shi LM, Fan Y, Myers TG, O'Connor PM, Paull KD, Friend SH and Weinstein JN (1998) Mining the NCI Anticancer Drug Discovery Databases: Genetic Function Approximation for the QSAR Study of Anticancer Ellipticine Analogues. Journal of Chemical Information and Computer Sciences 38: 189-199.

Smellie A, Teig SL and Towbin P (1995) Poling: Promoting Conformational Variation. Journal of Computational Chemistry 16: 171-187.

So S-S and Karplus M (1996) Evolutionary Optimization in Quantitative Structure-Activity Relationship: an Application of Genetic Neural Networks. Journal of Medicinal Chemistry 39: 1521-1530.

So S-S (2000) Quantitative Structure-Activity Relationships. In: Clark DE (ed) Evolutionary Algorithms in Molecular Design, pp. 71-97, John Wiley and Sons, New York.

Taylor RD, Jewsbury PJ and Essex JW (2002) A review of protein-small molecule docking methods. Journal of Computer-Aided Molecular Design 16: 151-166.

Thomsen R (2003) Flexible ligand docking using evolutionary algorithms: investigating the effects of variation operators and local search hybrids. Biosystems 71: 57-73.

Thormann M and Pons M (2001) Massive Docking of Flexible Ligands Using Environmental Niches in Parallelized Genetic Algorithms. Journal of Computational Chemistry 22: 1971-1982.

Ting A, McGuire R, Johnson AP and Green S (2000) Expert System Assisted Pharmacophore Identification. Journal of Chemical Informatics and Computer Sciences 40: 347-353.

Tufféry P, Etchebest C, Hazout S and Lavery R (1993) A Critical Comparison of Search Algorithms Applied to the Optimization of Protein Side-Chain Conformations. Journal of Computational Chemistry 14: 790-798.

Vedani A, Dobler M and Zbinden P (1998a) Quasi-Atomistic Receptor Surface Models: A Bridge between 3-D QSAR and Receptor Modeling. Journal of the American Chemical Society 120: 4471-4477.

Vedani A and Zbinden P (1998b) Quasi-atomistic receptor modeling A bridge between 3D QSAR and receptor fitting. Pharmaceutica Acta Helvetiae 73: 11-18.

Vieth M, Hirst JD, Dominy BN, Daigler H and Brooks CL (1998) Assessing Search Strategies for Flexible Docking. Journal of Computational Chemistry 19: 1623-1631.

Vinkers MH, De Jonge MR, Daeyaert FFD, Heeres J, Koymans LMH, Van Lenthe JH, Lewi PJ, Timmerman H, Van Aken K, and Janssen PAJ (2003) SYNOPSIS: SYNthesize and Optimize System in Silico. Journal of Medicinal Chemistry 46: 2765-2773.

Waller CL and Bradley MP (1999) Development and Validation of a Novel Variable Selection Technique with Application to Multidimensional Quantitative Structure-Activity Relationship Studies. Journal of Chemical Information and Computer Sciences 39: 345-355.

Walters DE and Hinds RM (1994) Genetically Evolved Receptor Models: A Computational Approach to Construction of Receptor Models. Journal of Medicinal Chemistry 37: 2527-2536.

Wegner JK and Zell A (2003) Prediction of Aqueous Solubility and Partition Coefficient Optimized by a Genetic Algorithm Based Descriptor Selection Method. Journal of Chemical Information and Computer Sciences 43: 1077-1084.

Wehrens R, Pretsch E and Buydens LMC (1998) Quality Criteria of Genetic Algorithms for Structure Optimization. Journal of Chemical Information and Computer Sciences 38: 151-157.

Weininger D (1988) SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. Journal of Chemical Information and Computer Sciences 28: 31-36.

Westhead DR, Clark DE, Frenkel D, Li J, Murray CW, Robson B and Waszkowycz B (1995) PRO_LIGAND: An approach to de novo molecular design. 3. A genetic algorithm for structure refinement. Journal of Computer-Aided Molecular Design 9: 139-148.

Xue L and Bajorath J (2000) Molecular Descriptors for Effective Classification of Biologically Active Compounds Based on Principal Component Analysis Identified by a Genetic Algorithm. Journal of Chemical Information and Computer Sciences 40: 801-809.

Yang J-M and Kao C-Y (2000) Flexible Ligand Docking Using a Robust Evolutionary Algorithm. Journal of Computational Chemistry 21: 988-998.

Yasri A and Hartsough D (2001) Toward an Optimal Procedure for Variable Selection and QSAR Model Building. Journal of Chemical Information and Computer Sciences 41: 1218-1227.

3 Mining a Chemical Database for Fragment Co-occurrence: Discovery of "Chemical Clichés"

Eric-Wubbo Lameijer[†], *Joost N. Kok*[‡], *Thomas Bäck*[‡,#], *Ad P. IJzerman*[†]

[†]Division of Medicinal Chemistry, Leiden/Amsterdam Center for Drug Research, Leiden University, Einsteinweg 55, 2300 RA Leiden, the Netherlands.

[‡]Leiden Institute of Advanced Computer Science (LIACS), Leiden University, Niels Bohrweg 1, 2333 CA Leiden, the Netherlands.

[#]NuTech Solutions, Martin-Schmeisser-Weg 15, 44227 Dortmund, Germany.

This chapter was first published in the Journal of Chemical Information and Modeling, reference: Lameijer, E.W.; Kok, J.N.; Bäck, T.; IJzerman, A.P. Mining a Chemical Database for Fragment Co-occurrence: Discovery of "Chemical Clichés". *J. Chem. Inf. Model.*, **2006**, *46* (2), 553–562.

Abstract

Nowadays millions of different compounds are known, their structures stored in electronic databases. Analysis of these data could yield valuable insights into the laws of chemistry and the habits of chemists. We have therefore explored the public database of the National Cancer Institute (>250,000 compounds) by pattern searching. We split the molecules of this database into fragments to find out which fragments exist, how frequent they are and whether the occurrence of one fragment in a molecule is related to the occurrence of another, non-overlapping fragment. It turns out that some fragments and combinations of fragments are so frequent that they can be called "chemical clichés". We believe that the fragment data can give insight into the chemical space explored so far by synthesis. The lists of fragments and their (co-) occurrences can help create novel chemical compounds by i) systematically listing the most popular and therefore most easily used substituents and ring systems for

synthesizing new compounds, by ii) being an easily accessible repository for rarer fragments suitable for lead compound optimization, and by iii) pointing out some of the yet unexplored parts of chemical space.

Introduction

Over the last two centuries, chemists have synthesized many millions of structures. Two of the largest chemical databases, the Beilstein[1] and the CAS[2], contain over 8 million and 25 million compounds, respectively. Such large collections of compounds could give much insight into chemistry, both into what kinds of compounds can be made with current chemical technology, and into which parts of chemical space have been extensively investigated or, conversely, barely explored.

In most investigations so far, databases have been used for classification of compounds: for example, when is a compound drug-like[3,4,5]? Or which substructures or properties correlate with mutagenicity[6,7]?

Relatively unexplored are the possibilities of general databases: collections of molecules for which no data other than their structure is available. The only investigations known to us which had this purpose[8,9] did so to extract a catalogue of rings respectively substituents for drug designers. However, it seems desirable to get more information out of those general databases than just rings. We could also learn much about chemistry and the habits of chemists by studying which substructures and substructure combinations occur.

In this investigation we want to delve deeper into the knowledge stored in the molecular structures. This would not only give us an extensive catalogue of fragments to re-use in synthesis of drugs and other compounds, but also insight into "chemical habits". What kinds of compounds are made frequently, and which substructures are relatively rarely found together in a molecule? Some of these rare combinations might indicate barely explored parts of chemical space, potentially interesting for designing new compounds.

In this work we will use the name "chemical clichés" for some of the most-occurring fragments and frequently co-occurring pairs of fragments. The word "cliché" originated in the French printing industry where it denoted a stereotype, a kind of stamp of for example a picture that was pressed on the paper to produce the same image many times. Nowadays the word cliché is mainly used to denote a trite expression, such as "missed by a mile" or "top research institute". However, some classes of chemical compounds also seem to be based on the same "stamp" with only slight variations, such as benzodiazepines and tricyclic antidepressants. Also, single fragments like the benzene ring can occur extremely often in molecules. We think that the word “cliché” is useful to describe this reuse of ideas, while stressing that the current templates might not be the only ones that are viable.

Then the question remains how to extract knowledge from chemical databases. Knowledge is usually found in occurrences and patterns: what occurs in nature, what does not occur, and which events occur together? What is correlated with what? Looking at the molecular structure as a whole is not very useful, since all chemical structures in a database are unique, so on that level they are incomparable. Splitting the molecules into the chemically smallest fragments, the atoms, will yield no more information than the periodic table. One should therefore look for chemical knowledge at a level between these extremes, and that is the level of the molecular fragments.

In this paper we first discuss our choice of fragmentation method. Then we will describe the method we used to detect whether two different fragments co-occur more or less often than expected, and thereafter we will present the results of the fragment mining and co-occurrence analysis. We will conclude with a discussion of our findings, suggestions for application of the data obtained, and directions for future investigations.

Database

In this investigation we used the public database of the United States National Cancer Institute (NCI). The August 2000 version, which we mined, contains 250251 structures[10]. The molecules in this database have been selected to be tested against cancer, so were deemed by the database compilers to possibly have biological activity. Since many of the compounds are experimental and have not been tested on bioavailability and safety in humans, the diversity in structures is quite large, and should give a decent cross section of the range and preferences of chemical synthesis.

Fragmentation method

To find patterns in structures, the first step is to break the molecules into fragments. Two categories of fragmentation methods can be distinguished: the “full substructure set”, in which all possible 1, 2, 3 …n-atom sized substructures of a molecule are detected, and “molecule parts”, in which a molecule is divided into a number of non-overlapping substructures.

While the full substructure set would give all information possible, in practice it yields huge numbers of substructures per molecule (several thousands for even a medium-sized molecule). This makes such kind of data mining computationally very expensive, especially for large collections of compounds. For an exploratory study such as this, a "molecule parts" method would be more suitable, since there are fewer parts, they do not overlap, and they correspond to chemically intuitive units.

The next question is at which point to "break" the molecules. The two main methods here are graph splitting[11,12,13] and virtual retrosynthesis[14,15,16]. Graph splitting breaks molecules at topologically interesting points, such as the bond between a substituent and a ring, while virtual retrosynthesis uses specific rules based on chemical reactions, and breaks for example ester bonds. Both methods yield manageable sets of substructures (10,000-100,000 for a medium sized database). From a chemical point of view, the retrosynthesis method seems most logical, however it does not reflect actual syntheses very well as for example Vinkers et al. found out[14]. The reason is that chemical reactivity depends on steric and electronic factors which are for a large part determined outside the three or four atoms of the "breakable bond" and its neighbors. Conversely, synthesis can often create bonds (such as alkane C-C bonds) which are not considered to be cleavable by most retrosynthesis algorithms, because typically only a few dozens of the hundreds of organic reactions are incorporated into the software. A last disadvantage is that different retrosynthetic rules give different fragment sets. In contrast, graph splitting is quite reproducible, easy to implement and divides structures in chemically intuitive units of "ring systems" and substituents. This is why we chose the graph splitting method.

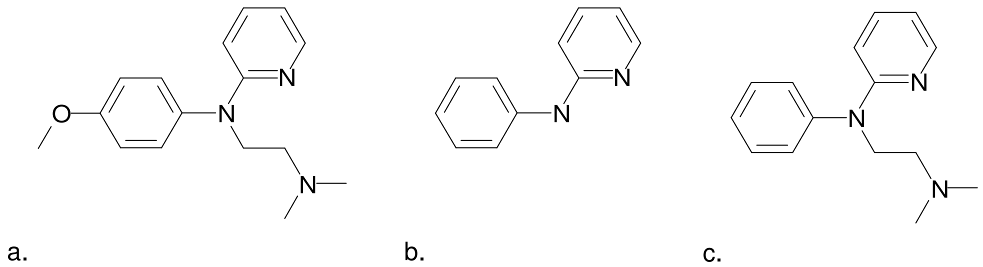


Figure 3.1: a. A drug molecule (pyrilamine) b. Framework of pyrilamine according to Bemis et al.[11], consisting of only the ring systems and the atoms that directly connect the ring systems. c. Framework of pyrilamine according to our definition so without substituents attached to the rings. This framework is later split into ring systems and linkers.

Deciding to do graph splitting is however not enough, since graph splitting can be done in a number of different ways. Bemis et al.[11] iteratively cut off all 1-connected atoms, so that only "substituents" and frameworks were left, the frameworks being the ring systems with the part of the linkers that directly connect them, see Figure 3.1.

We decided to go one step further and also split up the frameworks into ring systems and linkers. We therefore ended up with splitting molecules into substituents, ring systems, and linkers of different orders (linking two ring systems, three ring systems, etc.) See Figure 3.2 for an illustration of our fragment classes and the decomposition of an example molecule, folic acid.

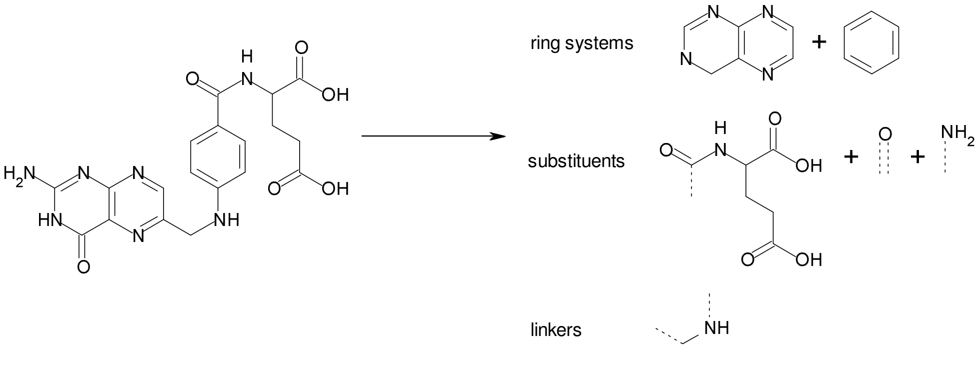


Figure 3.2: Our algorithm breaks the bonds between ring systems and the rest of the molecule, and thereby splits the molecule (in this example folic acid) into several types of fragments: ring systems, substituents, and linkers.

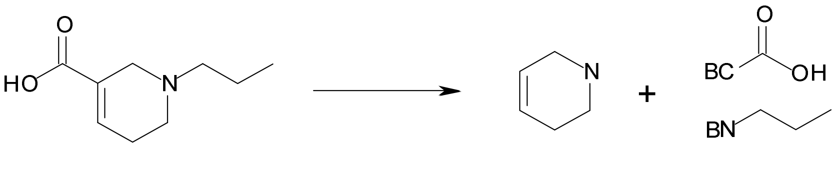


Figure 3.3: Storage format of ring systems and non-ring systems. While ring systems are stored as normal molecules, substituents and linkers include one or more "branching atoms" that encode the symbols of the atoms to which the substituent or linker is attached.

The ring structures were stored as normal molecules, only without hydrogen atoms (similar to the format of the NCI database itself). For the substituents and linkers we considered it useful, like Bemis et al.[12], to encode which atoms of the substituent/linker bind to the ring systems, as well as to which atom types they bind. We therefore encoded the ring attachment atoms as special types, the "BX" atoms, where X was the elemental symbol of the ring atom to which the substituent was attached. This encoding is illustrated in Figure 3.3.

Splitting molecules in this way can already yield useful information, such as which ring systems occur, and which do not (like an N_6-ring). But we could get even more information by also recording the frequencies of the substructures, as this would allow us to analyze frequency distributions and to explain why some fragments are more prevalent than others.

As a practical point, we had to find a method to encode the fragments uniquely, so that of each fragment mined we could determine whether it was already in the database or of a new fragment type. For this problem (the so-called "canonicalisation problem") various algorithms and notations have been developed, such as Unique SMILES[17]. In this investigation, we implemented a canonical code of which the first part included the number of atoms, the number of rings (in the case of ring systems) and the number of attachment points (in the case of substituents and linkers). The second part contained the atoms, which were sorted first on their number of neighbors and the number of bonds of those neighbors, second on atom type and finally on hybridization (sp, sp^2, sp^3). Since the fragments were relatively small, this simple method worked well.

Co-occurrence analysis

After the entire molecule database was split into fragments we performed a co-occurrence analysis: which fragments were unexpectedly often found together, and which seemed to "avoid" each other?

Of course, two fragments that are frequent would occur much more often together than two infrequent fragments; however, that would not necessarily mean that there is a relationship between the two. Therefore we decided to do a stochastic experiment.

First, we selected those fragments which occurred in more than 20 molecules (in order to obtain statistically significant and chemically useful results). Then we "simulated" an NCI database by randomly dividing the different fragments over as many "molecules" (bins) as they were part of in the real database. So if a certain fragment occurred in 500 molecules in the original database, it was divided over 500 randomly chosen bins of the 250,251 available in the simulated database. We counted how often each combination of fragments occurred, and repeated the simulation a thousand times. These results were compared with the co-occurrence counts of the NCI database.

For example, if we had taken a database of 1000 molecules, in which 500 phenyl rings occur and 100 methyl groups, there would be on the average 50 co-occurrences of these two groups. An experiment would find that they co-occurred together about 50 times, with a standard deviation (SD) of about 4.7. However, if in the real database they occur 75 times together, which is 5.3 SD from the expected value, this indicates a correlation, which may have synthetic and/or biological reasons.

Results of fragment finding

The mined NCI database contained 250251 compounds. These molecules were split into ring systems, substituents, and several types of linkers, in total 13509 different ring systems and 52103 different non-ring fragments. Of these non-rings 19602 were unconnected fragments, mostly anions such as sulfate and molecules without rings. More interesting were the other non-ring fragments: 18015 were substituents, 9675 linked two ring systems, 2531 linked three ring systems and 2280 linked four or more ring systems. The most highly connected linker was attached to eighteen ring systems. The number of different fragments in the largest categories, as well as the total occurrences in the molecules and some example fragments are shown in Table 3.1.

Visual inspection of the fragments and their occurrences led to the following observations, some of which were already known qualitatively, but which could now be confirmed quantitatively through the data mining:

1) Many of the ring systems and branches contain metal atoms or metallic atoms such as boron. In the case of rings, 2722 out of 13509 (20%) contained atoms other than C, N, O, and S, such as As, Fe, B and Si. Of the substituents, 1736 out of 18015 contained atoms other than C, N, O, S, and the halogens, less than 10%. The two and three-connected branches had 11% and 24% respectively, while most linkers with six or more attachment points contained metals or less common heteroatoms (B, P, Si, and such).
2) In general, the larger the ring or branch, the smaller its frequency seems to be.
3) Metals and higher-weight non-metallic elements both occur relatively rarely in fragments and make a ring or branch occur less often. Carbon atoms dominate rings and other fragments, followed by nitrogen and oxygen, which are in turn more prevalent than sulfur, phosphorus, and finally the metals.

4) In branches, a higher number of attachment points seems to mean that it is less used. Bemis et al.[11] did not find any frequent frameworks with 3-linkers or higher-order linkers in the Comprehensive Medicinal Chemistry (CMC) database[18]. Tables 3.1 and 3.2 confirm this observation.
5) The only exception to the rule that the more attachment points a linker has, the less frequent it is, is going from 5-attached linkers to 6-attached linkers. Inspection of the structures of the 6-linkers shows that most of them are symmetrical and therefore possibly easier to synthesize. However, 6 and multiples of 6-linkers are uncommonly popular (Table 2; compare 12 to 11 and 13, 18 to 17) – probably this is due to metal complexes and the high symmetry possible (both 2- and 3 fold symmetrical). Perhaps investigations of larger databases could confirm whether there really is a "rule of six".
6) The ratio of the occurrence of fragments to the number of unique fragments decreases as one goes from substituents to rings to linkers. The ring ratio (13509 unique rings, together occurring 416867 times in the database) is 31, for the 18374 substituents it is 33, for the 2-linkers 10, for the 3-linkers 3.2 times and for the four-linkers 2.5. It may be that the more unique fragments there are in a category, the more lopsided the distribution will be.

As illustration of our results, the top ten fragments of the most common fragment families are shown in Table 3.3.

Results of the co-occurrence analysis

Our investigations found 65612 fragment types in 250251 molecules. Correlating all these fragments to each other would have resulted in about 4 billion correlations, but most of these would be meaningless since about 70% of fragments occurs only once in the database. To reduce computational cost and find only the co-occurrences in a decently sized set of molecules (we set the threshold somewhat arbitrarily to at least 20 molecules), we only calculated co-occurrences for fragments which occurred in 20 or more molecules – 1895 fragments, just 2.9% of the total. Among these fragments were also some metal-containing and therefore less interesting fragments, which we removed. The final set therefore contained 1730 different fragments, 2.6% of the total number.

We created one thousand simulated databases, as described in the methods section, and calculated the expected occurrence of each pair of fragments, as well as the standard deviation of these co-occurrences (for among different simulations, the

Table 3.1: Overview of some of the fragment databases we created by fragmenting the NCI database. For example, the database of substituents (groups attached to only one ring system) contains 18,374 different types of substituents, which together occur 617,722 times in the NCI database. Also given is an example of the particular type of fragment, for the substituents this is the methyl group attached to a carbon atom in a ring system.

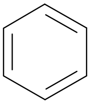
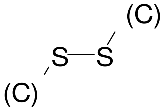
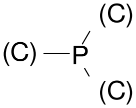
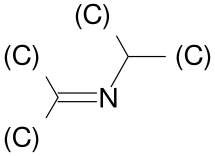
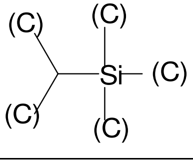
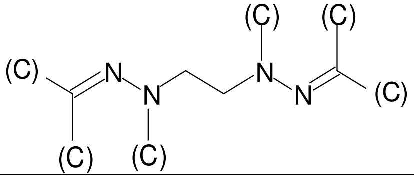
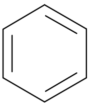
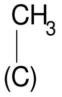
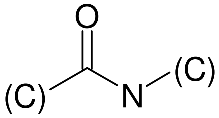
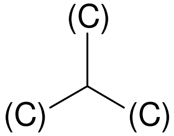
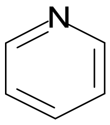
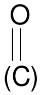
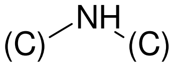
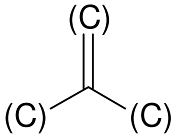
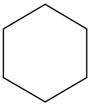
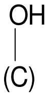
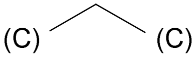
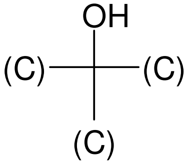
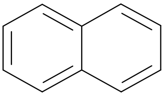
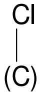
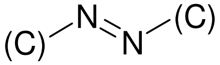
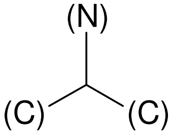
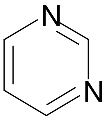
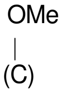
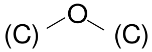
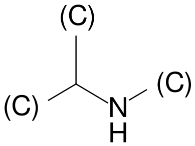
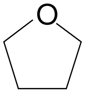
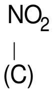
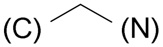
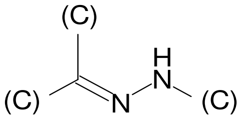
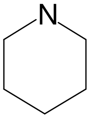
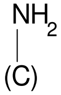
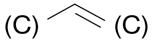
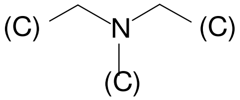
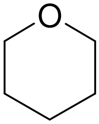
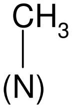
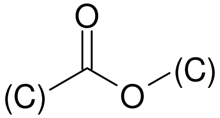
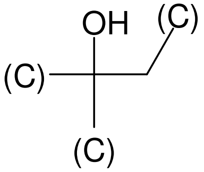
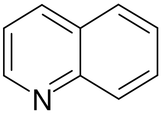
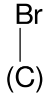
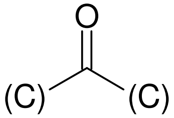
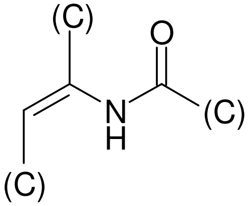
	Number of unique fragments	Number in database	Example
Ring systems	13509	416867	
Substituents	18374	617722	(C) —CH_3
2-linkers	9990	101402	
3-linkers	2602	5776	
4-linkers	974	2388	
5-linkers	126	177	
6-linkers	218	280	

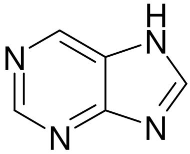
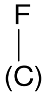
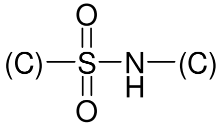
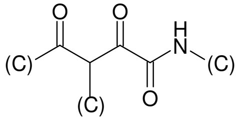
Table 3.2: Overview of the number of fragments linking seven or more ring systems. For example, there are 15 unique fragments which are attached to nine ring systems simultaneously.

Number of attachment points	7	8	9	10	11	12	13	14	15	16	17	18
Number of fragment types	22	15	15	1	1	26	0	0	0	0	0	2

Table 3.3: The most frequently occurring fragments in the NCI database. Listed are the top ten ring systems, substituents, 2-linkers and 3-linkers. The numbers in each cell represent the number of occurrences of the fragment in the database and how many percent this is of all occurrences of fragments of its type. For example, the keto substituent (C=O) occurs 77,907 times in the database and thereby represents 13% of all substituent occurrences; if one would randomly pick a substituent from a molecule, the chance is 13% that it is a keto group. The numbers in the bottom row are the total percentage of the top ten fragments.

	Ring systems	substituents	2-linkers	3-linkers
1	 200644 (48%)	 102157 (17%)	 4484 (4.4%)	 415 (5.0%)
2	 11442 (2.7%)	 77907 (13%)	 4266 (4.2%)	 384 (4.6%)

3	 7731 (1.9%)	 62949 (10%)	 3873, 3.8%	 240 (2.9%)
4	 6991 (1.7%)	 46734 (7.6%)	 3695 (3.6%)	 179 (2.1%)
5	 6185 (1.5%)	 42080 (6.8%)	 3638 (3.6%)	 107 (1.3%)
6	 5814 (1.4%)	 24997 (4.0%)	 2940 (2.9%)	 88 (1.1%)
7	 5352 (1.3%)	 19421 (3.1%)	 2699 (2.7%)	 87 (1.0%)
8	 5120 (1.2%)	 14654 (2.4%)	 2542 (2.5%)	 87 (1.0%)
9	 4526 (1.1%)	 3184 (1.8 %)	 2278 (2.2%)	 80 (0.95%)

10	 3991 (0.96%)	 7916 (1.3%)	 1966 (1.9%)	 76 (0.91%)
Total	62%	66%	32%	21%

number of co-occurrences of a pair of fragments can vary). Then we compared the expected co-occurrences and the deviation in SDs (the z-values) to the real co-occurrences in the NCI database.

The distribution of z-values of a sample simulated database yielded a Gaussian like distribution, as expected (Figure 3.4). The distribution of fragment co-occurrences in the NCI database in the same figure is, however, remarkably different.

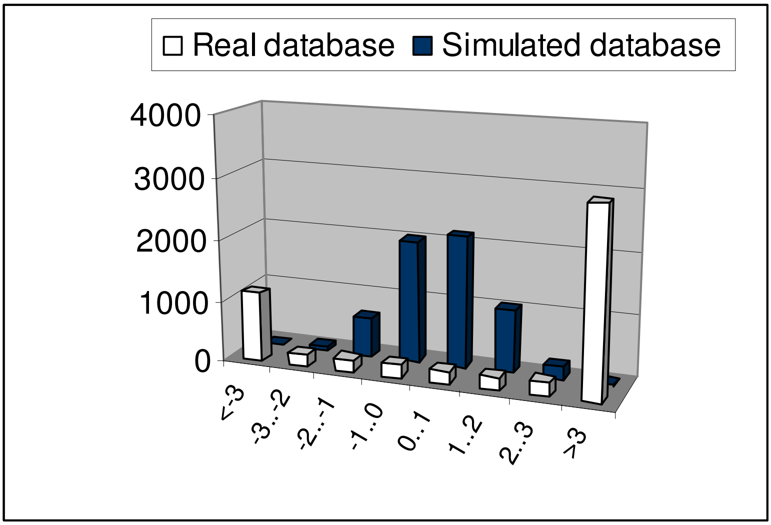
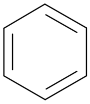
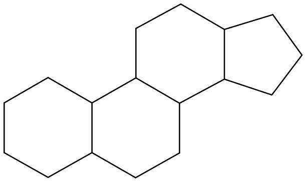
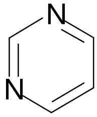
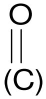
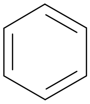
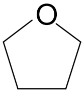
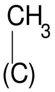
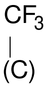
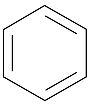
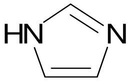
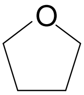
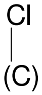


Figure 3.4: Overview of the number of standard deviations that the real co-occurrence of a fragment pair differs from how the co-occurrence would be if the distribution of fragments over the molecules in the database were random. The X axis displays the deviation range, the Y-axis the number of pairs in a certain range. Only pairs which occur over 20 times are counted here. The distribution of the simulated (random) database is Gaussian-like, but the real database has lots of fragments co-occurring much more or much less frequently than expected.

The NCI database turns out to possess both a large number of fragment pairs which seem to avoid each other (1110 of z-value<-3), and an even larger number of fragment pairs which seem to group together (2897 of z-value>3).

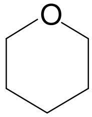
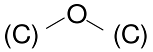
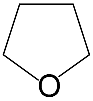
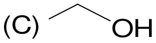
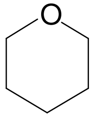
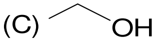
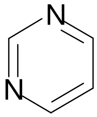
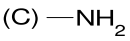
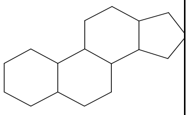
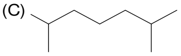
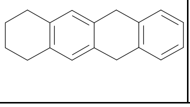
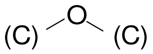
Table 3.4: Some fragment pairs which occur much less frequently together in one molecule than expected, and therefore seem to avoid each other. For example, the phenyl-group and the tetrahydrofuran group (row 3) are both very prevalent fragments, and would be expected to occur together in about 2653 molecules of a 250,000 molecule database. However, in the NCI database they are combined in only 270 molecules, giving an "expectation fraction" of 270/2653 or 0.10. This relation is highly significant, being 67 standard deviations under the expected value.

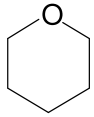
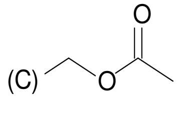
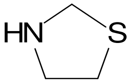
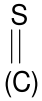
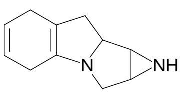
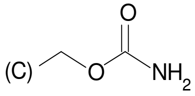
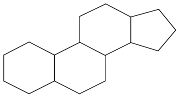
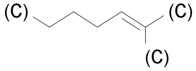
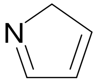
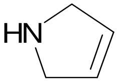
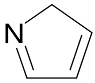
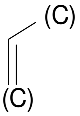
z-value	Fragment 1	Fragment 2	Expected occurrence	Real occurrence	Factor
-31			534	42	0.08
-36			1209	115	0.10
-67			2653	270	0.10
-19			544	139	0.26
-37			1186	323	0.27
-14			611	281	0.46

We first sorted our results on statistical significance, reaching z-values of up to 490 and down to -65. However, after viewing the results we realized that a significant effect does not have to be a very large effect. The most statistically significant negative correlation is that between the benzene and tetrahydrofuran rings, which occur about 10 times less together than expected. The second most significant correlation is between benzene and pyridine, only a 2.4 fold difference. However, *lower* significance is given to higher co-occurrence factors, such as 6.5 for the benzene – tetrahydropyran pair. Therefore we sorted all pairs that had sufficient z-values on their ratio of expected occurrence to actual occurrence. To illustrate our results, we have listed six of the most "avoiding" combinations with z-values<-5 (Table 3.4). These combinations were expected to occur at least five hundred times in the database, but were found much less frequently.

Even more numerous than the avoiding pairs were fragments which seemed to group together. Many of them co-occur so often, that they could be termed "chemical clichés". Often the z-values of the correlations were much bigger for these groups than for the avoiders. In Table 3.5 we show some of the fragment pairs with the strongest enrichment in occurrence for clichés which occur over a 1000 times, over 200 times, and over 50 times in the NCI database.

Table 3.5: Some fragment pairs which co-occur much more often than expected; shown are four fragment pairs which occur in more than one thousand molecules, four pairs which occur in more than two hundred molecules, and four pairs which occur in more than fifty molecules. For example, the tetrahydrofuran group and the –CH_2OH group would be expected to occur only 122 times together, but the pair appears in 2292 molecules (the explanation is of course that these would be ribose-containing molecules). The "gain factor" here is 2292 divided by 122 is 19, and the relation is highly significant, since the found occurrence is over 200 standard deviations from the expected occurrence in a random database (making the chance that their occurrence is independent under 0.0000001%).

z-value	Fragment 1	Fragment 2	Expected Occurrence	Real Occurrence	factor
206			45	1396	31
206			122	2292	19
117			97	1171	12
122			371	2677	7
117			2.3	206	88
185			6.1	463	76

108			7.0	286	41
125			21	591	28
425			0.024	65	2708
125			0.26	62	238
122			0.30	66	221
106			0.39	68	173

Discussion

In this work we performed fragment mining and co-occurrence analysis on a diverse, medium-sized chemical database. In this section we discuss what we can learn from the results, compare our work with that of other investigators, consider uses of the fragment data acquired, and hypothesize about possibilities for extension and improvement of this work.

First, we summarize our conclusions from the results obtained.

Our first observation confirms that of Bemis et al.[11,12] and Xue et al.[13], namely that chemical fragment distributions are extremely lopsided, with a few frequent fragments and many infrequent fragments. Our investigations of several different categories of fragments (rings, substituents, and linkers) however refine this rule; it seems that the classes of the most prevalent fragments, substituents and ring systems, have the most lopsided distribution. In the less used classes of fragments (such as 3- or 4-connecting linkers) the differences in occurrence between the "top-10" and the "bottom-10" fragments are less pronounced (Table 3.3). It may be that in a category of fragments which has not yet been used often, strongly preferred substructures cannot arise or have not arisen yet.

One could speculate on what influences the occurrence of a certain fragment. Three factors come into mind. First, synthetic feasibility/availability; how easy is it to synthesize the fragment, or is the fragment already incorporated into commercially available starting materials? The phenyl group would be a good example of this. Second, versatility; how easy is it to attach other groups to it. Third, popularity: a popular fragment accumulates more and more knowledge which makes it more attractive for use by others, since there is more knowledge available for its manipulation. This could lead to a kind of “winner takes it all” effect, in which relatively small differences in fragment quality may lead to big differences in use. Distinguishing between these possibilities would require further study, for example to find out how many combinations with other groups a certain fragment has per occurrence.

The fragment co-occurrences give other insights into chemical space. From looking at the structures of the fragments that seemed to avoid each other we could think of different reasons why fragments co-occur less frequently than expected. The first reason may be that there are different classes of compounds, such as natural compounds and “synthetic” compounds. In natural compounds, sugar and nucleobase systems may be more prevalent, while in many industrial chemicals the phenyl group plays a dominant role. An example would be the third pair of Table 4, in which the tetrahydrofuran ring (as part of ribose) would occur in the natural compounds, while the phenyl is more likely to occur in “synthetic” compounds. A second reason may have to do with ease of combination: a keto group cannot be attached directly to a phenyl ring, and therefore tends to occur less often with it. It can of course be attached to another ring system in the same molecule, but since effectively one part of the molecule has no positions available for it, the overall chance of the keto group occurring in a phenyl-containing molecule will be lower than average. Finally, some combinations may be found by the statistics to be less frequent than expected since one group is used as a replacement for the other group. Thus, bromine and chlorine relatively rarely occur in the same molecule, possibly because they have similar electronic and chemical properties. Likewise, napthalene and benzene can take similar roles as molecule cores, and “compete” since most molecules have only one or two ring systems.

Let us now turn to the possible reasons for the clichés. Looking at the clichés we found, the first likely reason for their existence is that synthetically the clichés do not represent the smallest building block of a molecule. If moieties such as ribose (instead of unsubstituted tetrahydrofuran) are the real building blocks used to create larger molecules, the co-occurrence of the tetrahydrofuran ring (present in ribose) and -OH groups is certainly not surprising.

The explanation for a number of other clichés is that they represent specific classes of biologically active compounds. As examples, we found dihydrocholesterol analogs (Table 3.5, fifth pair), doxorubicin analogs (Table 3.5, sixth pair), mitomycins (Table 3.5, ninth pair) and folic acid derivatives (not shown) listed as clichés with 70- to 2700-fold occurrence relative to expectation. These clichés do not so much reflect the choice of building blocks, but rather show the active structures nature provided and chemists explored around.

Let us now compare our results with those of others. Fragment mining has been done by several researchers, both as main research subject[11,12,13] and as a preparation for virtual synthesis[14,15,16]. In the methods section we already touched upon the different types of fragmentation, of which the retrosynthetic fragmentation, though not chemically perfect, has been applied and used by those researchers who want to perform virtual synthesis as a prelude to real synthesis. Studies with graph splitting have mainly focused on exploration of the (drug like) molecule space.

Co-occurrence analysis as reported here has to our knowledge not been done yet, and we therefore will focus our comparison on the diverse types of fragment finding as performed by other investigators and ourselves. The differences between our research in fragment frequency and that of others are caused by three factors: the database mined, the breaking points considered, and the ways in which the fragments are represented and distinguished.

Let us first compare the databases mined. Bemis et al.[11,12] used a rather small database, the Comprehensive Medicinal Chemistry (CMC) database[18], which was filtered to get an even smaller database containing only drugs and drug candidates (5120 compounds). On the other hand, Lewell et al.[9] used a big database containing several millions of compounds, many more than we used. However, the relatively small size of the NCI is sufficient for an exploratory study such as ours, and our algorithm is fast enough to make mining of databases of 10 million compounds quite doable on a personal computer. Fragment mining, though not as "new" anymore as in 1996[11], is something that has to be done periodically since the amount of data available is also growing and offering new opportunities. Another aspect is the quality of the data; a drug-like database might give more valuable information for the pharmaceutical industry, but might give a skewed image of chemical synthesis. A more general database, such as the NCI that we mined, seems more appropriate for getting general information about chemistry, but will give fragments which would be unsuitable for drug development.

The investigations also differed in the breaking points considered. Most authors have divided drug molecules into substituents and a kind of framework that contains ring systems and linkers. Bemis et al.[11,12] considered the substituents and the frameworks, which were the ring systems together with parts of the linkers. Lewell et al.[9] concentrated on the rings, Xue et al.[13] on a framework-like part of the molecule called the scaffold. To our knowledge, there has not been a separate investigation of linkers, especially not of the rarer linkers with more than two attachment points. However, linking ring systems together is important for drug development, and a catalogue of linkers of varying length could have similar usefulness as the ring system catalogue compiled by Lewell et al.[9] So the use of the linker breaking points by our method yields additional useful information.

The last point is substructure representation. The aspects relevant here are the representations of the atoms and bonds in the substructure itself, and the encoding of the attachment points. The atoms and bonds can be given a general type (like a wildcard that can represent any atom, or any heteroatom), which results in more "general" fragments. These general fragments will necessarily be fewer in number than the original (normal) chemical fragments. A more important issue, from a chemist's point of view, is that such a general fragment can encode many substructures of possibly vastly differing ease of synthesis. So before one chooses between using more general or more specific fragments, one should consider whether one just needs to know if frameworks with approximately the right size and shape are available, or whether one rather needs a specific substructure with a high chance to be synthesized easily.

Xue et al.[13] treated the substituents as unattached R-groups and did not distinguish between a methyl attached to a carbon ring atom and a methyl attached to a nitrogen ring atom. We and Bemis et al.[12] do distinguish between those options. For the ring systems, Lewell et al.[9] also considered ring systems with different attachment points (for example ortho- and meta-substituted phenyl) as distinct ring systems. Chemically, some positions in rings are easier to modify than others, but it is unclear how important this difference in reactivity is. Would other substitution positions be impossible? It is difficult to estimate the advantage gained by using only known ring substitution patterns against the loss of perfectly viable ring systems which accidentally have not been substituted in that particular pattern yet. For initial exploration of chemical space around a lead molecule, one would prefer substructures which are easy to incorporate in the molecule. For the fine-tuning of structures, however, it would be better to have more candidates available, even if not much is known yet on some of them. We could therefore say that the level of detail of substructure representation can be chosen relatively freely but different levels of details will be preferred for different phases of molecule design and by different chemists. Some chemists might choose a maximum amount of attachment information, while others might allow more “wildcards” in the structure. Ideally, one would therefore want to have a number of fragment databases, each with its own specificity, from which the most appropriate level of specificity can be chosen.

The next point is how we can make use of the fragment libraries and correlations.

The first use of the fragment libraries would be to give chemists more ideas for lead optimization. While investigators such as Lewell et al.[9] mainly considered ring systems that are sterically and electronically similar to a lead ring system to be useful for chemists, we suggest that using the most common as well as the least common fragments could also be effective. The most common fragments could be used as a kind of checklist in the first and most exploratory phase of lead optimization; these fragments are apparently often easy to incorporate into molecules, and thus can lead to fairly diverse exploration at relatively low costs in time and effort.

For example, consider benzodiazepines, which are widely used as tranquilizers. A typical benzodiazepine is shown in Figure 5.

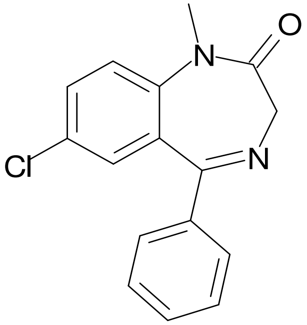


Figure 5: Diazepam, a typical benzodiazepine.

Almost all benzodiazepines use the phenyl ring as group. However, is this biologically necessary or just usage of the well known phenyl cliché? Using the program SciFinder[19] to search the CAS-database, we found over 14,000 phenyl-benzodiazepine compounds. Going down in our ring system list of Table 3.3, we found that the second and third most popular ring systems, pyridine and cyclohexane, have been tried a few hundred times as phenyl substitutes. The numbers four to ten of our list have in general only been used a few times up till a few dozen times, but numbers 6 (tetrahydrofuran), 8 (tetrahydropyran) and 10 (purine) never. So while the compounds in Figures 6a and 6b have been made, 6c and 6d are yet unexplored.

While it may be possible that these particular ring systems are difficult to incorporate, the search strongly suggests that by just going over the top positions of a list of ring systems or substituents one can easily generate a few dozen variations on a lead compound. Since all of these fragments are quite frequent, many of the suggestions are probably relatively easy to synthesize or incorporate.

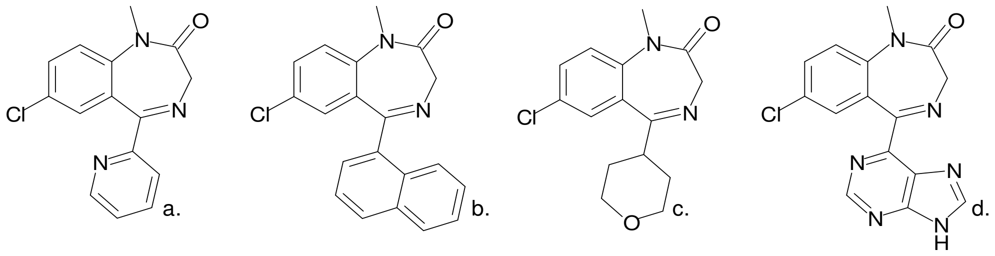


Figure 3.6: Non-phenyl benzodiazepines. Compounds with the first two types of attached ring systems (pyridine (a) and naphthalene (b)) are known. However, neither the tetrahydropyran (c) nor the purine ring (d) have so far been combined with the benzodiazepine scaffold.

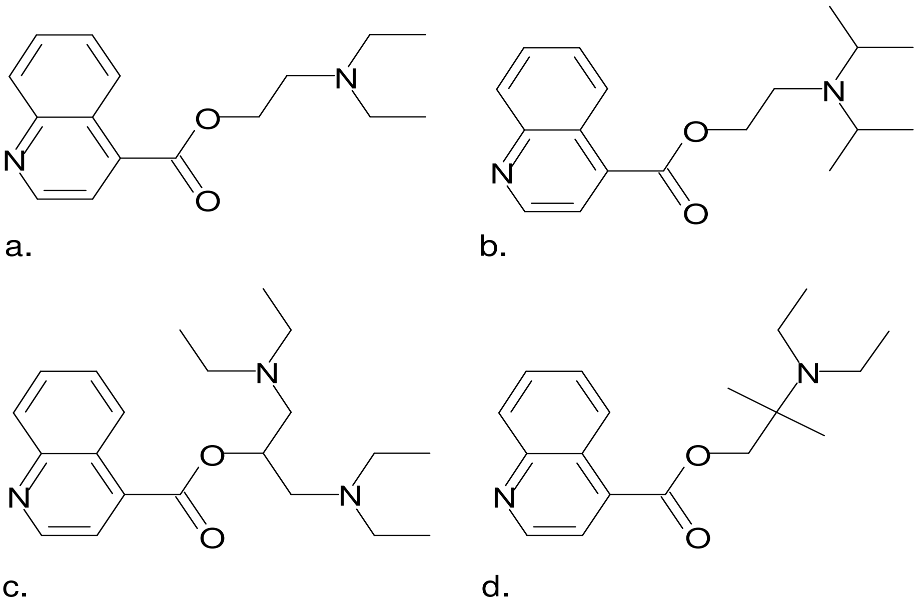


Figure 3.7: Finding cinchocaine derivatives: examples of local-anesthetic like tails which have been used (a, b) and which have not been used yet (c, d) with the quinoline group.

Another example of clichés is the class of local anesthetics, many of which have a phenyl ring (procaine, benzocaine, prilocaine), with as one of the few exceptions cinchocaine, which has a quinoline ring instead of the phenyl. Searching at the typical local anesthetic tail-pattern of $C(=O)OC_xN$ in the substituent database yields many variants of the standard $COOC_2H_4N(C_2H_5)_2$ pattern, all of which have been tried with phenyl (which is not surprising, since phenyl is the "golden standard" among rings), but some of the less frequent substituents have never been paired yet with the quinoline group (Figure 7). Some of these might also be worthy of further investigation.

Selecting fragments for rarity, conversely, can also pay off if the current scaffold is patent-protected; rare substructures can be especially helpful in this case as it is unlikely that many experiments have been done on them. Also, industries could deliberately add the more attractive of the rare fragments to their compound libraries and collections. In this way the libraries will become more diverse and thereby increase the coverage of chemical space and the chance of finding a lead compound in the first place.

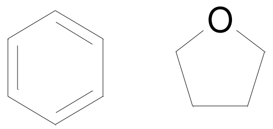
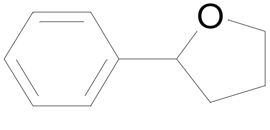
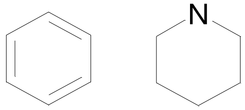
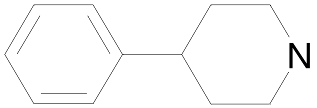
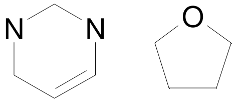
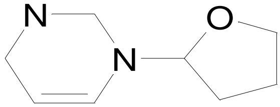
The fragment co-occurrence data can also have some applications. First, co-occurrence analysis can help database analysis by automatically finding clusters of biologically active or well-investigated structures. This can help “summarize” the database for a new user or alert an expert that a certain class of compounds is suddenly becoming popular. Secondly, co-occurrence analysis can add information to structure searching in databases. Often, when a substructure is entered, a long list of structures, each with its specific combination of R-groups, is returned. While currently there is only little attention paid to the number of times a certain R-group appears, a co-occurrence analysis could direct the chemist’s attention to the fact that a certain combination occurs quite often. So there would be something interesting with that combination.

A third application would be for chemists to find relatively unpopulated places in chemical space. For example, there would not be many good reasons why phenyl and tetrahydrofuran could not be combined into one molecule; the relative absence of the combination (Table 3.4, pair 3) suggests that a compound combining these groups might be worthwhile to synthesize.

In chapter 4 of this thesis we describe a procedure for generating new drug-like molecules. The molecules we discovered were small and relatively simple, but often not yet known in the literature as a compound or as a substructure of other compounds. The molecule we show in that article has a phenyl ring attached to a piperidine ring and a CH_2OH group. While the phenyl ring itself is very much a chemical cliché, it usually avoids both the piperidine ring (avoiding pair rank number 75) and the CH_2OH group (avoiding pair rank number 78) according to our data. This suggests that we might also be able to use this process the other way around: by combining fragments that are common but seem to avoid each other a person or computer program could find simple structures that have not been synthesized before.

Would we indeed be able to discover new compounds by considering the negative co-occurrences? To check that, we looked at three different pairs of rings, which were a strongly avoiding pair (phenyl and tetrahydrofuran, rank 7 of the list), the less strongly avoiding pair of phenyl and the somewhat less frequent piperazine ring (rank 75), and the cliché of the C_4N_2 ring depicted in the table together with the tetrahydrofuran (Table 3.6). The most strongly avoiding pair appears in about 10,000 molecules in the CAS database (so at a ratio of about one in 2,000 compounds in the database). Assuming independence, one would expect that the phenyl-piperazine pair occurs less often since the piperazine is (at least in the NCI database) rarer than tetrahydrofuran. However, since the groups avoid each other less, this combination occurs approximately 60,000 times, so six times as many. Finally, the C_4N_2-tetrahydrofuran cliché, despite both rings being relatively rare on their own, occurs in over 111,000 compounds, since this combination is the core of uracil and thymidine molecules.

Table 3.6: The avoiding groups can indicate "holes" in chemical space, where relatively little research has been done. In the first column of this table pairs of fragments are shown. The second column contains the z-value of correlation. Positive z-values mean that the fragments group together, negative z-values mean that they avoid each other. The third column contains a specific substructure which contains both fragments. We have always taken a substructure which contains the fragments directly connected to each other, and when there were multiple coupling possibilities, the substructure which was most frequently found by SciFinder[19]. In the final column the estimated number of known molecules containing the substructure is shown. So, the phenyl and tetrahydrofuran seem to avoid each other rather strongly (z=-57), and the substructure with the tetrahydrofuran attached at its 2-position to phenyl indeed occurs in only about 11,000 molecules.

Fragment pair	z-value	Substructure investigated	Occurrence in SciFinder (estimated by the program)
	-57		10,594
	-23		57,723
	+87		111,292

It may also be possible to apply fragment analysis and fragment co-occurrence analysis to other problems such as measuring chemical diversity. While there are several different measures of chemical similarity, the methods most similar to ours are those which make fingerprints of molecules based on the presence and absence of fragments (for example the MDL/MACCS keys and the BCI fingerprints[20]). The MDL and BCI fingerprints usually work with only very small fragments, such as "an atom in a multiple, nonaromatic bond located two bonds away from an atom with at least two heteroatom neighbors". In contrast, our method uses much larger and more specific fragments and generates only a few fragments per molecule, which will lead to a much larger emphasis on changes in framework (so naphthalene will be significantly different from benzene, though they both have aromatic carbon atoms). Diversity could be estimated by for example dividing the number of unique fragments by the size of the database. For the NCI this would be (not counting the unconnected fragments) 46010 / 250251 = 0.18, or as Ertl's research[8] suggested, by dividing the log values, which would yield 0.91 in our case. Calculating the entropy of the distribution might also be worthwhile (a database with 999 times fragment 1 and 1 time fragment 2 could be considered less diverse than when the division is 500 – 500).

The result of using larger fragments instead of small fragments would be that chemical diversity is enhanced; a problem might be ease of synthesis or cost of acquiring such a library. But diversity does perhaps not have to be high, since the research of Bemis et al.[11] has shown that half of all drug molecules have one out of only 32 different frameworks. One apparently does not need very high diversity to get active drugs on very different targets. On the other hand, if the NCI is representative of chemical space, most alternative frameworks have been rarely synthesized and screened, and would therefore have a much smaller chance of leading to a drug. Until it is shown that alternative frameworks are really much worse for drug design, the chemical diversity stimulated by our "big fragment" method could be a good start.

The final benefit of fragment mining and co-occurrence analysis might be psychological: thinking of substructures which have or have not been used together might make chemists more conscious of their choices, giving them more knowledge to decide whether to use clichés for use of synthesis, or avoid them to explore novel structural classes.

As final part of this discussion we would like to reflect on what directions our fragment mining investigations could take.

First of all, there are some possibilities for algorithm improvement. While doing a stochastic simulation of fragment pair expectance is relatively easy and computationally cheap (about 15 minutes on a 3 GHz PC), the averages and standard deviations can be calculated exactly with a so-called chi-squared distribution with one degree of freedom. This will be especially valuable for larger databases. Stochastics, however, may continue to play a role since they make it easy to add certain restraints (such as a maximum number of fragments per molecule, or multiple identical fragments per molecule) that are more difficult to enforce by mathematics. It would also be interesting to mine larger databases, such as ZINC[21] or PubChem[22]. In any case this is likely to add fragments to our databases, and perhaps discover new clichés or avoiding groups since more data can lead to more strongly pronounced z-values.

A second direction for further investigation would be to make the relationships of the co-occurrences more detailed. For example, currently we only detect whether two fragments are present in the same molecule. However, our method can be extended by taking into account whether the fragments are directly attached to each other, or whether the attachment point is consistent over many molecules. Additionally, detection of co-occurrences of three fragments or more could be a worthwhile extension.

A third development would be experimenting with different representations of the substructures; some chemists would not care whether a methyl group is attached to a ring-N or ring-C, others would like to be sure that a certain ring position is suitable for substitution. Atoms and bond types could be converted to wildcards to create a smaller library of general fragments, or conversely the connection points could be extended and classified for more certainty of ease of synthesis. In the end, we would like to have a system that provides the right kind of data for each application.

Conclusions

In this investigation, we mined the NCI database of 250,251 compounds. This resulted in over 60,000 fragments of different types: ring systems, substituents, and diverse kinds of linkers. Fragment occurrence is very skewed, with 70% of fragments occurring only once, and a few fragments (such as phenyl and methyl) being present in many molecules.

The fragment lists and co-occurrences can be used in different ways. In our examples we have shown how the fragment lists can be used to find new ring substituents for benzodiazepines and local anesthetics. Also, we found that co-occurrence analysis can automatically detect groups of biologically active compounds, such as the doxirubicin and mitomycin analogs in the NCI. Finally, co-occurrence analysis of the avoiding fragments can show “holes” in chemical space where there is room for small, novel compounds which may be biologically active.

Future directions of this work could be investigating either larger or more focused databases, taking information of how fragments are attached to each other into account and experimenting with different levels of substructural detail. Fragment analysis can show us many chemical patterns, but the conversion of pattern knowledge into chemical understanding has only just begun.

References

[1] http://www.mdl.com/products/knowledge/crossfire_beilstein/

[2] http://www.cas.org/chemplus/chemplus1.html

[3] Lipinksi, C.; Lombardo, F.; Dominy, W.; Feeney, P. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews* **1997**, *23*, 3-25.

[4] Oprea, T. Property distribution of drug-related chemical databases. *Journal of Computer-Aided Molecular Design* **2000,** *14,* 251-264.

[5] Xu, J.; Stevenson, J.; Drug-like Index: A New Approach To Measure Drug-like Compounds and Their Diversity. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 1177-1187.

[6] Klopman, G. Multi-CASE: a hierarchical computer automated structure evaluation program. *QSAR* **1992**, *11*, 172-184.

[7] Kazius, J.; McGuire, R.; Bursi, R. Derivation and validation of toxicophores for mutagenicity prediction. *J. Med. Chem.* **2005**, *48*, 312-320.

[8] Ertl, P. Cheminformatics Analysis of Organic Substituents: Identification of the Most Common Substituents, Calculation of Substituent Properties, and Automatic Identification of Drug-like Bioisosteric Groups. *J. Chem. Inf. Comput. Sci.* **2003,** *43,* 374-380.

[9] Lewell, X.; Jones, A.; Bruce, C.; Harper, G.; Jones, M.; Mclay I.; Bradshaw, J. Drug Rings Database with Web Interface. A Tool for Identifying Alternative Chemical Rings in Lead Discovery Programs. *J. Med. Chem.* **2003**, *46*, 3257-3274.

[10] http://cactus.nci.nih.gov/ncidb2/download.html

[11] Bemis, G.; Murcko, M. The Properties of Known Drugs. 1. Molecular Frameworks. *J. Med. Chem.* **1996**, *39*, 2887-2893.

[12] Bemis, G.; Murcko, M. Properties of Known Drugs. 2. Side Chains. *J. Med. Chem.* **1999**, *42*, 5095-5099.

[13] Xue, L.; and Bajorath, J. Distribution of Molecular Scaffolds and R-Groups Isolated from Large Compound Databases. *J. Mol. Model* **1999,** *5,* 97-102.

[14] Vinkers, M.; De Jonge, M.; Daeyaert, F.; Heeres, J.; Koymans, L.; Van Lenthe J.; Lewi, P.; Timmerman, H.; Van Aken, K.; Janssen P. SYNOPSIS: SYNthesize and OPtimize System in Silico. *J. Med. Chem.* **2003,** *46*, 2765-2773**.**

[15] Schneider, G.; Lee, M.; Stahl, M.; Schneider, P. De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks *Journal of Computer-Aided Molecular Design,* **2000,** *14*, 487-494.

[16] Lewell, X.; Judd, D.; Watson, S.; Hann, M. RECAP-Retrosynthetic Combinatorial Analysis Procedure: A Powerful New Technique for Identifying Privileged Molecular Fragments with Useful Applications in Combinatorial Chemistry. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 511-522.

[17] Weininger, D.; Weininger, A.; and Weininger J. SMILES. 2. Algorithm for Generation of Unique SMILES Notation. *J. Chem. Inf. Comput. Sci.*, **1989**, *29*, 97-101.

[18] CMC is currently (2005) marketed by Elsevier MDL, see also the website http://www.mdl.com/products/knowledge/medicinal_chem/index.jsp

[19] We used SciFinder Scholar marketed by CAS: http://www.cas.org/SCIFINDER/SCHOLAR/

[20] Wild, D.; Blankley, C. Comparison of 2D Fingerprint Types and Hierarchy Level Selection Methods for Structural Grouping Using Ward's Clustering. *J. Chem. Inf. Comput. Sci.*, **2000**, *40*, 155-162.

[21] Irwin, J.; Shoichet, B. ZINC – A Free Database of Commercially Available Compounds for Virtual Screening. *J. Chem. Inf. Model.* **2005**, *45*, 177-182.

[22] http://pubchem.ncbi.nlm.nih.gov/

4 The Molecule Evoluator. An interactive evolutionary algorithm for the design of drug-like molecules

Eric-Wubbo Lameijer[1], Joost N. Kok[2], Thomas Bäck[2] and Ad P. IJzerman[1]

[1]Leiden/Amsterdam Center for Drug Research, Division of Medicinal Chemistry, PO Box 9502, 2300RA Leiden, The Netherlands

[2]Leiden Institute of Advanced Computer Science, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

This chapter was first published in the Journal of Chemical Information and Modeling, reference: Lameijer, E.W.; Kok, J.N.; Bäck, T.; IJzerman, A.P. The Molecule Evoluator. An Interactive Evolutionary Algorithm for the Design of Drug-Like Molecules. *J. Chem. Inf. Model.*, **2006**, *46*(2), 545-552.

Abstract

We developed a software tool to design drug-like molecules, the “Molecule Evoluator”, which we introduce and describe here. An atom-based evolutionary approach was used allowing both several types of mutation and crossover to occur. The novelty we claim is the unprecedented *interactive* evolution, in which the user acts as a fitness function. This brings a human being’s creativity, implicit knowledge and imagination into the design process, next to the more standard chemical rules. Proof-of-concept was demonstrated in a number of ways, both computational and in-the-lab. Thus, we synthesized a number of compounds designed with the aid of the Molecule Evoluator. One of these is described here, a new chemical entity with activity on α-adrenergic receptors.

Introduction

It has been estimated that the combination of the four elements C, N, O and S only may lead to the design of 10^{60} molecules with maximally 30 non-hydrogen atoms[1]. Currently, only over 26 million (in between 10^7 and 10^8) organic and inorganic compounds have been synthesized since the foundation of organic chemistry in the 19th century[2]. Obviously, designing new molecules (*de novo* design) is crucial to cover more of the "chemical space". Computational methods have been employed to this end, among which so-called evolutionary algorithms (reviewed in ref. 3). These algorithms are based on principles from biology, such as natural selection and Darwinian evolution through mutation and crossover.

There are at least two problems with the application of evolutionary algorithms in drug design. The first is that linear 'genes' as used in biological evolution are ill-suited to represent molecules. Molecules are graphs that have to obey certain chemical rules. Converting a molecule into a bit string or fixed-size vector of numbers as used in conventional evolutionary algorithms will therefore often result in mutation and crossover operations that produce invalid molecules.

The second problem relates to natural selection and a useful fitness function. While currently the most common methods use a similarity index to a reference compound[4-7] or calculations of the binding strength to a target[8,9], only a few examples have been published of successful applications of evolutionary algorithms to find new, biologically active structures. Only Schneider et al[7] claimed success since their algorithm found a ligand with a similar kind of activity as the lead compound, be it approximately 1000 times less potent.

In this study we propose a new approach for the use of evolutionary algorithms in *de novo* drug design, called the "Molecule Evoluator", to address these problems. We introduce an atom-based method with a set of mutation operators that contains all one-atom and one-bond mutations. This will enable a fuller search of the chemical space and finer optimization of the molecular structure than is possible with most other published methods. Secondly, we use another approach to natural selection that is new in evolutionary algorithms in drug design: we make the medicinal chemist the "fitness function" of the proposed structures. This would largely eliminate structures that are difficult to make in the laboratory and enable the program to optimize the molecular structure by using the chemist's knowledge about structure-activity relationships.

We will first introduce a new representation of molecules, the so-called TreeSMILES notation, and the evolutionary operators we used. Subsequently we will discuss how the user’s choices affect the fitness of the molecules. We will then shortly describe the graphical user interface of the Molecule Evoluator, together with the results of some of our experiments in interactive drug design.

Methods

Molecule representation

A molecule can be considered to be a connected graph consisting of one or more atoms (nodes) connected by bonds (undirected edges). One of the main constraining rules of chemistry is the valence of each type of atom. Moreover, atoms such as sulfur can have several valence states, of which bivalent sulfur and hexavalent sulfur are the most common. In our genotype, the starting point in each evolutionary algorithm, we handle this by creating a separate symbol for each valence state, so that divalent sulfur atoms are encoded by “S” and hexavalent sulfur atoms by “Sh”. Thus, whether a graph represents a valid molecule does not merely depend on the structure of the graph, but also on the identity of each particular node.

Molecules are graphs that can be of different sizes; they may also contain branches and cycles. Since they too are subject to the laws of chemistry they additionally have to obey certain rules that usually prohibit changing one node/edge without changing its neighbors. To develop a computer representation of a molecule that can store these properties and that is convenient to mutate and cross over, we looked at some common molecular data formats.

There are various ways to represent molecules on the computer. Nowadays, the two most common representations are the MOL-file format[10], which is a list of atoms and bonds in the molecule, and the SMILES-notation[11], a line notation/string that is human-readable and that can easily be transformed into a 2D-structure. In the SMILES notation the nonlinear parts of a molecule are encoded with brackets to indicate branches and numbers to label rings, as is illustrated in Figure 4.1A. By default the hydrogen atoms are not incorporated into the SMILES notation, since their presence can be deduced from the rules of chemical valence.

Neither of these representations is ideal, however, when molecules are mutated and crossed over. Since chemical valence rules explicitly state how many bonds any

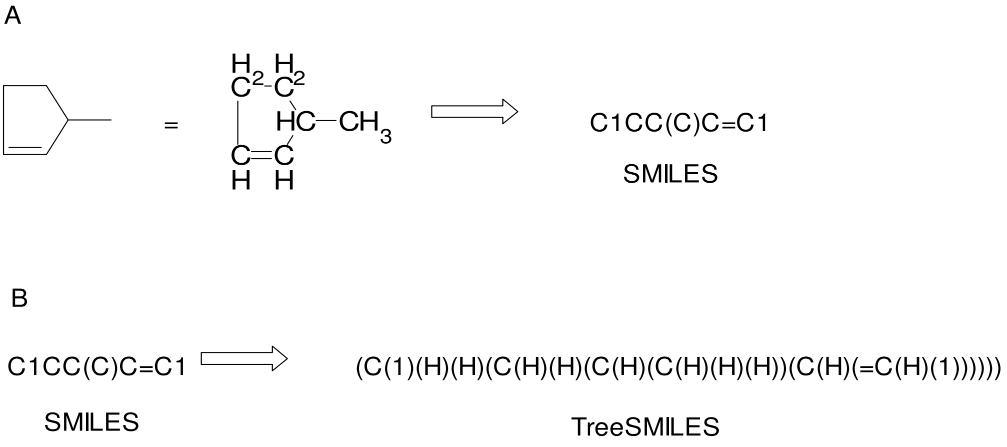


Figure 4.1:
A) Example of a SMILES representation of a molecule. One bond of each ring is chosen at random (in this case, the bond from the top left to the bottom left atom) and is designated by a unique number. Both atoms participating in a ring bond get the bond number(s) immediately after them in the line notation. Branches are indicated by brackets.
B) SMILES versus TreeSMILES. By making the hydrogen atoms explicit going from SMILES to TreeSMILES the notation becomes less compact and less human-readable. However, the positions available for substitution are now readily interpreted by the computer.

atom has, both the most common 2D-MOL-file format and the SMILES notation take the hydrogen atoms and the bonds to which they are attached for granted. We should therefore calculate for each atom whether it has the right number of hydrogen atoms to be mutated in some way. To avoid these recalculations, we decided to explicitly add the hydrogen atoms to the representation. Also, we decided to use a SMILES-like structure as our main molecule representation, particularly for reasons of computational efficiency. Making the hydrogen atoms explicit we obtain a bracket-rich, expanded SMILES representation (Figure 1B) to which we can apply mutation by relatively simple algorithms. We coin this chemical notation "TreeSMILES".

Crossover and mutation

Crossover is implemented like crossover in standard genetic programming[12]: subtrees of two different molecules are selected and swapped. The only complication in the present study is that the trees represent graphs and can contain cycles, while subgraphs are not allowed to contain incomplete cycles. So when a subtree at a random "root" atom is selected, the subtree is first checked for unmatched ring bonds, and if these are present the current root is discarded and another subtree is selected for crossover.

Mutation, however, is the most important variance operator in the Molecule Evoluator. Theoretically, changing a graph into any other graph can be performed by a limited set of operations: adding nodes, adding edges, deleting nodes, deleting edges. While this would be sufficient from a graph-theoretical point of view, these operations are more complicated in a chemical system since the valence rules must be obeyed, and the graph must remain connected: deleting the indicated carbon atom in Figure 4.2 would also require deleting its three attached hydrogen atoms and attachment of a hydrogen that replaces the carbon. In the Molecule Evoluator, deleting an atom involves removal of the hydrogen atoms attached to it and renaming the atom itself into a hydrogen atom.

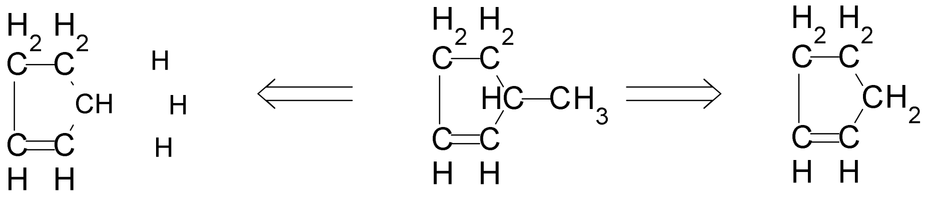


Figure 4.2. Why simple mutations are often wrong mutations. When removing the rightmost carbon from the middle molecule, only removing the carbon (left picture) results in a wrong molecule: due to the loose hydrogen atoms the structure is unconnected (and therefore not a valid molecule), also the carbon atom to which it was connected has three bonds instead of four, violating chemical valence rules and also making the left molecule invalid. The molecule on the right shows the valid version of this mutation: the to-be-deleted atom is changed into a hydrogen and its own hydrogen atoms are removed.

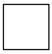
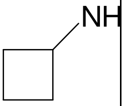
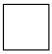
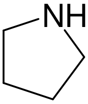
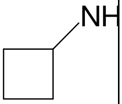
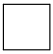
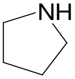
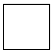
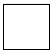
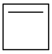
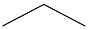
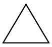
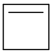
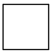
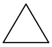
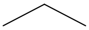
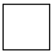
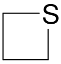
The implemented mutations, graphically shown in Table 4.1, are as follows:

1) *Add atom/group*: this replaces a hydrogen atom in the molecule with a non-hydrogen atom or a larger chemical group such as a phenyl group. In the case of atom addition, the remaining bonds of the added atom are filled with hydrogen atoms.
2) *Insert atom*: also adds an atom, but does this by inserting the new atom (which should have a valence of two or higher) into a bond. The remaining bonds of the new atom are completed by adding hydrogen atoms to it.
3) *Delete atom*: this removes an atom that is attached to only one non-hydrogen atom (with a single bond) by first deleting the hydrogen atoms attached to it, and renaming the atom to a hydrogen atom.
4) *Uninsert atom*: This removes an atom that has exactly two non-hydrogen neighbors. It removes the atom and its hydrogen atoms and subsequently creates a bond between its two neighboring non-hydrogen atoms.
5) *Increase bond order*: if two atoms that are bonded to each other both have at least one hydrogen atom, those hydrogen atoms are removed and an extra bond is created between the atoms (the bond order is increased from single to double or from double to triple).
6) *Create ring*: similar to *increase bond order*, but works between two atoms that are not bonded to each other. These atoms are connected using a single bond (the two hydrogen atoms are changed into ring indices).
7) *Decrease bond order*: if there is a double or triple bond between two atoms, its bond order is decreased by one and a hydrogen atom is attached to each of the two atoms.
8) *Break ring*: this mutation chooses a single bond in a ring, breaks that bond and adds hydrogen atoms to correct the valences. The algorithm should be able to break any bond in the ring, which is not easy to do with a tree structure[13]. To solve this problem, we converted the TreeSMILES into an adjacency list. In the adjacency list, any ring bond can be broken easily and afterwards a new TreeSMILES representation is built. This is the only mutation where we found it necessary to temporarily convert the TreeSMILES representation of the molecule into an adjacency list format for easier modification. Since other operators such as crossover are more easily implemented for a TreeSMILES string, we decided not to use the adjacency list for the other mutations. Changing representations is probably a convenient way to accommodate different mutations. To our knowledge, this technique has not been used before in evolutionary algorithms in *de novo* design, and is probably also rare in evolutionary algorithms in general. However,

we think that it might also have applications outside the Molecule Evoluator.

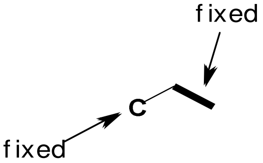
9) *Mutate atom*: a non-hydrogen atom is changed into another non-hydrogen atom which has a valence of at least the number of bonds of the original atom with other non-hydrogen atoms.

Table 4.1: Schematic overview of the different mutations in the Molecule Evoluator. Most leave the TreeSMILES string fairly intact and can be performed by string editing. The only exception is the "break ring" mutation, which can substantially rearrange the TreeSMILES.

Mutation name	**Initial structure**	**Final structure**	**Initial TreeSMILES**	**Final TreeSMILES**
Add atom			...(C(H)(*H*)(C…	...(C(H)(**N(H)(H)**)(C…
Insert atom			...(C(H)(H)(C...)...	...(C(H)(H)(**N(H)**(C...))...
Delete atom			..(C(H)(*N(H)(H)*)(C...	...(C(H)(**H**)(C...
Uninsert atom			...(C(H)(H)*(N(H)(*C...*))*...	...(C(H)(H)(C...)...
Increase bond order			...(C*(H)*(H)(C*(H)*(H)(...	...(C(H)(**=**C(H)(…
Create ring			(C(*H*)(H)(H)(C(H)(H)(C(*H*)(H)(H)))	(C(**1**)(H)(H)(C(H)(H)(C(**1**)(H)(H)))
Decrease bond order			...(C(H)(=C(H)(…	...(C(**H**)(H)(C(**H**)(H)(...
Break ring			(C(1)(H)(H)(C(H)(H)(C(1)(H)(H)))	(C(C(H)(H)(H))(H)(H)(C(H)(H)(H)))
Mutate atom			...(C(H)(H)(*C(H)(H)*(C...	…(C(H)(H)(**S**(C...

We also allow the user to select atoms and bonds that will remain unaltered by crossover and mutation (‘fix’ option). Therefore we have further modified the data structure of the TreeSMILES by using an array of character pairs instead of a normal string/array of characters, in which the first character of the pair is the normal TreeSMILES character and the second character is a flag, which indicates whether the atom or bond designated by the first character can be modified (Figure 4.3).

Figure 4.3. The TreeSMILES notation of n-propane (C_3H_8) that incorporates fixable characters ("charf"s). In this example, one atom and one bond are fixed.



(	C	(	H	)	(	H	)	(	H	)	(	C	(	H	)	(	H	)	(	C	(	H	)
ø	F	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	ø	F	ø	ø	ø	ø

(	H	)	(	H	)	)	)	)
ø	ø	ø	ø	ø	ø	ø	ø	ø

Fitness

The final component of the evolutionary algorithm is the fitness function. So far, investigations on *de novo* design with evolutionary algorithms have used several types of a fitness function, including:

- similarity to a target molecule[4,5]
- QSAR-functions[13]
- ligand-protein docking[8,9]
- experiment[14]

For drug design, each of these methods has its advantages and disadvantages. Similarity approaches often result in molecules very similar to the target molecule, in many cases even the target molecule itself, which is not useful for designing truly novel molecules. QSAR functions should be able to optimize activity, yet have important disadvantages. First they require quite a lot of reprogramming for each new class of molecules that is investigated. Secondly, they are generally difficult to use for finding very active compounds since they grow less and less reliable as the molecular structures deviate more from the average structure (and thereby activity) of the training dataset[15]. Unfortunately, this is exactly what would happen during optimization. The last theoretical method ("docking") is still too inaccurate for optimization, and may yield molecules with an activity that is orders of magnitude lower than the calculated value[9]. Experiments as fitness function, finally, are generally slow and expensive, and so far have only been performed for a class of molecules that was particularly easy to synthesize[14]. Thus experimental fitness has yet to prove to be practical in a more realistic drug design scenario.

As an entirely different approach, we decided to use the *user* as a fitness function. This concept has been recently employed in other application areas and is often called interactive evolutionary computing[16-18]. While a user cannot know the binding strength of a given molecule, this defect may not be much worse than the inaccuracy of scoring functions. A definite advantage in letting the user choose would be that intensive feedback from a medicinal chemist would make the compounds easier to synthesize, and steer the evolution away from areas which have already been explored. Furthermore, the algorithm could still be easily coupled to experimental results or advanced computed fitness functions if so desired. In Algorithm 4.1 the general workflow within the Molecule Evoluator is detailed, from the creation of the initial population to the interactive evolution of new molecules.

Algorithm 4.1: General workflow of the Molecule Evoluator

```
Create initial generation of 1..generation_size molecules
      // ( generation_size set by user, by default 12)
   For each molecule, get molecule structures from one of
   these sources:
   a. MOL or SD-files
   b. Sketches by user in editor of Molecule Evoluator
   c. Random generation of molecules (see algorithm 4.2)
```

```
Evolve:
      Repeat while the user wants to continue:
          -The user may set or change ratios between
          modification, combination and de novo molecule
          generation.
          -The user selects *m* most interesting molecules of
          the current generation.
          Copy the *m* selected molecules to the next
          generation.
          -For each of the ( *generation_size* - *m* ) molecule
          slots left in the next generation, do the
          following:
                    -generate a new molecule with one of the
                    three methods in ratios set by the user:
                    -*modification*: create a new molecule by
                    mutating the old molecule
                    -*combination*: create a new molecule by
                    crossing over two selected molecules (can
                    be the same molecule - this is allowed in
                    tree-crossover)
                    -*de novo generation*: create a new molecule
                    by de novo generation, see algorithm 2.
                    -add the new molecule to the next
                    generation.
          -The user may change molecules by fixing (as
          explained in the text) or editing, or replace
          molecules by loading or drawing new molecules.
          -Make the next generation the current generation.
```

As a further example, Algorithm 4.2 performs the random generation of molecules as mentioned in Algorithm 4.1.

Algorithm 4.2: Random generation of a molecule

```
set the current molecule to a methane molecule

pick a random number x from 1 to MaxNumberOfAtoms
repeat x times:
       replace a hydrogen atom of the current molecule
       by a random atom or fragment from the
       atom/fragment database

pick a random number y from 0 to MaxNumberOfRings
repeat y times or until no more rings can be made:
make a ring in the current molecule

pick a random number z from 0 to
MaxNumberOfDoubleBonds
repeat z times or until no more bonds can be oxidized:
       oxidize a random bond in the current molecule,
       so increment the bond order
```

Since a user cannot evaluate as many structures as a computer program and preliminary experiments have shown that users only want to see "good" structures, we added several descriptor calculations to the Molecule Evoluator. The selected descriptors are (physico-)chemical parameters of a compound, such as the number of hydrogen bond donors/acceptors, molecular weight, logP (lipophilicity), Polar Surface Area, the number of rotatable bonds, and the number of aromatic systems and substituents. Physico-chemical parameters of a compound, either calculated or experimentally determined, can also constitute a useful fitness function, for instance in a certain weighed combination. It is increasingly realized that they determine e.g., aspects of absorption and blood-brain barrier passage[19], although they usually are of lesser importance for the interaction with the target protein. Upper and lower bounds for all these descriptors can be set by the user as a filter to create more 'realistic' molecules. Additionally, we implemented some 'chemical' filters. The aim was to eliminate molecules with undesirable (sub)structures such as strained paracyclophanes. Similarly, molecules not allowed because of Bredt's rule[20] are also automatically discarded. Both the physical and the chemical filters can be switched on and off.

Results

The graphical user interface of the Molecule Evoluator

When the program is started, a ‘seed’ molecule (indole in this case) can be either loaded or drawn and used for a first generation of derivatives (Figure 4.4). Alternatively, the Molecule Evoluator itself can initialize the population with random molecules. The latter is done by taking the computer’s clock time and using it as a seed in the *rand* function in the C programming language.

As an example one of the suggested molecules (3-methylindole) was selected for yet another round of evolution. Again by pressing the "Go" button a window appears that contains the selected molecule (this so-called elitism is on by default) together with the newly generated analogs (Figure 4.5). The user can again select the most attractive molecules, press “Go”, and this process is repeated until the user has gathered enough ideas.

Comments from medicinal chemists on previous test versions of the program have led us to include three extra features that allow the user more control over the evolution, i) editing the molecules directly, ii) fixing parts of the molecule, and iii) using filters to prevent that certain unrealistic molecules are shown to the user. We next discuss these features in more detail.

Editing molecules is useful when the user wants to start the evolution with a molecule that has not been stored yet in the computer and must be drawn. Additionally, if during the evolution a given molecule suggests a more interesting structure, editing the molecule into the desired structure can be done ‘on-the-fly’. This will allow the user to evolve molecules immediately from the desired structure, rather than wait until it will be generated by the program. Editing the molecules is performed in the "Molecule Edit" window (not shown), which pops up when the user clicks on a molecule in the main window. Editing is quite similar to that in normal chemical drawing programs. After the popup window has been closed, the drawn structure is converted into TreeSMILES-format.

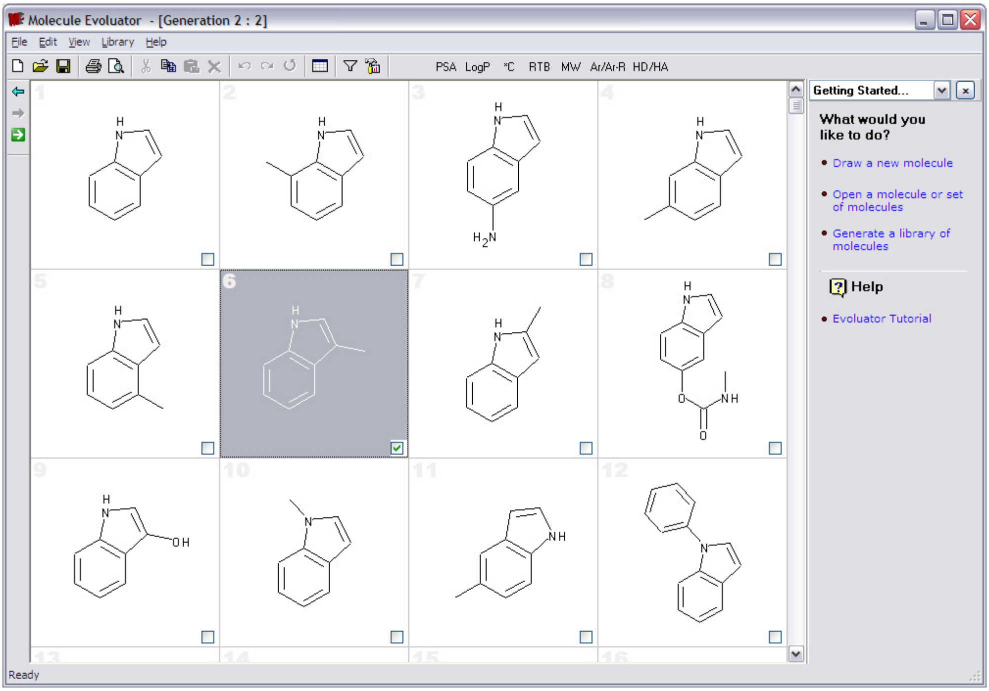
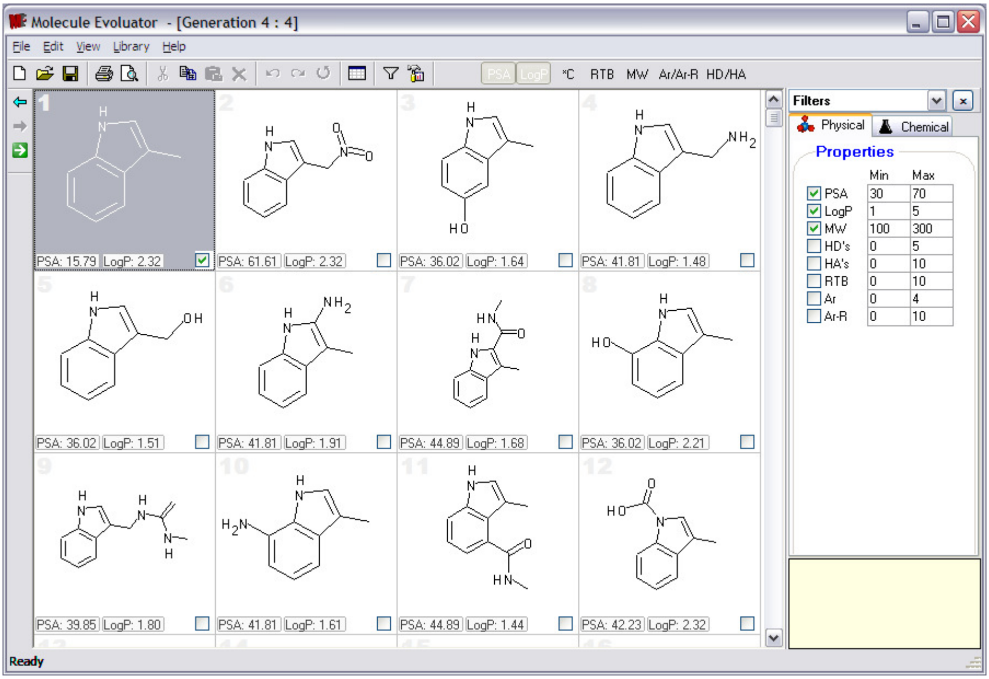

Molecule Evoluator - [Generation 2 : 2]
File Edit View Library Help
PSA LogP °C RTB MW Ar/Ar-R HD/HA
1
2
3
4
5
6
7
8
9
10
11
12
H
N
H_2N
O
NH
OH
Getting Started...
What would you like to do?
Draw a new molecule
Open a molecule or set of molecules
Generate a library of molecules
Help
Evoluator Tutorial
Ready

Figure 4.4: A molecule, in this case indole (#1), is drawn or loaded as a 'seed' for a population of 12 derivatives generated by pushing the "Go" button (green arrow in left-hand toolbar). In this simple example with atom additions only the methylated analog #6 was selected for further evolution (see Figure 5).

Fixing part(s) of the molecule can be useful in cases where structure-activity relationships demand that a particular, necessary part of the molecule is present in all its descendants. The Molecule Evoluator allows this conservation with the "fix atoms or bonds" option in the "Molecule Edit" window, which enables the user to generate new molecules with the conserved part remaining intact, and only variation on the "free" atoms.


Molecule Evoluator - [Generation 4 : 4]
File Edit View Library Help
PSA LogP °C RTB MW Ar/Ar-R HD/HA
1 PSA: 15.79 LogP: 2.32 ☑
2 PSA: 61.61 LogP: 2.32 □
3 PSA: 36.02 LogP: 1.64 □
4 PSA: 41.81 LogP: 1.48 □
5 PSA: 36.02 LogP: 1.51 □
6 PSA: 41.81 LogP: 1.91 □
7 PSA: 44.89 LogP: 1.68 □
8 PSA: 36.02 LogP: 2.21 □
9 PSA: 39.85 LogP: 1.80 □
10 PSA: 41.81 LogP: 1.61 □
11 PSA: 44.89 LogP: 1.44 □
12 PSA: 42.23 LogP: 2.32 □
Filters
Physical
Chemical
Ready

Properties

	Min	Max
☑ PSA	30	70
☑ LogP	1	5
☑ MW	100	300
□ HD's	0	5
□ HA's	0	10
□ RTB	0	10
□ Ar	0	4
□ Ar-R	0	10

Figure 4.5: Pressing "Go" (green arrow in the left-hand toolbar) generates mutants of methylindole (see Figure 4) according to preset filters (PSA, logP and MW in right-hand panel). PSA and logP values are displayed in the individual boxes, according to buttons pushed in the upper toolbar.

The filters are the third extra feature. In the "Physical Filters" panel (see also Figure 4.5) the ranges are set within which the physicochemical properties of a molecule must lie for the molecule to be incorporated into the population (for example: molecular weight between 100 and 300). Molecules outside these boundaries imposed by the user will be automatically eliminated by the Molecule Evoluator and will therefore not be shown to the user. Additionally, some chemical structures which are usually undesirable, such as hemiketals, can be forbidden in the "Chemical Filters" panel. The Molecule Evoluator creates offspring molecules using mutation and crossover until feasible molecules – fulfilling all filter conditions – have been found.

In addition to these three main control features, there is a "Parameters" panel (not shown), in which the user can influence the evolutionary process itself. The user is allowed to steer evolution by, amongst other things, enabling/disabling certain kinds of mutations. For example, the “decrease bond order” mutation tends to partially reduce phenyl rings, which is chemically undesirable. As an alternative to fixing the phenyl bonds explicitly, disabling this mutation will protect the bonds from being reduced. This will however also prevent useful mutations, such as those which reduce a ketone (C=O) to an alcohol (CHOH). Another option would be a special version of the "decrease bond order" mutation that does not reduce aromatic rings; this is something we are currently considering. Other mutation types (eight in total) can also be toggled ‘on’ or ‘off’. By default, each of these eight categories of mutations is applied with the same frequency (0.125). When mutation types are disabled, the remaining active mutation types are still applied with identical frequency, so if for example only three mutations are allowed, the probability of each of them is 0.33. Another option is to change the ratio of mutation/crossover. Since we learned that most medicinal chemists prefer mutation over crossover, the default settings are 90% mutation and 10% crossover (and so are applied with probabilities of 0.9 and 0.1, respectively). A third option is to allow the Molecule Evoluator to occasionally add random molecules to the population. The relative amount of random molecules is approximately 16% (so 1-2 new random molecules in a new population). This option is off by default. Lastly, the user can toggle elitism on and off. Elitism conserves the selected molecules and makes them also the first molecules on the screen in the next generation.

Experiments

We conducted a computational experiment first for a simple proof-of-concept. For this we used a dataset of biological activities of neuramidase inhibitors[21]. Using the measured activities as input for the evolutionary algorithm we located the experimental minimum (IC_{50} = 1 nM, a 6300-fold improvement over the original structure) within four generations.

To test whether we could use the Molecule Evoluator to discover interesting new molecules with possible biological activity, we performed an experiment using its option for random molecule generation.

First we generated a library of 10,000 small molecules (150D < MW <250D) with drug-like features: either one or two aromatic rings, 5 or fewer rotatable bonds, 2 or fewer hydrogen donors, 4 or fewer hydrogen acceptors, and a polar surface area of at most 70Å^2. Out of this library, three sublibraries of 100 compounds were chosen randomly. Each of these sublibraries was presented to a different chemist, who could choose and modify the molecules created by the program. Out of the 300 compounds, 35 were chosen for further investigation.

Checking the latter molecules in the SciFinder[2] and Beilstein databases[22] (in April 2003) we found that six structures represented chemical classes yet unknown in literature. Based on these six core structures ten molecules were designed, of which eight compounds were synthesized successfully. One typical compound with a calculated log P value of 1.65, a calculated polar surface area of 32 Å^2 and a molecular weight of 191D, is shown in Figure 6. It proved active on both α_1- and α_2-adrenergic receptors, i.e. at 10 µM it displaced more than 50% of the radioligand used in binding assays on the two receptor subtypes. The full results of this experimental proof-of-concept are described in chapter 7.

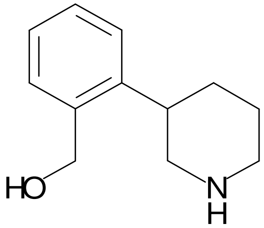


Figure 6. One of the compounds synthesized on the basis of a random generation of virtual molecules by the Molecule Evoluator.

Discussion

In this paper we have presented an evolutionary algorithm to help design new molecules. As reviewed very recently, evolutionary algorithms form one of the approaches to computer-based *de novo* design of drug-like molecules[23]. A large variety of methods is being used to this end, even when confined to evolutionary concepts only. The two most important components of all evolutionary algorithms used in chemistry/drug design are the molecule representation and the fitness function.

The molecule representation

One of the main choices when creating a de novo design algorithm is whether to make the algorithm atom- or fragment-based. Atom-based algorithms work by mutating atoms, and can therefore fine-tune each structure optimally[4,5,13,24]. On the other hand, several investigators construct molecules using larger fragments[7,9,24]. This has the advantage that the representation can be simpler, since there is generally no need for the genome to contain cycles (for these are incorporated into the fragments). Also fragment-based methods have the potential benefit of easier synthesis. While the current version of the Molecule Evoluator uses both atoms and fragments to construct molecules, its mutations are atom-based. We believe that atom-based evolution is superior to fragment-based evolution for adapting the molecular structure. The main disadvantage of using fragments instead of atoms is that most mutations in fragment-based evolution are "macromutations" that change the molecule into something completely different, with a vastly different fitness value. In most cases, it is not clear whether fragment-based evolution is an improvement over random search, unless the fitness function is fragment-based. However, this is certainly not the case in drug design where biological activity is subtly dependent on the molecular structure. We have observed that making an atom-based algorithm interactive, as in the Molecule Evoluator, partially compensates for the disadvantage that molecules generated on the basis of atoms are generally more difficult to synthesize. This is because the medicinal chemist has the option to immediately discard or modify structures at will.

Using the user as a fitness function

The most important problem of the *de novo* design programs which have been described in literature is the difficulty of creating a fitness function that is relevant to drug design. In this work, we propose to use an evolutionary algorithm not as a black box that will give the user the right answer when given the right question, but as a means of supporting the creativity and imagination of the user by interactive evolution, thereby automatically incorporating the user's explicit as well as implicit (subconscious) knowledge of the problem domain.

Using user feedback as fitness function has several advantages and disadvantages, and some consequences that require special adaptations and modifications of the software.

One disadvantage of user interaction is that the population must be small. It is unlikely that any chemist would want to see 50 to 100 molecules before pressing "Go" again. However, small population sizes (12 as a default in the current version of the program) may lead to premature convergence, i.e. tempt the user to stop (too) early. Another hurdle relates to programming the software: the more the user can interact with the program, the more is required from the user interface. In this project, more time was spent on constructing the user interface than on creating and fine-tuning the evolutionary algorithm. Modifications of the evolutionary algorithm should in many cases be reflected by changes in the user interface, and this makes programming and testing new ideas more time-consuming than in a non-interactive system. Software validation is also more difficult – one cannot well run hundreds of tests automatically to objectively verify whether the algorithm outperforms other algorithms. A user is not an objective function that can be easily shared with others. Finally, the user may see the computer program as a competitor, not as a tool, and might not like to work with it since “someone else” has thought of the molecule. While the computer has so little knowledge that one should say that the user has invented the molecule (and recognized it as something interesting), the user may perceive his or her position differently.

There are however also many advantages to user interaction. One attractive advantage is that the feedback from the user can produce molecules which can be synthesized more easily in the laboratory than is possible with computer-generated, random molecules. The difficulty of synthesis would also be automatically adapted to the user’s level of knowledge and experience.

A second advantage is that the program can use all kinds of rules and problem domain knowledge that the user has. The alternatives, expert systems and flexible input, have distinct disadvantages in this case. Creating an expert-system is time-consuming and must be done anew for each optimization project. Flexible input would require the domain expert, namely the chemist, to learn a complicated language or user interface which would definitely diminish accessibility of the software and thereby its use greatly. The program can even benefit from the user’s subconscious rules, which cannot be programmed since they are unknown and may be very difficult to derive. Furthermore, as the user's problem knowledge grows, this knowledge is automatically updated and applied to the process without time-consuming intervention by programmers. In experimental sciences, seldom all required knowledge is known beforehand, and allowing experiments with the computer can also lead to finding new rules and discarding obsolete ones.

A third advantage is that the software can stimulate computer use by medicinal chemists. Far too often, compounds suggested by the "computational department" are rejected by medicinal chemists for reasons of synthesis, and collaboration between the departments is hampered by busy schedules and the necessity to have meetings for feedback. This makes collaboration slow and difficult, and probably results in chemists mainly designing their own compounds without the help the computer could give. We believe that creating a program for the problem domain experts instead of for computer experts can lead to better use of the help that the computer could give in the drug design process, and perhaps even to increased understanding and a more fruitful collaboration between medicinal chemists and computer scientists.

Finally, a program such as the Molecule Evoluator may make a chemist more conscious of his/her own design process, i.e. which rules he or she follows. Consciousness of the rules and methods can lead people to experiment with them and occasionally break them for enhanced creativity.

Adding extra user control to the evolution

We found that when we added interactivity to the evolutionary algorithm, it was not enough to restrict the user's influence to selection only. The users were generally quite "impatient" and wanted more control to accelerate or even directly manipulate the evolution, so we added features to enable this. First, we incorporated edit functions to directly modify the molecular structure. Second, we added an option for selecting a part of the molecule to remain constant. Third, we allowed the settings (which mutations are allowed, what range a property may have) to change interactively. We think that these options will make the Molecule Evoluator more attractive for drug design since they give the user more control over the evolution. We must however beware of the complication that having a feature is not enough if the user does not know that the feature is there or how to use it. Good user interface design is necessary for users to learn to use the multiple filters without having to read the manual. The second danger is perhaps more serious: by discarding "bad molecules" one may at the same time eliminate paths to escape from local optima. Secondly, if all molecules shown are good according to a specific user's criteria, it may be exactly what the user had designed him/herself anyway, thus eliminating the added value of the program. However, lack of control may frustrate and bad structures may irritate the user, so we aimed for a middle road between control and creativity.

Conclusions and future perspectives

In this report we have described the "Molecule Evoluator". It is a software program based on evolutionary algorithms and has been created to aid chemists in designing new drug-like molecules. With this program all relevant chemical mutations are possible. The most distinguishing feature of the Molecule Evoluator relative to other *de novo* design programs is that it has the user as fitness function. In this way it can combine the domain knowledge of the chemist with the memory and processing speed of the computer. We therefore added a graphical user interface for the evolution and extended the program with options for directly editing the molecule, marking part of a molecule as conserved, and calculating relevant physicochemical parameters.

Considering the algorithms used and the feedback from users so far, there are several directions open for future investigation. First, many molecules generated by the program seem difficult to synthesize. Perhaps encoding explicit chemical knowledge in the program or using chemical databases could help improve this. A second direction would be to create a command-line version which links to other software such as docking programs, since the “high-resolution” optimization resulting from our atom-based model might be very useful for optimizing lead compounds. Third, more selection criteria could be added such as additional physicochemical properties or an input method for QSAR-formulas.

Acknowledgements

We thank Hans Brussee, Lisa Chang and Reynier Tromp of the division of Medicinal Chemistry at Leiden University for their feedback during the first stages of development of the Molecule Evoluator, and Siegfried Nijssen for his helpful discussions and providing the first "charf" class implementation to conserve atoms/bonds. Biopartner First Stage Grant and Gruenenthal GmbH are thanked for financial support.

References

[1] Bohacek, R.S.; McMartin, C; Guida, W.C. The art and practice of structure-based drug design: a molecular modeling perspective. *Med. Res. Rev*. **1996**, *16*, 3-50.

[2] As taken from www.cas.org/scifinder/ataglance.html, as accessed on August 18, 2005.

[3] Gillet, V.J. In Evolutionary algorithms in molecular design; Clark, D.E., Ed.; Wiley-VCH: Weinheim, Germany, 2000; Chapter 4, pp 49-69.

[4] Douguet, D.; Thoreau, E.; Grassy, G. A genetic algorithm for the automated generation of small organic molecules: Drug design using an evolutionary algorithm. *J. Comput.-Aided Mol. Des.* **2000**, *14*, 449-466.

[5] Globus, A.; Lawton, J.; Wipke, T. Automated molecular design using evolutionary techniques. *Nanotechnology* **1999**, *10*, 290-299.

[6] Glen, R.C., and Payne, A.W.R. A genetic algorithm for the automated generation of molecules within constraints. *J. Comput.-Aided Mol. Des.* **1995**, *9*, 181-202.

[7] Schneider, G.; Lee, M-L.; Stahl, M.; Schneider, P. De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks. *J. Comput.-Aided Mol. Des.* **2000**, *14*, 487-494.

[8] Pegg, S.C.-H.; Haresco, J.J.; Kuntz, I.D. A genetic algorithm for structure-based de novo design. *J. Comput.-Aided Mol. Des.* **2001**, 15, 911-933.

[9] Vinkers, M.H.; De Jonge, M.R.; Daeyaert, F.F.D.; Heeres, J.; Koymans, L.M.H.; Van Lenthe, J.H.; Lewi, P.J.; Timmerman, H.; Van Aken, K.; Janssen, P.A.J. SYNOPSIS: SYNthesize and Optimize System in Silico. *J. Med. Chem.* **2003**, 46, 2765-2773.

[10] MDL-file format http://www.mdli.com/downloads/public/ctfile/ctfile.jsp as accessed on August 18, 2005. See also: Dalby, A.; Nourse, J.G.; Hounshell, W.D.; Gushurst, A.K.I.; Grier, D.L.; Leland, B.A.; Laufer, J. Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 244-255.

[11] Weininger D. SMILES: a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31-36.

[12] Banzhaf, W.; Nordin P.; Keller, R.E.; and Francone, F.D. *Genetic Programming - An Introduction.* Morgan-Kaufmann, San Francisco CA, 1998.

[13] Nachbar, R.B. Molecular Evolution: A Hierarchical Representation for Chemical Topology and Its Automated Manipulation. In *Genetic Programming 1998: Proceedings of the Third Annual Conferenc*e, (University of Wisconsin, Madison, Wisconsin, July 22-25, 1998). Morgan Kaufmann, San Francisco, CA, 1998, 246-253.

[14] Kamphausen, S.; Höltge, N.; Wirsching, F.; Morys-Wortmann, C.; Riester, D.; Goetz, R.; Thürk, M; Schwienhorst, A. Genetic algorithm for the design of molecules with desired properties. *J. Comput.-Aided Mol. Des.* 2002, *16*, 551-567.

[15] Sheridan, R.P.; Feuston, B.P.; Maiorov, V.N.; Kearsley, S.K. Similarity to molecules in the training set is a good discriminator for prediction accuracy in QSAR. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 1912-1928.

[16] Banzhaf, W. In *Handbook of Evolutionary Computation;* Bäck, T.; Fogel D.B.; Michalewicz, Z., Eds.; Oxford University Press, New York, and Institute of Physics Publishing, Bristol, 1997.

[17] Bentley, P.J. *Evolutionary Design by Computers;* Morgan Kaufmann Publishers, San Francisco, 1999.

[18] Takagi, H. *Interactive Evolutionary Computing: Fusion of the capacities of EC optimization and human evaluation*. Proceedings of the IEEE **2001**, *89*, 1275-1296.

[19] Lipinski, C.A., Lombardo, F., Dominy, B.W., and Feeney, P.J. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug Delivery Rev.* **1997**, *23,* 3-25.

[20] See: www.iupac.org/goldbook/B00732.pdf, as accessed on November 21, 2005.

[21] Kim, C.U.; Lew, W.; Williams, M.A.; Liu, H.; Zhang, L.; Swaminathan, S.; Bischofberger, N.; Chen, M.S.; Mendel, D.B.; Tai, C.Y.; Laver, W.G.; Stevens, R.C.; Influenza neuraminidase inhibitors possessing a novel hydrophobic interaction in the enzyme active site: design, synthesis, and structural analysis of carbocyclic sialic acid analogues with potent anti-influenza activity. *J. Am. Chem. Soc.* **1997**,*119*, 681-690.

[22] http://www.beilstein-institut.de/englisch/1024/chemie/index.php3 , as accessed on August 18, 2005.

[23] Schneider, G.; Fechner, U. Computer-based *de novo* design of drug-like molecules. *Nature Rev. Drug Disc.* **2005**, *4*, 649-663.

[24] Brown, N.; McKay, B.; Gilardoni, F.; Gasteiger, J. A graph-based genetic algorithm and its application to the multiobjective evolution of median molecules. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 1079-1087.

5 Using Data Mining to Improve Mutation in a Tool for Molecular Evolution

Eric-Wubbo Lameijer[1], Ad P. IJzerman[1] and Joost N. Kok[2]
[1]Leiden/Amsterdam Center for Drug Research, Division of Medicinal Chemistry, PO Box 9502, 2300RA Leiden, The Netherlands
[2]Leiden Institute of Advanced Computer Science, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands

This chapter was first presented at the Congress on Evolutionary Computation 2005. Reference: Lameijer, E.W.; IJzerman, A.P.; Kok, J.N. Using data mining to improve mutation in a tool for molecular evolution. Congress on Evolutionary Computation 2005, 314-321.

Abstract

We have developed an evolutionary algorithm-based program for drug design, the Molecule Evoluator. This program transforms known molecules into new molecules which may have improved properties relative to the parent molecule. Transforming the parent molecule into a derivative by mutation is necessary to find molecules with increased fitness. However, mutations that just randomly add and substitute atoms often result in molecules that contain undesirable chemical substructures, and can therefore not be used as drugs. We therefore want to add knowledge to the program about which mutations result in proper chemical structures and which ones do not. In this research we have mined a large chemical database, the World Drug Index, to obtain the frequencies of small substructures in drug-like molecules. Some of our mutation operators were subsequently modified to use these frequencies. Testing the new mutation frequencies on another large database of molecules, the NCI database,

we found that the knowledge-based mutations more often produced existing molecules than the original mutations. This suggests that the modified mutations produce molecules that are easier to synthesize and more drug-like compared to the molecules generated using the original uninformed mutation operators.

Introduction

Pharmaceuticals have a major impact on both public health and the economy. The worldwide sales of pharmaceuticals were 491.8 billion dollars in 2003, and were expected to grow by several billion dollars in 2004 (Class 2004). Pharmaceuticals have also greatly improved human health. Many diseases that used to be dangerous or even deadly, such as pneumonia and tetanus, have become much less dangerous, and people with chronic ailments, such as diabetics and heart patients, live longer and healthier lives thanks to the currently available medication.

There are however still a number of diseases which are difficult to treat or where current therapy has severe side effects. Dementias such as Alzheimer’s disease are yet untreatable, advanced cancer can only be cured in rare cases, and viral diseases remain difficult to fight.

Given this demand for more and better treatments, pharmaceutical companies are continually working to expand their arsenal of drugs. This is however not easy. A medication such as a tablet works since it contains a specific chemical compound, the molecules of which bind to large biological molecules (such as proteins) that are involved in the disease. Drug molecules make these biomolecules more or less active by binding to them. A good drug molecule should influence the biological process of the disease in a beneficial way, but it should also be able to enter the part of the body that needs to be treated, and should not cause side effects which would make the treatment more damaging than the disease itself. It turns out to be very difficult to create such a compound; it is estimated that only one out of 5000 screened candidate compounds reaches the market as a drug (Rees 2003).

The difficulty of finding new drug molecules has pushed the pharmaceutical industry to try to improve its output using both experimental and computational methods. Of these computational methods, the Evolutionary Algorithms (EAs) are widely applied. Not only are they used to gain insight in the structures and functions of genes and proteins (Fogel 2004), which is important for discovering which proteins to target with a drug, they have also been applied to designing the drug molecules themselves, such as designing compound libraries, docking small molecules into proteins, and finding structure-activity relationships. Good overviews are given in the book edited by Clark (Clark 2000) and in chapter 2 of this thesis. However, one of the most interesting applications is the design of new drug molecules themselves, the so-called *de novo* design.

Several EAs for *de novo* design have been described in literature (Brown 2004, Douguet 2000, Glen 1995, Globus 1999, Kamphausen 2002, Nachbar 1998, Pegg 2001, Schneider 2000, Vinkers 2003, and others). The main differences between the various methods are the molecule representations and the fitness functions.

The molecule representation methods in EAs for *de novo* design can be divided into atom-based and fragment-based methods. Atom-based methods build and modify molecules by adding and modifying individual atoms. While atom-based methods can create many more molecules than fragment-based methods, they often create molecules which are difficult to synthesize (Douguet 2000). Fragment-based methods on the other hand create molecules by connecting existing substructures. Fragment-based methods therefore tend to create molecules that are easier to synthesize (though this is not guaranteed (Vinkers 2003)), yet these methods seem less suitable for optimizing molecular structures. Since the mutations replace/add or delete entire fragments (5-20 atoms), each mutation is a macromutation which results in a very different molecule, the fitness of which may be very dissimilar to that of the parent. Also, since fragment-based methods use bigger building blocks which cover only a small fraction of all building blocks theoretically possible, they cannot cover the search space of drug-like molecules as fully as atom-based methods.

In the EA-based program we are developing, the Molecule Evoluator (chapter 4 of this thesis), we want to be able to fine-tune molecular structures and to generate all possible drug-like molecules. Therefore we have chosen for the atom-based representation.

The second important aspect of the EAs for *de novo* design is the fitness function. The fitness functions that are most often used are “docking” (a procedure that fits the molecule into a three dimensional model of the target protein and returns the approximate binding energy), similarity to an existing drug molecule, and experiments. However, these have not yet proven to be very useful since they are either too slow and expensive to apply (experiments), or too inaccurate for optimization (docking, molecular similarity). The best result published so far seems to be a derivative evolved by Schneider (2000), which had a thousand-fold *lower* activity than the lead compound.

We therefore decided to try an alternative approach which could still use the optimization power of an EA but would tap a different source of knowledge, the medicinal chemist him/herself. Since the chemist performs the role of fitness function, the Molecule Evoluator can use the chemist's knowledge about how the molecular structure influences the biological activity and how difficult the synthesis of the molecule would be. In a typical session, the user would take a known molecule to seed the population and let the program generate derivatives by applying various mutations, such as adding or deleting atoms, breaking or making rings, etc. The chemist will then, based on his/her estimates of structure-activity relationships and ease of synthesis, select the most interesting mutants, which will then be used by the program to generate a new population of derivatives.

The general mechanism of the Molecule Evoluator and a screenshot of the program in action are shown in figures 5.1 and 5.2, respectively.

Now we arrive at the main topic of our paper: improved mutation.

One of the consequences of the atom-based mutations in the Molecule Evoluator is that changing an atom into another atom can result in a molecule that is unstable or difficult to synthesize, decomposing before it can exert any effect on the human body. For example, changing a hydrogen atom into a fluorine atom is fine when the hydrogen is attached to a carbon atom, but will result in a highly reactive and therefore un-drug-like compound when the hydrogen is attached to an oxygen atom. Whether a mutation leads to a chemically acceptable molecule therefore depends for a large part on the atoms surrounding the mutation. We therefore want to modify the different mutation functions in such a way that they will result in more reasonable mutants. For this we however need knowledge about which mutations are reasonable.

Our plan is to derive this knowledge from large chemical databases and adapt our mutation operators accordingly.

The outline of the rest of the paper is as follows: first we discuss our data mining approach, then we introduce some of our mutation operators and discuss how we adapted them using the data mining results. Subsequently we will present the results of our experiments with the adapted mutation operators, comparing them with the original operators and finally we will give our conclusions and indicate some of the remaining questions and possibilities to use data mining for an EA such as the Molecule Evoluator.

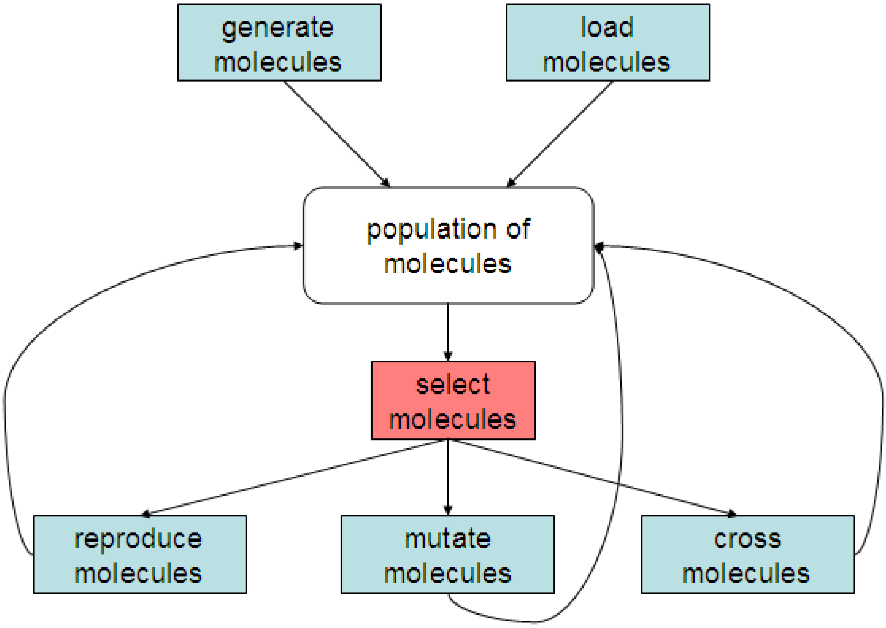
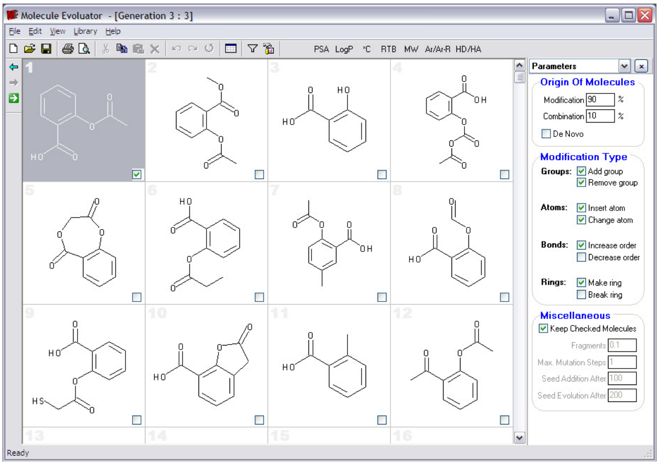


Figure 5.1: Overview of the evolutionary algorithm of the Molecule Evoluator.


Molecule Evoluator - [Generation 3 : 3]
File Edit View Library Help
PSA LogP °C RTB MW Ar/Ar-R HD/HA
Ready

Parameters

Origin Of Molecules

Modification 90 %

Combination 10 %

□ De Novo

Modification Type

Groups: ☑ Add group ☑ Remove group

Atoms: ☑ Insert atom ☑ Change atom

Bonds: ☑ Increase order □ Decrease order

Rings: ☑ Make ring □ Break ring

Miscellaneous

☑ Keep Checked Molecules

Fragments 0.1

Max. Mutation Steps 1

Seed Addition After 100

Seed Evolution After 200

Figure 5.2: The Molecule Evoluator generating variants of acetylsalicylic acid (Aspirin^{TM}), the top left molecule.

Data Mining Approach

The process of discovering knowledge in databases is often called data mining. There are many data mining methods (Witten 2000), able to handle widely different kinds of data. However, the chemical applications of data mining are currently still quite limited. This is probably because the most important data mining one can do in this field is finding the relation between molecular structure and particular properties. This is however very difficult. One of the reasons for this is that many of the existing data sets are relatively small (dozens to hundreds of items). Also, the question remains how to represent a chemical structure (a graph) so that it can be related to its properties. This is still an open question and may depend on the specific application.

One of the problems of the Molecule Evoluator was that many of the molecules created by it were estimated by medicinal chemists to be difficult to synthesize. While of course the chemist can filter out these molecules manually, in general there were so many “bad” molecules per generation that evolution slowed down (many mutants were not attractive, had very low fitness), even to the point that the user, who provides the necessary fitness function, got annoyed.

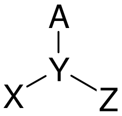
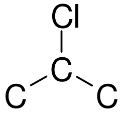
The most obvious fix, as proposed by the chemists, would be a library of “forbidden substructures”. However, this fix has some disadvantages. First, chemical rules are seldom absolute. Many substructures which are not particularly stable can and do occur in drug molecules, only relatively rarely. Eliminating them entirely would make the Molecule Evoluator incapable of finding some real drugs, which would strongly limit its usefulness. Second, forbidding substructures does not solve the problem of frequent versus infrequent substructures. While C-N-C is a perfectly reasonable substructure, it is much rarer than C-C-C, and a program creating equal amounts of both would produce molecules which look rather unusual and still may be difficult to synthesize. Third, all chemists are necessarily subjective and may have studied only about a few thousand structures in their lives, a small fraction of all molecules ever made. A computer can easily search the millions of molecules which have been created so far and can update its knowledge much more quickly.

This gave us the idea to use a large database of drug molecules (the World Drug Index) to find the frequencies of all occurring substructures and adapt the mutation operators so that the mutants will be more drug-like and easier to synthesize than when using “uninformed” mutations.

We used the 2002 edition of the World Drug Index (Daylight 2005), containing approximately 32000 drugs and other pharmacologically active compounds. An in-house program counted the frequencies of all atoms (X), atom pairs (X-Y), atom triplets (X-Y-Z) and sets of four atoms (both the linear X-Y-Z-A and the T-shaped X-Y(-Z)-A). The types of substructures mined are shown in table 5.1. The substructures and their occurrences were collected in a file. The counting algorithm counted all groups starting at each atom in the molecule. So an N-C substructure yields 1 N-C and 1 C-N count, while a C-C substructure yields 2 C-C counts. For our mutations we have corrected for this factor by dividing the counts of symmetric pairs and triplets of atoms by 2, the T-structure-occurrences were divided by 6 if the three atoms surrounding the core atom were identical, and divided by 2 if only two of them were the same.

For computational efficiency, the frequencies of the substructures are read into memory at the start of the program, so for each mutation there are typically only about 10 numbers which have to be fetched and scaled to determine the type of mutation, which is a negligible amount (<0.1%) of the total runtime of the Evoluator.

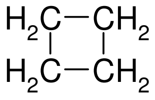
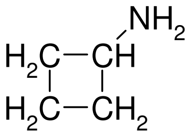
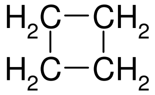
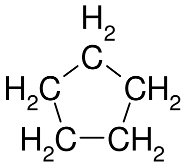
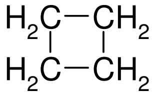
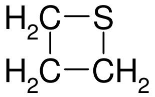
Table 5.1: The substructure patterns mined from the World Drug Index. The letters denote any possible atom type, the '-' characters all possible bond types, to wit single, double and triple bonds.

Substructure pattern	X	X—Y	X—Y—Z	
Example	N	C=C	C—O—C	

Adaptation of the Mutation Operators

To investigate the effects of adding knowledge to the mutation operators, we chose to modify three mutations that the Molecule Evoluator uses to create new molecules. These are the "add atom", "insert atom" and "change atom" mutations, shown in table 5.2.

Table 5.2: The effects of the three mutation operators in this study.

Mutation name	Initial structure	Final structure
Add atom		
Insert atom		
Change atom		

- The "add atom" mutation adds a non-hydrogen atom to the structure by replacing a hydrogen atom by another atom, and adding hydrogens as necessary to fill the remaining bonds of the new atom.
- The "insert atom" mutation takes a single bond between two non-hydrogen atoms and inserts an atom between them.
- The "change atom" mutation changes a non-hydrogen atom into another non-hydrogen atom (so it can exchange a carbon atom for a nitrogen atom, for example).

The three mutations originally used estimated frequencies of the diverse atom types, chosen such that the resulting molecules seemed drug-like. These estimates are shown in table 5.3.

Table 5.3: The initial probabilities to add a certain type of atom to a molecule using the “add” mutation.

Atom type	Add-frequency
C	0.725
O	0.109
N	0.109
S	0.022
P	0.000
F	0.007
Cl	0.014
Br	0.007
I	0.007

However, since the mutations were still context-independent, rare and reactive subgroups such as O-O and N-F did occur much more frequently than would be expected from looking at the World Drug Index. For example, the reactive O-O bond occurs only 1.5 times per 10000 C-C bonds in the World Drug Index, but in the original Evoluator it was generated 62 times per 10000 C-C bonds, 40 times more frequently.

We therefore decided to try and improve the mutations by making them context-dependent. We introduce new versions of three of our mutation operators, the *add atom* mutation, the *insert atom* mutation and the *change atom* mutation.

Add atom mutation: The *add atom* mutation works by picking a hydrogen from the molecule and replacing it by a non-hydrogen atom. This non-hydrogen atom was originally picked from the standard frequency table (table 5.3). We however modified the algorithm to first look at the atom to which the hydrogen was attached (a hydrogen atom has only one bond and is therefore attached to only one atom).

If the hydrogen was attached to an atom of type X, the program looked up the counts of the various X-Y substructures and converted these into a probability table.

This table indicated the probability that the hydrogen atom would be replaced by an atom of type Y. So since there were 8246 C-Cl bonds, $2.5\cdot 10^{-3}$ part of the total number of C-X bonds, and 12 O-Cl bonds, $3.5\cdot 10^{-5}$ of the total of O-X bonds, the probability of substituting the hydrogen by Cl would be $2.5\cdot 10^{-3}$ in the case of a C-H group, and $3.5\cdot 10^{-5}$ in the case of an O-H group.

Based on these frequencies, the Y-type was selected using a random-number generator.

In our experiments we found that using the raw WDI data improved upon our original frequencies. To our initial surprise, however, the substructure frequencies of the resulting molecules were significantly different from the WDI-frequencies. One of the causes seemed to be that carbon atoms, having four bonds, have on average 2-3 hydrogens attached, other atoms less. This skewed the distribution markedly, resulting for example in 60% more nitrogen atoms per carbon than expected. We have improved on this situation by changing the input frequencies by an adaptive procedure until the output frequencies resembled those of the WDI fairly well. This algorithm is depicted as algorithm 5.1. Its results are illustrated in figure 5.3.

Insert atom mutation: When the Molecule Evoluator must choose an atom type to insert between two atoms of type X and Y, the program searches for all X-A-Y-patterns in the substructure database and makes a probability table of how likely it is to find an atom of type A bonded to type X and Y. For example, if the bond is between C and O, the probability of finding a carbon-in-between pattern (CCO) is 0.97, and nitrogen in between 0.012, in contrast to the “raw” probabilities of C versus N of 0.37 versus 0.04 in the entire World Drug Index.

Change atom mutation: The change atom mutation was the most complicated mutation to modify as the atom to be changed can have one, two or three non-hydrogen atoms surrounding it. Depending on the number of surrounding non-hydrogen atoms, the X-A, X-A-Y or X-A-(Z)-Y patterns are looked up and the frequency table is created. Table 5.4 is an example of one of these tables generated by the program.

Algorithm 5.1: Algorithm for iterative refinement of the input two-atom frequencies to produce molecules with similar two-atom substructure frequencies as the World Drug Index.

```
inputFrequencies = WDIfrequencies;
   do
      generate database of molecules using
         inputFrequencies;
      newFrequencies = count two-atom frequencies
         in new database;
      frequencyCorrectionFactors=
         newFrequencies/WDIFrequencies
      mediumCorrectionFactors
         =(frequencyCorrectionFactors + 1 ) / 2
      inputFrequencies = inputFrequencies *
         mediumCorrectionFactors
   while newFrequencies differ significantly from
      WDIFrequencies
```

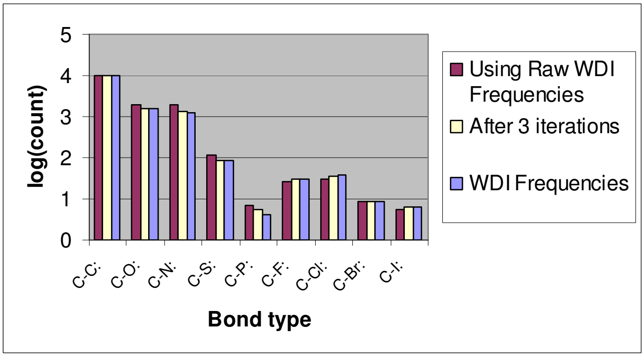


Figure 5.3: Iterative refinement results in the substructure counts getting closer and closer to those of the World Drug Index (all counts are scaled to make the count of CC-bonds 10000). Since the y-scale is logarithmic, some of the gains in accuracy seem smaller than they are: the number of C-N bonds went from 66% too much to 8% too much, and C-Cl from 23% too little to 2% too little.

Table 5.4: Using substructure counts to calculate the probability that an atom flanked by a C and a N atom is changed into a specific other atom type. Substructures which are chemically impossible, such as "C-H-N" in which the hydrogen has two bonds instead of the allowed maximum of one, are indeed not found in the database (count is 0).

Substructure	Count	Probability
C-C-N	323618	0.964
C-H-N	0	0.000
C-O-N	1179	0.004
C-N-N	9060	0.027
C-S-N	1693	0.005
C-P-N	11	0.000
C-F-N	0	0.000
C-Cl-N	0	0.000
C-Br-N	0	0.000
C-I-N	0	0.000

Experiments

After we had modified the three mutation operators to use the data of the World Drug Index, we wanted to find out whether making the mutation operators context-sensitive increased their likeliness to generate "normal" molecules. As a test, we took the database of the National Cancer Institute (National Cancer Institute 2005), 250251 compounds. The NCI database contains molecules which were tested for biological activity, i.e. anti-tumor activity. It has only about 3% overlap with the World Drug Index (Voigt 2001), making it suitable for validation in this study.

The question then remaining is how to validate whether modifying the mutation operators improves the drug-likeness and ease of synthesis of the mutants.

The best proof of this would take an existing compound, apply a mutation to it, and find that the derived compound also exists in the database.

However, as it has been estimated that there are over 10^{60} molecules possible (Bohacek 1996), it seemed unreasonable to demand that our new algorithm would transform all existing NCI molecules into other existing NCI molecules. However, we can reduce the search space by splitting the molecules into fragments (figure 5.4). The chance that a certain fragment is mutated into another known fragment is likely to be much larger than the theoretical molecule to molecule "mutation success ratio" of $2{,}5\cdot 10^{5}/10^{60}$, since the smaller size of the fragments will greatly reduce the search space.

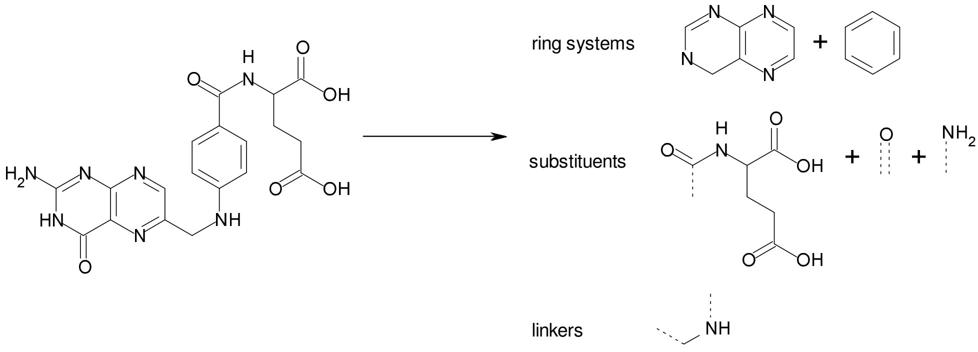


Figure 5.4: Splitting an example molecule, folic acid, into fragments.

The set of fragments we took from the NCI database was the 6765-item 'one-connected branches' set, that is, all non-ring parts of the molecules that were attached to only one ring, such as the =O, $-NH_2$ and –C(=O)NHC(COOH)CCCOOH groups in figure 4. First we removed the branches with atom types that the Molecule Evoluator does not recognize, such as metal atoms (which are extremely rare in drugs) to get a data set of 6564 branches. We then performed atom mutations on the 1000 most frequent branches of the set and recorded how many were mutated into other existing branches. This experiment was repeated 20 times for both the old and new versions of the three mutations. Student's t-test indicated that all three mutations were improved significantly (values ranging from 0.01 to $7\cdot 10^{-14}$). The details of the runs are shown in the appendix at page 152.

The results of the mutation-experiments are shown in table 5.5.

Table 5.5: Average probability that a specific mutation of a NCI branch will produce another NCI-branch for the informed and uninformed mutations. The t-test column contains the probability that the observed differences in performance are due to chance.

	Old average	New average	t-test
Add atom	0.3107	0.3205	0.0108
Insert atom	0.3626	0.4134	6.73E-14
Change atom	0.0722	0.0768	1.4E-07

Discussion

Using knowledge to guide mutation seems to improve upon the old, uninformed methods. Could we improve the Evoluator further by using even bigger substructures to guide the mutations? To some extent, this may be desirable. For example, a C(=O)OC is an ester group, and the second O can be easily exchanged for N, probably more easily than when the (=O) is lacking. However, one should be cautious when interpreting the extra data gained from using larger groups since:

1) Larger groups have lower frequencies, conclusions drawn from them will be statistically less reliable.
2) High occurrence of very large substructures may just reflect existing chemicals, chemical “prejudice” and biological coincidence rather than fundamental rules of chemistry. For example, in the NCI database are many sugar groups attached to purine rings. This however does not reflect any chemical rule that this is especially easy to synthesize but merely the biological coincidence that some nucleosides contain a purine attached to a sugar group, and since many nucleoside-like compounds are active against cancer or HIV, many researchers have made variants of sugar-purine compounds.
3) The further atoms are removed from the atom that is to be mutated, the smaller is their influence; there will be diminishing returns in taking larger and larger substructures, in which the new information will slowly become too noisy to be useful.

So while we could enlarge the substructures used by the mutation operators, the value of such an extension should be critically investigated.

Another extension would be to modify the other mutation operators so that they use the knowledge in the database. This might not be very straightforward for some operators: the “delete atom” would have to go over each 3-atom set in the molecule, and delete atoms with a probability inversely proportional to the occurrence of the 3-atom set relative to the two-atom set. On the other hand, to perfect the “make ring” and “break ring” mutations we would have to mine the distribution of ring sizes and ring frequencies instead of the substructure frequencies. Mining and implementing the acquired data would probably be quite straightforward in that case.

Data mining to modify the mutation operators seems useful in this investigation. While we do not know the extent in which data mining to make mutations context-dependent is applied in the EA community, it may be quite useful when there are large databases available of reasonably fit individuals. However, the question remains whether an individual does not exist in the database because it is not fit or just because it has not been thought of yet. A “negative” database of very unfit individuals could help resolve such cases, though these unfortunately do not seem to be very prevalent, at least not in the drug development community.

In any case, it seems useful to add data to our EA. Of course, we should apply the mining with care, and not impose so many rules on newly generated compounds that they cannot have novel or interesting structures anymore. Using our non-informed mutations, the Molecule Evoluator was probably a bit too much “original”, making large parts of the offspring molecules unsuitable for further evolution. Data mining will help to diminish this percentage of molecules and thereby speed up evolution. However, the logical limit would be requiring absolute certainty that a compound can be made. And that can currently only be reached by knowing that the molecule already exists in a molecular database. This would unfortunately preclude finding any novel molecules and greatly limit the optimization. We should therefore find a way between unpractical novelty and conservative clichés. But such a discussion will only be about how important we allow data mining to become in our approach: that data mining is useful, is clear.

Conclusions

In this research, we have improved the mutations of the molecule design program “The Molecule Evoluator” by using data mined from a drug database. Using the counts of various small substructures in the database, we found that the amount of unusual substructures decreased, and that mutations had a larger chance to transform existing molecular fragments into other existing molecular fragments. We think that the enhanced generation of existing structures additionally suggests that also the non-existing mutants generated by our “informed” mutation operators may be closer to molecules that are drug-like and can be synthesized than the mutants generated by the uninformed mutations are. Mining databases may be a good method to making the *de novo* generated molecules easier to synthesize while keeping the advantages of covering the full space of drug-like molecules and the likely faster and more robust optimization that atom-based molecule optimization can give.

Acknowledgments

The authors thank Thomas Bäck for his help in improving this document and for the ongoing inspiration and support he provides to the Molecule Evoluator project.

References

Bohacek, R.S., McMartin, C., and Guida, W.C., The Art and Practice of Structure-Based Drug Design: A Molecular Modeling Perspective. *Medicinal Research Reviews 16* (1996) 3-50.

Brown, N., McKay, B., Gilardoni, F., and Gasteiger, J. A Graph-Based Genetic Algorithm and Its Application to the Multiobjective Evolution of Median Molecules. *Journal of Chemical Information and Computer Sciences 44* (2004), 1079-1087.

Class, S. Health care in Focus. *Chemical & Engineering News,* Dec 6th 2004, 18-29.

Clark, DE (ed) (2000) Evolutionary Algorithms in Molecular Design. Wiley-VCH, Weinheim. Daylight (2005) http://www.daylight.com/products/ databases/WDI.html

Douguet, D., Thoreau, E. and Grassy, G. A genetic algorithm for the automated generation of small organic molecules: Drug design using an evolutionary algorithm. *Journal of Computer-Aided Molecular Design 14* (2000), 449-466.

Glen, R.C., and Payne, A.W.R. A genetic algorithm for the automated generation of molecules within constraints. *Journal of Computer-Aided Molecular Design 9* (1995), 181-202.

Globus, A., Lawton, J. and Wipke, T. Automated molecular design using evolutionary techniques. *Nanotechnology 10* (1999), 290-299.

Kamphausen, S., Höltge, N., Wirsching, F., Morys-Wortmann, C., Riester, D., Goetz, R., Thürk, M. and Schwienhorst, A. Genetic algorithm for the design of molecules with desired properties. *Journal of Computer-Aided Molecular Design 16* (2002), 551-567.

Nachbar, R.B. Molecular Evolution: A Hierarchical Representation for Chemical Topology and Its Automated Manipulation. In *Genetic Programming 1998: Proceedings of the Third Annual Conference* (University of Wisconsin, Madison, Wisconsin, July 22-25, 1998). Morgan Kaufmann, San Francisco, CA, 1998, 246-253.

National Cancer Institute (2005) http://cactus.nci.nih. gov/ncidb2/download.html. Most recent access May 2005, last version of 2D database (August 2000) used.

Pegg, S.C.-H., Haresco, J.J., and Kuntz, I.D. A genetic algorithm for structure-based de novo design. *Journal of Computer-Aided Molecular Design 15* (2001), 911-933.

Rees, P. Big pharma learns how to love IT. *Scientific Computing World* (2003), 16-18.

Schneider, G., Clément-Chomienne, O., Hilfiger L. Schneider, P., Kirsch, S., Böhm, H.-J., and Neidhart, W. Virtual screening for bioactive molecules by evolutionary de novo design. *Angew., Chem. Int. Ed. 39* (2000), 4130-4133.

Vinkers, M.H., De Jonge, M.R., Daeyaert, F.F.D., Heeres, J., Koymans, L.M.H., Van Lenthe, J.H., Lewi, P.J., Timmerman, H., Van Aken, K., and Janssen, P.A.J. SYNOPSIS: SYNthesize and Optimize System in Silico. *Journal of Medicinal Chemistry 46* (2003), 2765-2773.

Voigt, J.H., Bienfait, B., Wang S., and Nicklaus M.C. Comparison of the NCI Open Database with Seven Large Chemical Structural Databases. *Journal of Chemical Information and Computer Sciences 41* (2001), 702-712.

Witten, I.A., Frank, E. (2000) "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations", Academic Press, San Diego.

Appendix: Comparing the old with the new mutation operators

To assess whether incorporating mined data into the mutation operators of the Molecule Evoluator resulted in producing more realistic molecules, the 1000 most frequently occurring one-connected branches of the NCI database were mutated using both the knowledge-less and the knowledge-including versions of three mutation operators. The old versions did not use any substructure frequency data, the new versions used this data to calculate the relative frequencies of each substitution. The number of branches in each run that was mutated into one of the other 6564 branches was counted for 20 runs (maximum score of each run is 1000). In the table the results of the individual runs are shown, as well as the averages over the runs and the *t*-test probabilities that the new mutation versions differ from the old mutations.

Run Index	Old Add	New Add		Old Insert	New Insert		Old Change	New Change
1	299	305		373	416		68	75
2	309	355		344	406		77	77
3	317	326		356	418		75	78
4	303	315		374	401		69	77
5	316	317		350	423		74	78
6	305	320		354	413		72	78
7	320	297		355	408		74	76
8	317	350		387	410		78	77
9	302	309		376	419		71	79
10	313	317		349	421		72	75
11	300	324		349	400		69	75
12	311	318		361	419		74	77
13	312	323		365	415		69	77
14	325	318		356	412		72	78
15	314	328		352	429		72	77
16	300	321		389	419		71	76
17	316	302		365	404		71	76
18	305	331		371	409		73	76
19	324	321		388	404		70	75
20	306	312		337	422		72	78
AVERAGE	310.7	320.45		362.55	413.4		72.15	76.75
TTEST	0.0108			6.73E-14			1.4E-07	

6 Evolutionary algorithms in *de novo* molecule design: comparing atom-based and fragment-based optimization

Eric-Wubbo Lameijer,[†] *Chris de Graaf,*[‡] *Daan Acohen,*[†] *Chris Oostenbrink,*[‡] *and Ad P. IJzerman*[†,*]
Leiden/Amsterdam Center for Drug Research, Division of Medicinal Chemistry, Leiden University, PO Box 9502, 2300RA Leiden, The Netherlands, and Leiden/Amsterdam Center for Drug Research, Division of Molecular Toxicology, Department of Chemistry and Pharmacochemistry, Vrije Universiteit, De Boelelaan 1083, 1081 HV Amsterdam, The Netherlands.

Abstract

Traditionally, drugs have been discovered by scanning libraries of natural and synthetic compounds. Compounds that were identified as having a desirable biological activity were subsequently optimized for use as drugs, a process being performed by medicinal chemists suggesting and trying out structural modifications. Nowadays, however, there are also investigations to whether the design process can be sped up by letting computers do part of the molecule design. In theory, computers could generate and (virtually) test many more structures than a medicinal chemist could design in a similar amount of time, possibly yielding better compounds at a smaller cost. Of these computational methods for drug design, a prominent category is that of evolutionary algorithms, which attempt to optimize drug molecules similar to how nature optimizes animals and plants: by mutation, cross-over and selection. While the general approach of using evolutionary algorithms seems promising, current literature lacks comparisons on which approaches and parameter settings work better or worse for drug design. Since evaluation of new compounds is expensive, whether it is done computationally or experimentally, it is important to develop efficient evolutionary algorithms that

require as few evaluations as possible to find molecules with higher drug-likeness or enhanced affinity to the target protein. In this study, we investigate the effect of one of the major choices in evolutionary algorithms for drug design: what difference it makes in practice whether molecules are built up from atoms or from multi-atom building blocks.

Introduction

It is hard to develop a molecule that can activate or inhibit a particular disease-related enzyme or receptor, since there is usually barely any information on what kind of structure would be needed, nor on how to adapt a lead molecule to improve affinity or selectivity. Therefore, much effort goes into synthesizing and screening large numbers of compounds, in the hope that a large amount of trial and error will discover compounds with the desired properties (Rees, 2003).

However, investigators hope to be able to replace at least part of the expensive “wet” trial and error with “virtual” trial and error, on the computer. Nowadays, many computer programs exist that can aid in the design of new molecules and molecule libraries. Such approaches and programs have been summarized in a number of reviews (Westhead, 1996; Gillet, 2000; Hann, 2000), as well as in chapter 2 of this thesis. A major class of these computational methods applied in *de novo* drug design is that of the evolutionary algorithms (EAs). EAs are promising for drug design, since they mimic the powerful optimization process of natural evolution. Biological evolution can be considered to be an adaptation of designs (organisms) to optimally solve a specific problem (fill a biological niche). Evolutionary algorithms, following that example, also mutate and combine designs, and preferentially procreate designs with the highest quality (‘fitness’). EAs have been applied to fields as diverse as stock market prediction, optical systems, and car safety (Carter, 2005; Koza, 2005; Tan 2005). Not surprisingly, there have been quite a few applications in drug design too, varying from docking to library design to QSAR (Hopfinger 1996; Jones, 1997; Morris, 1998; Kimura, 1998; Gillet, 1999; Sheridan, 2000). In this investigation, however, we focus on their application in *de novo* molecule design.

Evolutionary algorithms have been applied quite often to *de novo* molecule design (Payne, 1995; Nachbar, 1998; Globus, 1999; Douguet, 2000; Schneider, 2000; Pegg, 2001; Vinkers, 2003; Brown, 2004; Dey, 2008; Nicolaou, 2009), and most authors have claimed some success in optimizing molecular structures. However, the abundance of methods hides the fact that we actually know very little about what kind(s) of evolutionary algorithm should be used for drug design.

While each published method has been claimed to have some success in designing new lead compounds, that unfortunately is not a good guide: all optimization methods, even random search, will eventually produce results that improve upon a given starting situation. If we assume that all optimization methods can theoretically produce all possible drug-like molecules, the main question is not so much *whether* a particular method can optimize molecules, but how *efficiently* it does so. If a search method is more efficient, it is superior. And efficiency in this field is not just a theoretical concern. When designing compounds for real world applications, it will always be necessary to synthesize and test a number of compounds, with all associated costs. Therefore, it matters a great deal whether an optimization process needs one hundred or one hundred thousand molecules to improve the activity of a lead compound to a certain extent. In this respect, it is unfortunate that the investigations described so far use different evolutionary approaches and different test systems, since this makes it difficult to compare evolutionary methods and settings, and find out which work best.

In our opinion, a good start of the investigation of how to design an evolutionary algorithm for drug design is by investigating the impact of which is perhaps the main decision for every EA designer in this field: the choice of molecular building blocks. From which "units" molecules are constructed determines in which ways a molecule can be changed and optimized, and which molecules can and cannot be created by the EA. For example, if the basic building set does not include halogen atoms, a large part of chemical space, including drugs such as fluoxetine and haloperidol, cannot be found. And if the building blocks are large, over 10 atoms, the evolutionary algorithm will not produce many small molecules.

In general, EA designers choose one of two main options. The first option is to consider atoms and bonds as the basic building blocks of molecule construction. This is called "atom-based" evolution. Alternatively, a molecule can be considered to be built out of several multi-atom fragments, considered to be unchangeable semi-independent units, like a carboxylic acid group, a phenyl ring, etc. This is called "fragment-based" evolution. The choice between atom-based evolution and fragment-based evolution influences which molecules can be designed by the evolutionary algorithm, in which ways molecules can be mutated and combined, and how much difference there is between a molecule and its offspring. The choice between atom- and fragment-based evolution could therefore have a great influence on the molecule optimization process, yet we could not find any previous investigations into the effect of building block choice. We therefore chose to investigate this factor.

Literature shows that many if not most investigations (Payne, 1995; Nachbar, 1998; Globus, 1999; Douguet, 2000; Brown, 2004) use the atom-based approach. Typical mutations are removing an atom, adding an atom, breaking a bond, or making a bond, for example in ring formation. Sometimes a method uses both atoms and fragments, such as the investigation by Nicolaou et al. (Nicolaou, 2009), but even then the method retains the basic benefits and drawbacks of atom-based approaches: since every individual atom can be changed, the entire chemical space can be covered, but some of the molecules suggested may be difficult to synthesize or chemically unstable.

Fragment-based evolution has also been applied by researchers (Schneider, 2000; Pegg, 2001; Vinkers, 2003; Dey, 2008), be it perhaps less frequently than the atom-based approach. The main difference with the atom-based approach is that molecules are considered to consist of several independent multi-atom units, which are unchangeable. This means that an isopropyl-fragment could never be mutated into a *t*-butyl fragment, though it could be replaced by one. Fragment-based mutations commonly add fragments to a molecule, remove them, or combine the fragments of two molecules. Fragment-based evolution also requires the designers to make some additional choices: what kind of fragments should be used, which and how many fragments should be used, in which ways are the fragments allowed to be attached to one another, and should one take the similarity of fragments into account when mutating a molecule (and if so: how?).

It is not clear from existing literature whether atom-based or fragment-based evolution should be preferred, as no study has compared them yet on the same problem. The main advantage of fragment-based evolution over atom-based evolution is that the produced structures seem easier to synthesize. While this apparent ease of synthesis can be quite deceptive in practice (Vinkers, 2003), it is certainly preferable over the much larger chance that a molecule created by atom-based algorithms needs major modifications before synthesis can take place (see for example the work of Douguet (Douguet, 2000)). On the other hand, atom-based evolution has advantages too. First of all, atom-based evolution does not require one to make the–always somewhat haphazard–choice of building blocks and attachment rules. Secondly, atom-based evolution can in theory construct all possible molecules, in contrast to the fragment-based approach which can typically “only” create 10^{10} to 10^{20} (Pegg et al. estimate a lower bound of 10^{12} for their HIV-RT library (Pegg, 2001)). Being able to create 10^{12} molecules may look impressive, however it represents only a tiny fraction of the total number of possible drug-like molecules, estimated to be 10^{60} or higher (Bohacek, 1996). This implies that fragment-based evolution can only construct about 1 of every 10^{40} possible molecules, which is far less than one molecule in a database containing all compounds currently known to man (which would have about 10^7-10^8 entries, as the largest compound databases such as CAS (CAS, 2009), Beilstein (Beilstein, 2009) and PubChem (Pubchem, 2009) contain about 48 million, 10 million and 19 million entries respectively as around June 2009. Thirdly, the big jumps taken by fragment-based evolution through changing 5-10 atoms simultaneously may make optimization much more difficult than when only one or two atoms are changed at a time. This could be compared to carving a statue with a pickaxe instead of with a chisel: it is almost impossible to exert the precise and subtle control needed for creating the exact result one wants. Also, having fewer choices may result in fewer pathways to get to one's destination, and more dead ends.

Arguments such as the above can theoretically justify preferring either atom- or fragment-based evolution. However, the real relative efficiencies of the approaches are yet unknown since investigators have used one method or the other, and a comparison of atom-based evolution versus fragment-based evolution has not been reported. We therefore decided to undertake a study to find out what the differences in performance are, if any, between atom-based and fragment-based evolution.

We compared atom-based with fragment-based evolution by creating a population of 50 molecules, and evolving it for 10 generations with either method. To simulate the drug optimization process (at least the optimization of affinity), we decided to optimize molecule binding to HIV-reverse transcriptase (HIV-RT), one of the major targets for AIDS therapy. HIV-RT was used in earlier fragment-based evolution studies (Vinkers 2004, Pegg 2001), and continues to be of interest to drug designers and a test for de novo design methodologies (for example Jorgensen, 2006; Barreiro, 2007). Therefore, HIV-RT seems more or less a benchmark drug target that can make comparisons between different de novo design algorithms easier.

The fitness of the molecules during the evolution was defined as their docking score in HIV-RT. While docking scores are not very accurate for predicting binding affinity of a potential drug to a protein, of all evaluation methods it seems best suited to base our investigations on. The highly complex interactions between a flexible ligand and a heterogeneous, irregularly shaped cavity, as modeled by docking, resemble the biological binding process more than for example 2D similarity to known ligands does. We do not expect docking in combination with the evolutionary algorithms described in this paper to result in 'true leads' for RT-inhibitors, as docking algorithms are not yet accurate enough for that purpose. However, for an investigation into making evolutionary algorithms as efficient as possible in designing drug-like molecules, docking is probably a good approximative method to weed out inefficient evolutionary algorithms, before performing the final fine-tuning with the more realistic but also much more expensive and time-consuming synthesis and biological testing of compounds. One could compare this to first testing a probe for exploring the planet Mars in a cold desert on earth: while the circumstances of the test are not totally accurate, it is a much easier, faster and cheaper method to detect flaws than simply sending the design to Mars for the real test.

Specifically, the fitness of each molecule was defined as its docking score into the crystal structure model of HIV-RT using the docking program GOLD (Jones, 1997). The GoldScore is a unitless quantity which should be increased to obtain more favorable affinities. We use the same crystal structure as Vinkers (Vinkers, 2003) and Pegg (Pegg, 2001), pdb-code 1RTI (Ren, 1995). However, since the scoring functions in this investigation differs from that of, for example, Vinkers, we cannot easily compare our optimized molecules with those of others. Different scoring functions mean that molecules optimized for our scoring function may not do so well in Vinkers' fitness measure, and vice versa. However, if ever more comparative studies such as this one are undertaken, 1RTI seems a good place to start.

In this paper we will first discuss the evolution settings, the mutations we used, and the docking approach taken. After that we will focus on the structures of the compounds and on the improvements in docking scores over the course of the atom- and the fragment-based evolution. Finally, we will compare our methods and results with those of others and suggest directions for further investigations.

Methods

Description of the algorithm

The evolution experiments required several steps: generating the initial population of molecules, selecting and modifying the molecules, and evaluating their fitness by docking and scoring them. The general algorithm of the evolution is given in Algorithm 6.1. The following paragraphs will discuss the various steps in more detail.

Algorithm 6.1: The basic algorithm used for evolving molecules with a high fitness (in our case, a high docking score).

```
A) Create initial population
   -while the initial population contains fewer than 50
    molecules:
      -generate a molecule out of NCI fragments
      -add this molecule to the population if it satisfies
       the physicochemical restraints.
B) Perform evolution
   -while there are fewer than 10 generations
      -calculate the fitness of each molecule in the current
       generation by docking it into HIV-RT
      -if the current generation is not the initial
       generation:
         -select 10 molecules by tournament selection
          from the previous generation
         -add these molecules and their fitnesses to the
          current generation
      -make a new generation next to the current generation.
      -while there are fewer than 45 molecules in the new
      generation
         1] choose one molecule from the current generation
            by tournament selection.


         2] choose crossover or mutation
            -if crossover:
               -select a second molecule from the current
                generation by tournament selection
               -cross the two molecules, and pick one of
                the products at random
               -if crossover cannot be performed since
                there are no suitable breaking points, go
                back to step 1.
            -if mutation:
                -pick a mutation type at random
                -perform the mutation on the selected
                 molecule
                -if the mutation cannot be performed due to
                 lack of suitable atoms or bonds, go back
                 to step 1.
            -if the resulting molecule obeys the
             physicochemical constraints and was not part of a
             previous generation:
                -add the new molecule to the new generation
        -endwhile
        -for j=1 to 5
            -generate a molecule at random
            -if the molecule obeys the physicochemical
             constraints:
                -add the molecule to the new generation
        -rename the current generation to "previous
         generation", and the new generation to "current
         generation"
    -endwhile
```

The molecules of the initial population were built out of fragments from the NCI database (NCI, 2000). To obtain these fragments, the molecules in this database were divided into parts by an algorithm we described in chapter 3, which splits molecules into fragments by breaking the bonds which connect the ring systems with the rest of the molecule. This splitting results in ring systems, branches and linkers, as illustrated in Figure 6.1. The branches and linkers also store information about the atom type of the ring atom(s) they were attached to, to help virtual "reverse synthesis".

Splitting the molecules of the NCI database across ring attachment bonds also resulted in data about their fragment composition. For example, 7.5% of all molecules consisted of exactly two ring systems, two branches, and one linker connecting two fragments, like the acetophenazine molecule in Figure 6.1. This information was also used in constructing the random molecules of the initial population. Each molecule was created by first picking a fragment composition (for example, a 7.5% chance to have the "2-2-1" composition), and then picking the fragments themselves out of the 300 most occurring ring systems and the 300 most occurring non-ring systems (branches and linkers). The probability of a particular fragment being selected was proportional to its occurrence in the NCI. Since our database however also incorporated data to which atom type each non-ring fragment was attached, initially selected fragments for which there was no suitable atom type in the rings to connect to were replaced by other fragments. For example, the keto group created by dissecting the molecule in Figure 1 would be stored as (bc)-C(=O)C, to indicate that it can only be re-attached to a carbon atom, and not to for example a nitrogen atom in piperidine. In some cases, the atom type demanded by a particular fragment was not available in the fragments that should be linked to it: for example, a methyl fragment that in the NCI was attached to a nitrogen atom ((bn)-CH_3) could not be attached to a phenyl ring (other fragments, like (bc)-CH_3, however, could).

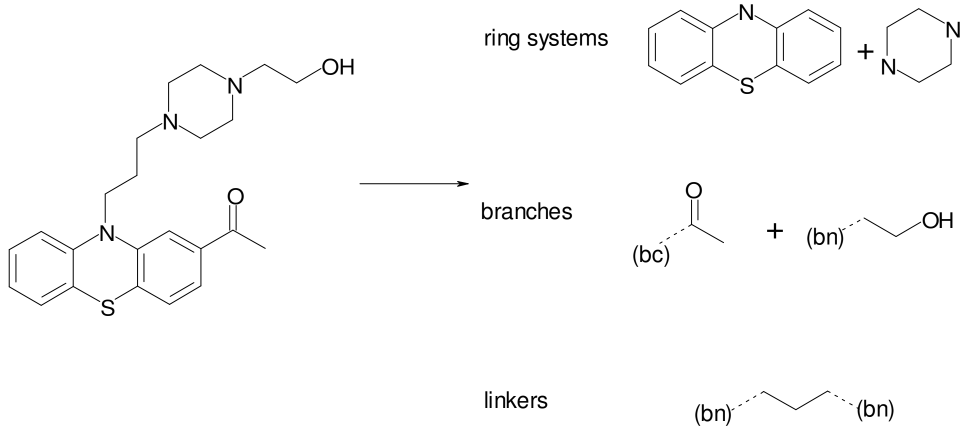


Figure 6.1. The fragments used in molecule construction and fragment-based evolution were obtained by splitting the molecules in the NCI database into ring systems, linkers and branches. This example shows which fragments would be obtained from an acetophenazine molecule.

Molecules were generated until there was a population of 50 molecules that satisfied the following demands that we considered suitable for lead-like molecules: 1) polar surface area (calculated according to (Ertl, 2000) smaller than 120 Å^2, 2) molecular weight between 150 and 300, 3) at most 5 hydrogen bond donors, 4) at most 10 hydrogen bond acceptors and 5) at least one hydrogen bond acceptor. In addition, a maximum of 5 rotatable bonds was allowed per molecule.

After the initial population was generated, it was evolved (via atom- and fragment-based evolution) as follows. First the fitness scores of all molecules were calculated by docking them with GOLD in the HIV-RT crystal structure and selecting the best ranked pose for every molecule (the preparation of the molecules for docking in GOLD, and the settings used in GOLD, are described in more detail on page 167). Then the next population was made by first creating 45 molecules through mutating and crossing over parent molecules, selected by 4-sized tournament selection, which means that out of four random molecules in the parent population the molecule with the highest docking score was chosen. The 45 new molecules also had to obey the constraints listed in the previous paragraph, be it that after the first population up to 10 rotatable bonds were allowed, and the maximum molecular weight was increased to 500. In the last step, 5 molecules were added from the 200-molecule database containing random molecules similar to those of the initial population (the initial population actually consisted of molecules 1-50 of this database, the five molecules added to the first generation of the atom-based evolution were molecules 51-55, the five molecules added in the second generation of the atom-based evolution were molecules 56-60, etc. The fragment-based evolution added other series of five molecules from this same database, so a random molecule would never appear more than once in either the atom-based evolution or fragment-based evolution). The new population therefore also consisted of 50 molecules in total.

Since we did not want to evaluate molecules more than once, we introduced the restraint that molecules created by mutation and crossover would only be added to the new population if they had not occurred yet in previous generations. This could, however, hamper optimization, since the best molecules would be forgotten after one generation. In contrast, most evolutionary algorithms allow a good individual to survive indefinitely and to even clone itself, so it can eventually fill the entire population. We tried to avoid such loss of diversity yet conserve the memory of the best molecules by adding some old molecules to each generation. After the 50 molecules of a population were generated in the normal way, we added 10 of the best molecules of the previous generation (selected by tournament selection), making the effective population size 60. The next generation was created by mutating and crossing those 60 molecules. So, in principle, a superior molecule from generation 1 could contribute its genetic information directly to several generations by first being a parent for the molecules in the next generation (generation 2), and subsequently be added to the generation of its children by being chosen as one of the ten grandparents (so forming the extended generation 2 together with its direct offspring). In this way, it can produce offspring into the generation of its grandchildren (generation 3). As a member of the extended generation 2 it could be also added to the generation 3 as one of the best of the generation 2, into the extended generation 3, becoming one of the sixty parents for the molecules in generation 4. While the randomness of tournament selection makes it unlikely that any molecule, no matter how good, will remain producing offspring for more than three generations, it will buy the best molecules two or three generations extra time to produce high quality offspring.

Manual filtering of molecules

Occasionally, a molecule generated by atom-based evolution seemed extremely difficult to synthesize. This was often the result of the evolutionary algorithm creating an extra bond within a benzene ring or another small ring. Examples are given in Figure 6.2. To prevent evaluation, evolution and accumulation of these 'useless' structures, we visually inspected all molecules created and removed the structures deemed unsynthesizable (to not fall below our target of 45 structures, we actually created initial populations of 50 molecules, of which the last five molecules were spares which could be used to replace molecules among the official 45 if any were flawed). Filtering did not take much time for these small-scale experiments, and molecules were only rarely rejected (typically two or three molecules out of 50). We have recently added a ring structure filter (based on existing NCI rings and NCI ring templates), which will probably make such manual filtering unnecessary in future experiments.

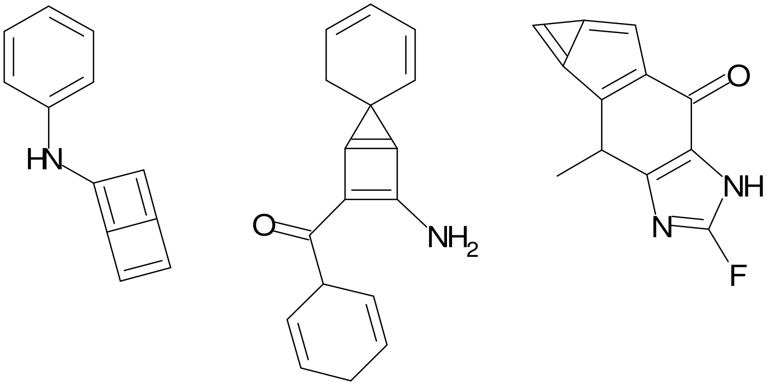


Figure 6.2: Atom-based evolution occasionally produced structures which contained ring systems that seemed far too strained to allow for a feasible synthesis. Such structures (a few examples are shown here) were manually removed from the population to prevent them from "polluting" the evolution.

Molecule modifications

All evolution experiments were performed with the Molecule Evoluator (chapter 4), which was adapted to include fragment-based mutations and crossover. Both the atom-based mutations (which were already present in the Molecule Evoluator) and the fragment-based mutations (which were added during this investigation) will be described in the next paragraphs.

The atom-based mutations either manipulate atoms (adding atoms, inserting atoms in a bond, removing atoms, 'uninserting' atoms, changing atom type) or bonds (increasing bond order, reducing bond order, making or breaking rings). These nine operators were also described in chapter 4.

While the mutations create a derivative from one parent molecule, the atom-based crossover operator combines two input molecules. The crossover first chooses a random non-ring single bond in each molecule, and, by breaking these bonds, splits the molecules into two parts. Subsequently, one part of the first molecule is recombined with one part of the second molecule, and the other part of the first molecule is recombined with the other part of the second molecule. This results in two offspring molecules, of which one is chosen randomly as the crossover product. The crossover is shown in Figure 6.3.

The second approach, fragment-based evolution, considers a molecule to consist of a small number of fragments, and its mutations therefore replace and move bigger parts of the molecules than the atom-based approach does. The mutations and crossover move the fragments which have been defined in the previous paragraphs: ring systems, linkers, and branches. In our implementation the fragment-based evolution uses two types of mutations: "rotation", in which a ring substituent is moved to another position of the same ring, and "exchange", in which two fragments that are connected to the same ring or linker swap positions. Examples of these mutations are given in Table 6.1. It should be noted that branches should always be attached to the same atom type to which they were originally attached, since this attachment information is part of our branch data (as explained above).

The fragment-based crossover, like its atom-based counterpart, takes two molecules, but instead of breaking a random bond, only breaks the bonds that connect fragments. Therefore, only intact fragments are moved around and recombined with each other. The requirements of the fragment-based crossover are that the fragments to be exchanged are of the same type (ring or non-ring) and that, if the fragments are rings, they are attached to the same number of non-ring fragments. For example, a phenyl ring with three substituents and a cyclobutadiene ring with two substituents

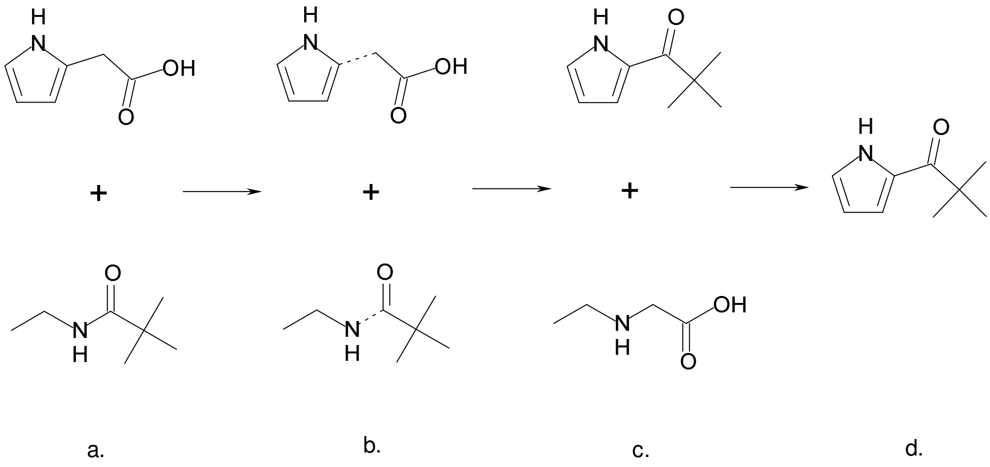
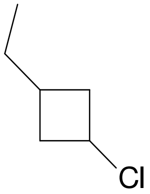
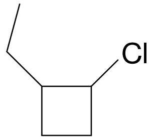
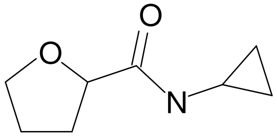
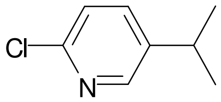
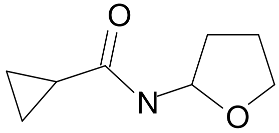
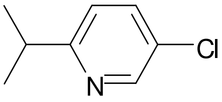


Figure 6.3: The molecule crossover used in atom-based evolution. a) Two parent molecules are selected, b) the molecules are split in two by breaking random non-ring bonds, c) the molecule parts are recombined, d) one molecule is selected at random as the crossover product.

cannot be interchanged, while a phenyl ring with two substituents could be interchanged with the cyclobutadiene. This second requirement was introduced to improve ease of synthesis, since it could prevent too big steps in chemical space, like for example replacing a phenyl ring with three substituents by a cyclopropane ring which usually has only one or two. Lastly, the branches and linkers should be reattached to their correct atom types; the crossover will not link a cyclohexane ring to a (bn)-isopropyl fragment. The fragment-based crossover is illustrated in Figure 6.4.

In both atom-based and fragment-based evolution, the mutation:crossover ratio was set to 85:15, based on experience with the interactive version of the Molecule Evoluator. All mutation subtypes were applied with equal probability; so the probability of applying one of the nine atom-based mutations, such as “insert atom”, was 11%, and the fragment mutations ‘exchange’ and ‘rotate’ both had a 50% chance of being chosen. While these ratios are almost certainly not optimal, we chose them as the starting point for this study, to be adjusted for future investigations depending on their apparent relative usefulness we would find here.

Table 6.1: The mutations used in fragment-based evolution. Rotation moves branches (non-ring structures) around rings, 'exchange' exchanges the positions of either two rings or two branches.

Mutation type	Initial structure	Final structure
Rotation		
Exchange	 	 

Protein preparation and automated docking simulations

The crystal structure we used, (pdb-code 1RTI (Ren, 1995)) is a co-crystal of the HIV-RT with a small-molecule inhibitor, HEPT (1-(2-hydroxymethyloxymethyl)-6-phenyl thiothymine). Despite the similarity of this ligand to a nucleoside, HEPT does not bind to the site where the new DNA is synthesized. Instead, like nevirapine, it binds at some distance from it, at the so-called non-nucleoside binding site (Ren, 1995). Compounds binding at the non-nucleoside site force HIV-RT in a non-active conformation. Figure 6.5 shows the binding mode of HEPT in the 1RTI crystal structure. In some HIV-RT-ligand crystal structure complexes, H-bond interactions between the ligand and a water molecule located in the solvent channel are observed (Esnouf, 1997; Shen, 2003). While in many cases it is advisable to keep the water molecules in or near the binding site when docking (Klebe, 2006), this water molecule is not observed in the 1RTI crystal structure and is not conserved amongst all HIV-RT-ligand crystal structure complexes. Furthermore, it was not found to significantly influence automated docking

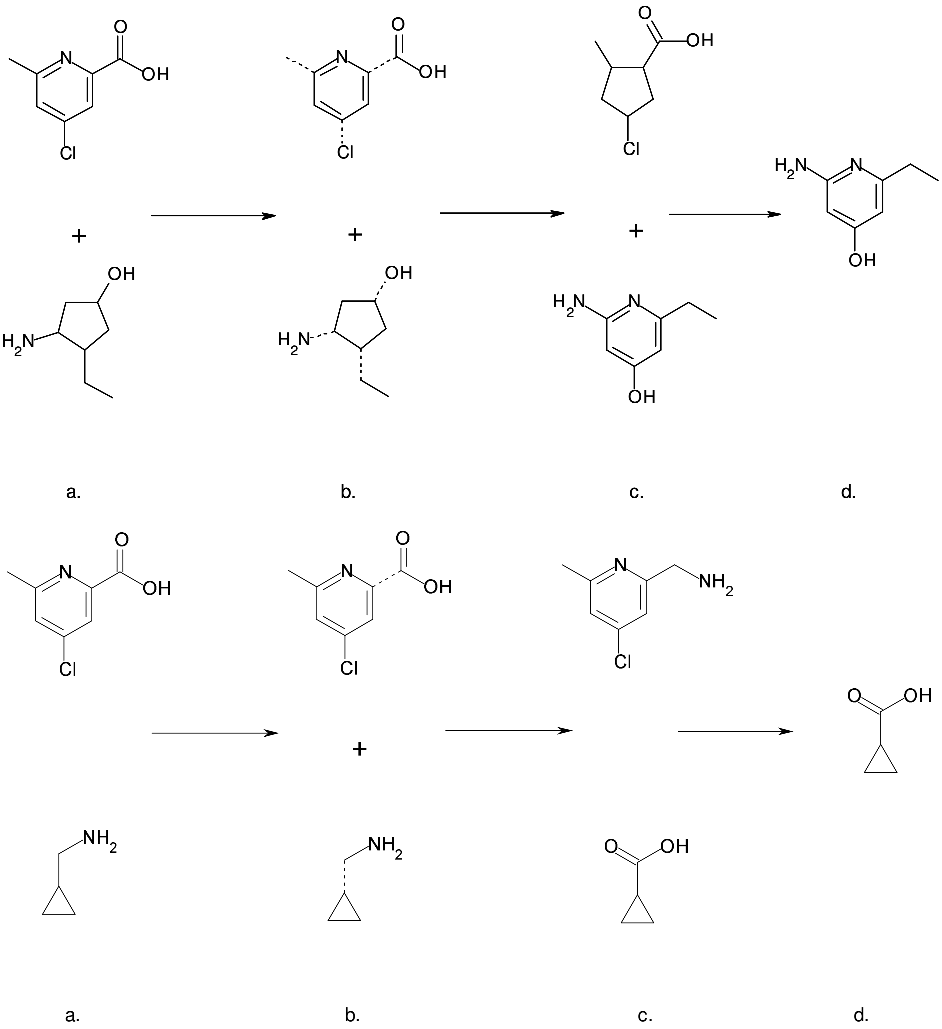


Figure 6.4: Fragment-based molecule crossover, which can both be performed on ring systems (top) and on branches (bottom). a) Two parent molecules are selected, b) the molecules are split: in both molecules a random fragment of the same type (branch or ring) is selected and the bonds between the chosen fragment and the rest of the molecule are broken c) the molecule parts are recombined, d) one molecule is selected at random as the crossover product. Note that the CH_2NH_2 group could also have been exchanged with the methyl or chlorine, the pyridine-cyclopropane interchange is however not possible since the ring systems have one and three groups attached, respectively.

simulations of HIV-RT-ligands (Titmuss, 1999). Therefore, protein mol2 files for GOLD docking were generated using the Biopolymer module in Sybyl (TRIPOS) by removing the ligand and crystallographic water molecules from the pdb-file, and adding all hydrogen atoms. Molecules were generated as 2D structures in MDL MOL-file format (Dalby, 1992) by the Molecule Evoluator and converted to 3D structures in mol2 format by the program MOE (Chemcomp. Corp.) GOLD docking was performed using "*7-8 times speed up*" settings (Jones, 1997). The active site centre as determined by the PASS program (Brady, 2000) was taken as the starting position of the GOLD flood fill algorithm. To meet aspects of calculation time and data size on one hand, and convergence criteria and statistical relevance on the other hand, 15 independent docking runs were performed for each docking case.

Estimating ease of synthesis

As last part of the experiment, we checked ease of synthesis. We asked an experienced medicinal chemist, J.B., to check of each structure whether it could be synthesized reasonably easily. If not, we asked him to suggest a suitable derivative. All suggested replacement structures were docked using the normal docking procedure, and their fitness values gathered and compared to the fitness scores of the original compounds.

Results and discussion

Docking the molecules into HIV-RT

Docking the EA-generated compounds in the HIV-RT crystal structure resulted in the dockings shown in Figure 5. The binding pocket can be divided into a large hydrophobic subpocket 1 (enclosed at the top by residues Y188, W229, F227), and two small subpockets 2 (formed by L234, P236, and Y318) and 3 (formed by V178 and Y181) (Hopkins, 2004), which is located close to a hydrophilic solvent channel, proposed to be the ligand access channel (Shen, 2003). Typical NNRTIs like HEPT form H-bonds with the backbone of K101 and/or K103, and in some cases also with a water molecule located in the solvent channel (Ren, 1995; Esnouf, 1997). Furthermore typical NNRTIs bind to subpocket 1 via hydrophobic and aromatic interactions (Titmuss, 1999) and can form additional hydrophobic and aromatic interactions with subpockets 2 (e.g., HEPT and efavirenz) and 3. The compound with the highest fitness in the first generation only occupied subpocket 1, while the best compounds found by atom-based evolution and fragment-based evolution occupied subpockets 2 and 3,

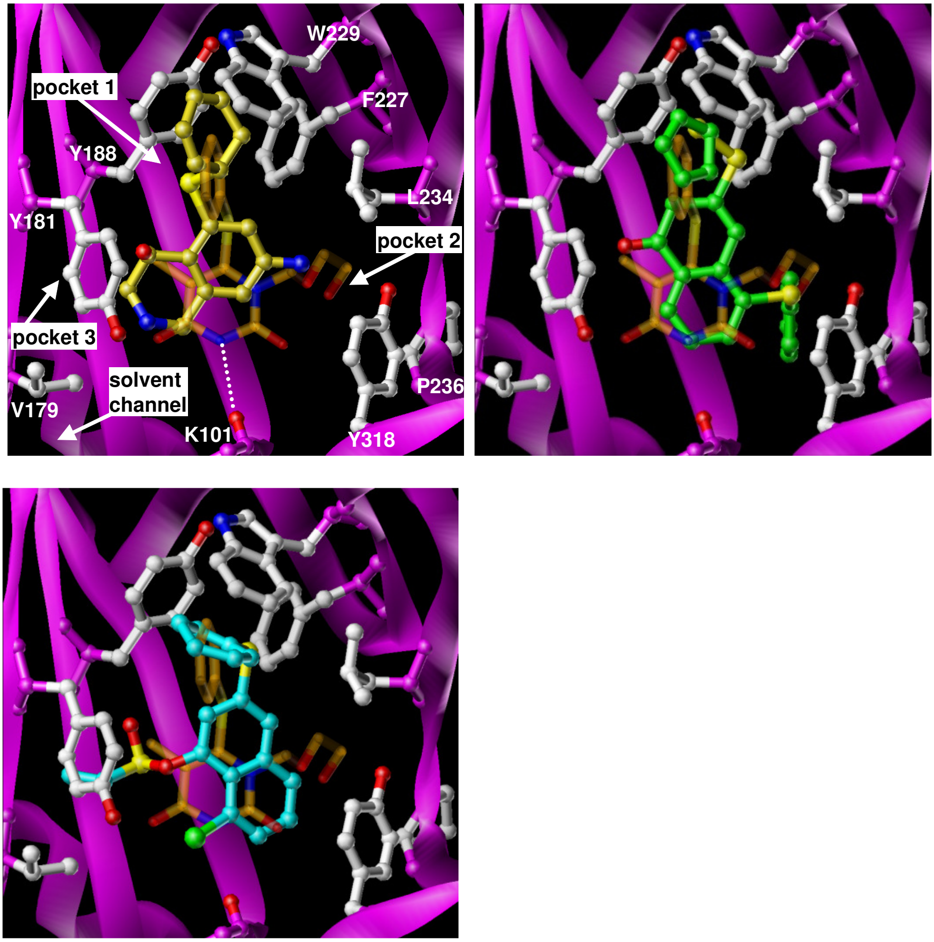


Figure 6.5. The dockings of the compounds generated by the evolutionary algorithm into the original crystal structure of HEPT bound to HIV-reverse transcriptase. HEPT is shown as a transparent brown structure. In addition, (a) contains the compound with the highest fitness in the first generation (yellow) docked into the crystal structure, (b) the docking pose of the best compound found by atom-based evolution (green), and (c) the docking pose of the best compound found by fragment-based evolution (cyan).

respectively, in addition to subpocket 1. None of the compounds formed H-bonds with the backbone of K101 and/or K103. However, hydrophobicity is another key feature driving the potency of inhibitor binding to the NNRTI site (while GoldScore does not have a separate term for hydrophobic interactions, it calculates van der Waals interactions and therefore assigns a higher score to a ligand that better fits the three-dimensional structure of the binding pocket). Optimization of hydrophobic interactions in pocket 1 is already observed in the first generation cycle (see the left panel of Figure 6.5), in which an S-phenyl group attached to a two-ring ring system dips into the aromatic cluster of Y188, W229, F227. A two-ring core, consisting of two fused 6-rings, is maintained throughout both atom-based and fragment-based evolution (see Figures 6.6 and 6.7), even though the original 1,2,3,4-tetrahydroisoquinoline ring system is changed during atom-based and fragment-based evolution into a 1,2-dihydronaphthalene ring system and a naphthalene ring system, respectively. Furthermore, the initial aromatic ring dipping into pocket 1 is only slightly changed into other functional groups of the same hydrophobic character along the different evolution pathways. The evolution has led to probing and optimization of hydrophobic interactions with different smaller subpockets of the HIV-RT binding site. Re-positioning of the pocket 1-occupying subsituent from the *5'* to the *6'* position accommodates interactions of hydrophobic substituents on other positions at the two-ring core with subpockets 2 and 3. In the atom-based evolution, the initial hydroxyl group sticking out in the direction of subpocket 2 (Figure 6.5a) is replaced by an S-phenyl group which can interact with subpocket 2 formed by L234, P236, and Y318 (Figure 6.5b). During fragment-based evolution, an ethenesulfinate group was attached *meta* to the hydrophobic group interacting with subpocket 1. This ethylenesulfinate group has just the appropriate size to dip into the small subpocket 3 (Hopkins, 2004).

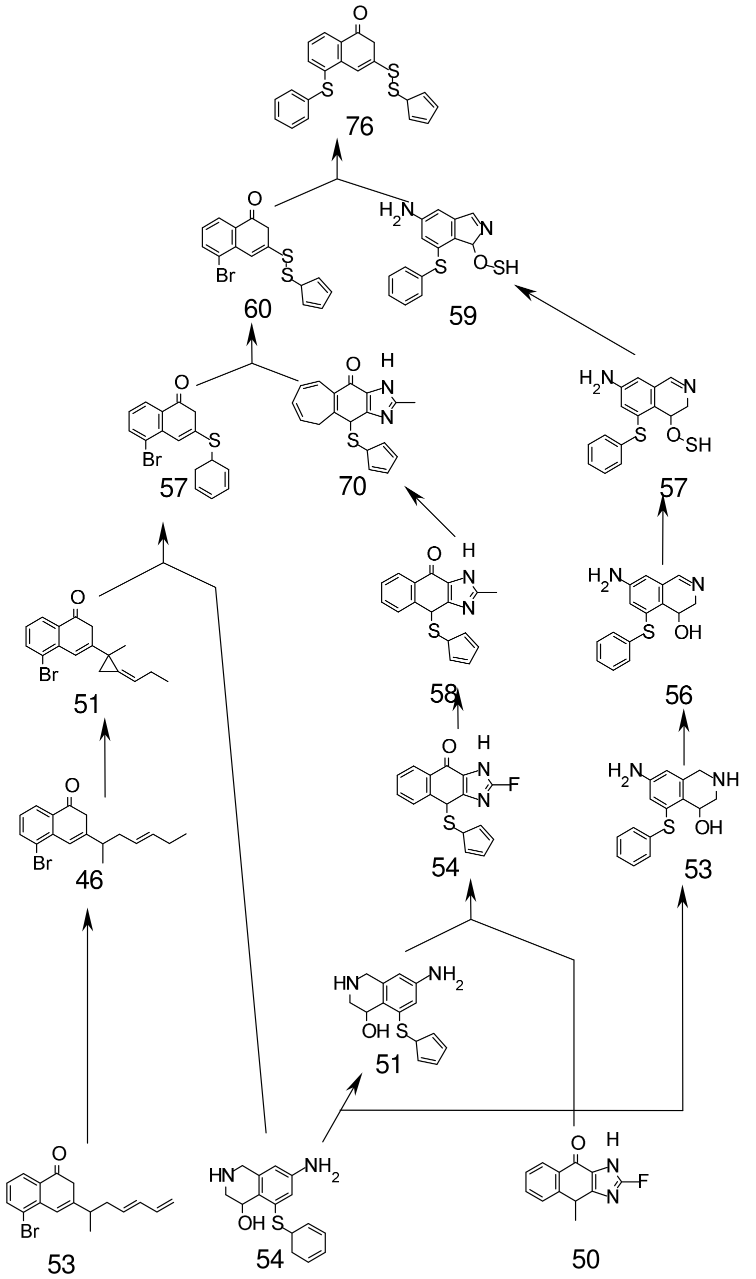


Figure 6.6. The optimization trajectory leading to the best molecule of the atom-based evolution. The numbers under the structures indicate their fitness score.

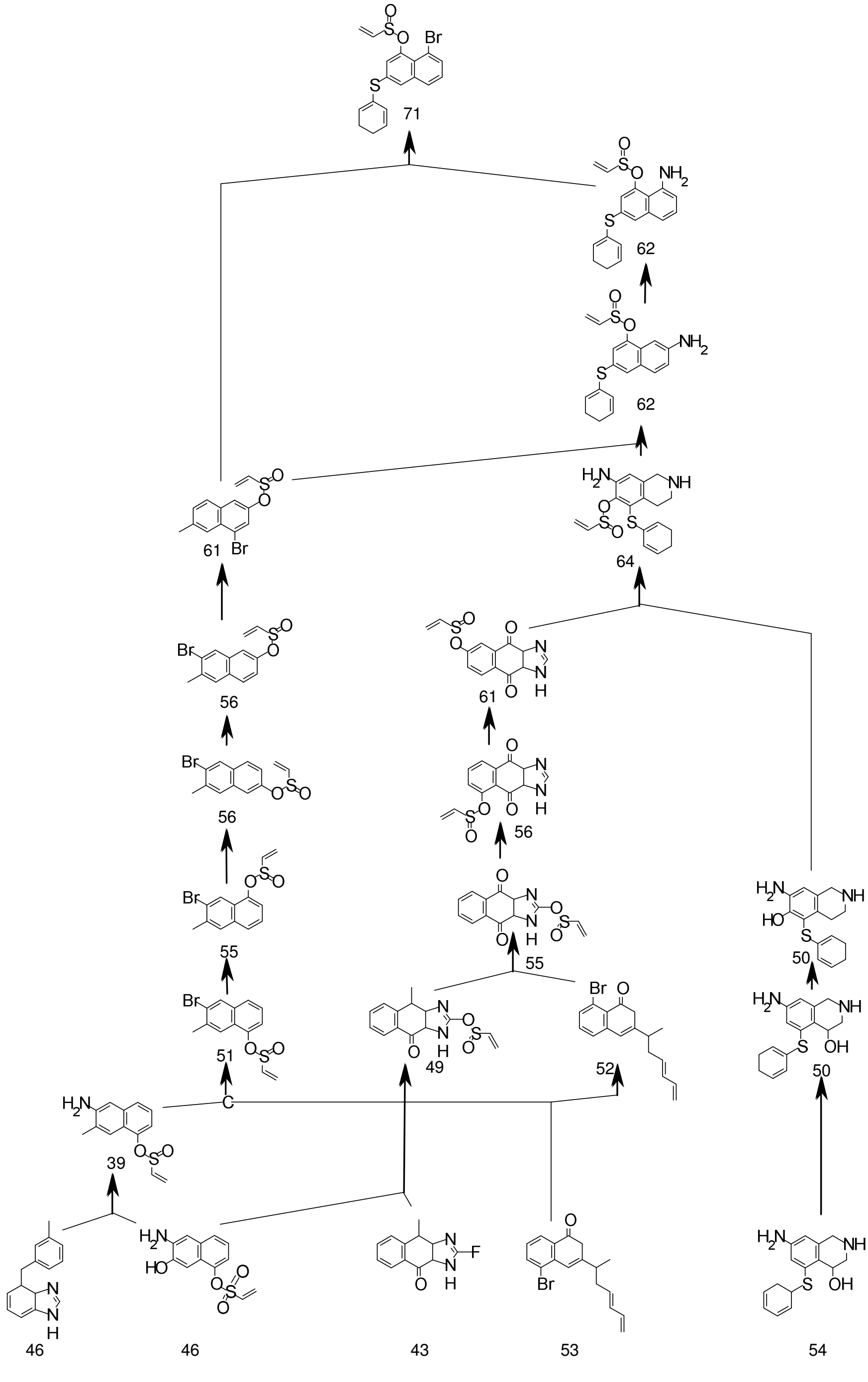

71
62
62
61
64
56
61
56
56
56
55
55
51
49
52
50
50
39
C
46
46
43
53
54

Figure 6.7: The optimization trajectory leading to the best molecule produced by the fragment-based evolution. The numbers under the structures indicate their fitness score. Note that the $SO_3C_2H_3$-group of the second parent molecule was modified to a $SO_2C_2H_3$-group by an error in our program. This bug was later fixed. In later fitness comparisons, not much influence on fitness was found by the presence or absence of the extra oxygen atom, so this glitch will probably not have influenced evolution much.

The change in fitness over the generations

To study the differences between the atom-based and fragment-based evolution, we first gathered of each generation the maximum fitness value (the fitness of the "best" molecule) and the average fitness value. These values are plotted in Figure 6.8.

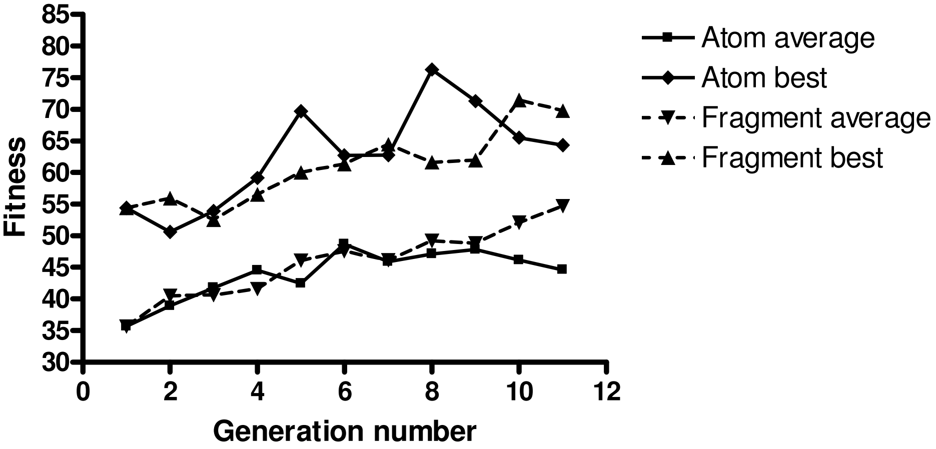


Figure 6.8: The average and the best fitness of molecules in each generation for the atom-based and fragment-based evolution.

Figure 6.8 shows that both the average fitness and the maximum fitness of the molecules in the population grow as the evolution proceeds. This means that new and better molecules are found, which implies that evolution improves upon pure selection (since pure selection would cause the average and maximum fitnesses of the later generations to approach the maximum of the first generation). However, virtual screening of a large library will also increase maximum fitness, as there is always a probability that a new molecule will improve upon the known compounds. We should therefore analyze our data further to be able to say whether evolution is truly more effective or efficient than random search.

Fitting the fitnesses of the randomly generated first generation to a Gaussian (Figure 6.9) results in a best fit with mean value 36.7 and a standard deviation of 4.3.

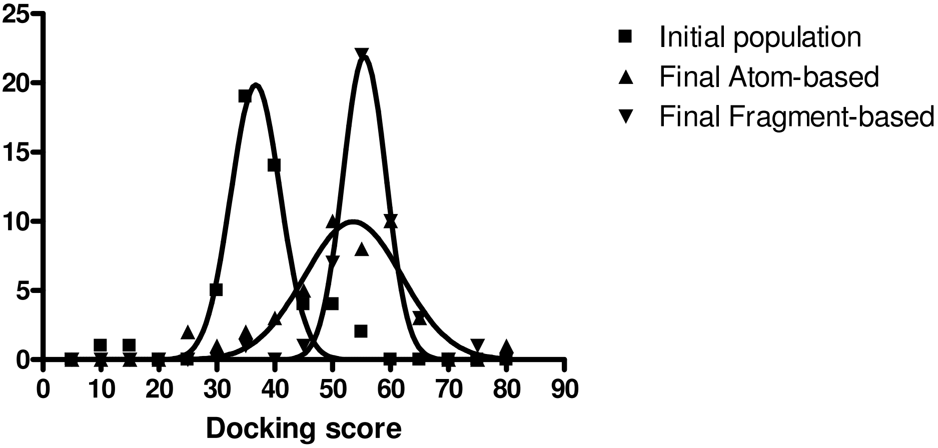


Figure 6.9: The distribution of fitness scores of the molecules of three populations fitted to Gaussians. The populations are the initial population, the atom-based population containing the highest-scoring atom-based molecule (8th atom-based generation), and the fragment-based population containing the highest-scoring fragment-based molecule (the 10th fragment-based generation).

The best overall result of evolution (score=76) lies about 9 standard deviations from this average. Therefore the probability that a random search would produce a molecule with this or a higher fitness would be smaller than 10^{-9}%. Scanning 500 molecules and getting the improvement shown in Figure 6.9 suggests that this improvement cannot be the result of a random search only. Therefore evolution seems truly more efficient than random search. We should note hereby that the distribution of the initial population, as shown in Figure 9, is Gaussian-like, but not exactly Gaussian. For example, there are about 5 molecules with a fitness score of about 50, which is more than would be expected from a true Gaussian distribution. Therefore, putting the odds that random search produces a similar improvement as the evolutionary algorithm at 1:10^{11} may be mathematically questionable. However, the distance between the best fitness found by the EAs and the average and best fitnesses of randomly produced molecules seems large enough to indicate that both of our evolutionary algorithms noticeably improve on random search.

Fitness peaks/plateaus

While evolution improves molecular structures significantly, the improvement (at least in atom-based evolution) stops quite quickly, around the 8th generation. It is very unlikely that by then the 'best possible' molecule has already been found. What then causes this stagnation? And how could it be prevented in future experiments?

The first explanation is that we simply have evaluated too few generations for full optimization. EAs are known to have periods of stasis, and are usually run over many more generations than 10: in other applications of evolutionary algorithms to *de novo* design molecules are evolved over 50 to 20,000 generations (Nachbar, 1998; Vinkers, 2003; Brown, 2004). Often, lack of improvement in fitness only means that the population is at some local optimum, and needs to be evolved for a few more generations before better molecules are found. Running the EA for more generations (something we did not do due to limitations on computational time and our own time – a number of steps such as screening for unusual structures had not yet been automated and required manual intervention) may result in renewed improvement with better molecules found.

Our second hypothesis is that the stagnation and deterioration of scores has uncovered an inefficiency in the EA that should be addressed instead of compensated through evaluating more generations. The most likely explanation of the decrease is that our rule that each molecule generated must be "new" forces the evolution away too quickly from a local optimum; better variants of the best molecule cannot be built anymore since the best molecule has already been "forgotten". The result is a downhill trend in the fitness scores, the molecular structures going away from the last found optimum, until a novel structure is found which can be optimized again. In our case, it may be that the requirement that each molecule must be new may prevent a true local optimum to be reached since eight generations is too short for full optimization; of any compound, including the best compound, several hundreds of derivatives can be made through atom-based mutation. Therefore, the one or two generations the best molecule had to procreate are not enough for a good "full" local optimization. Therefore, it would be best to drop the "only new molecules" requirement. Interestingly, the fragment-based evolution does not show a similar deterioration. It is possible that the fragment-evolution has not reached a similar plateau yet, since it reached its peak later than the atom-based evolution.

Comparing the molecules created by atom-based and fragment-based evolution

Next to the fitness scores, we compared the molecules resulting from atom-based and fragment-based evolution. The top 5 molecules found by both approaches are shown in Figure 6.10, together with their fitness scores.

The most interesting observation to us was that fragment-based evolution seems approximately as good as atom-based evolution. This seems to refute our hypothesis that atom-based evolution should go more smoothly and therefore be better than fragment-based evolution. Why were we wrong? To find an explanation, we studied the different optimization trajectories and the molecules produced in more detail. The evolution trajectories leading to the best molecules found by the atom-based and the fragment-based evolution, respectively, are shown in Figures 6.6 and 6.7.

Though there are obviously differences between the optimization trajectories, the atom-based evolution and the fragment-based evolution are quite similar in terms of 1) the part of the initial population that is used, 2) the presence of local optima, and 3) the high occurrence of crossover. Each of these similarities will be discussed shortly below.

To produce their best molecules, both atom-based and fragment-based evolution apparently use only a small part of the initial population (3 to 5 molecules), and no *de novo* molecules which have been later added appear in the trajectories of Figures 6.6 and 6.7. This suggests that the 4-size tournament selection is quite strict, and quickly fills the population with high-scoring mutants of the best molecules, giving “newcomers” little chance to contribute to the gene pool. It may be worthwhile to experiment with less strict selection (such as 3-sized tournaments or 2-sized tournaments) to see if this allows more of the information in the population to be used in developing more diverse and eventually even better molecules.

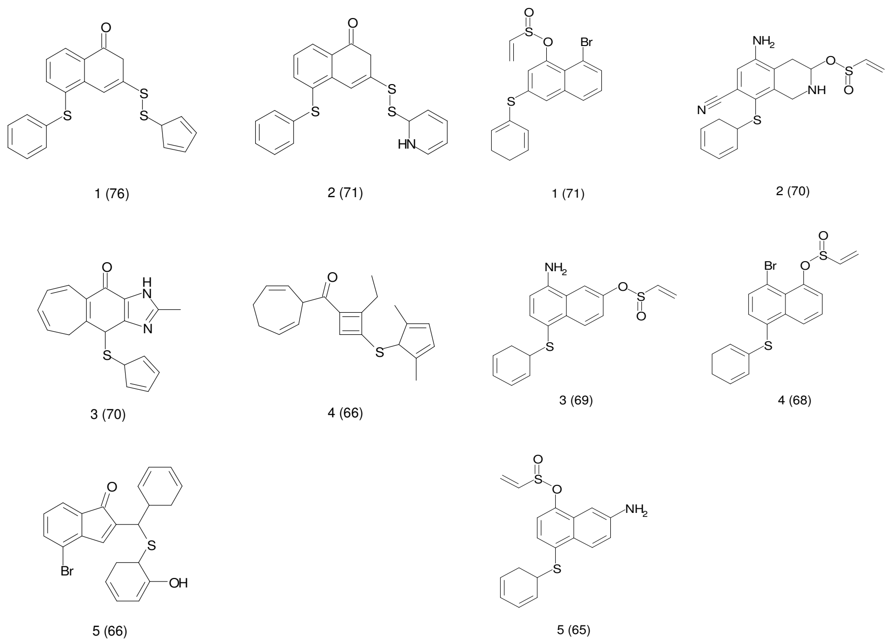


Figure 6.10: The most highly scoring five molecules found by atom-based evolution (left) and fragment-based evolution (right). The numbers in the brackets indicate the docking score.

Another reason for not making the selection pressure too high is the presence of local optima. It is quite common for a high-scoring molecule (like the bottom-left molecule in the atom-based evolution, score 53) to produce a lower-scoring molecule (score 46 in this case) which however gives birth to new molecules that have a fitness comparable to or even higher than that of its (great)grandparent (52 and 57, respectively). One may argue that this could be caused by imprecision in the docking procedure, which may assign somewhat different scores to one molecule if the docking experiment were to be repeated. In such a case the intermediate molecule may indeed have an intermediate score if more calculations would be performed. Alternatively, there could really be local maxima in an optimization trajectory, which would trap a greedy (“only better offspring”) algorithm too easily. Methods like evolutionary algorithms (used here) or simulated annealing, which generally explore around local optima, may therefore be inherently more suitable for molecular space than a greedy search which discards all variants with a lower score than the highest scorer. Another reason not to select too greedily is that both computational fitness functions and experimental affinity measures are bound to have some noise, with some superior molecules perhaps scoring slightly worse than their competition due to the random nature of the docking process and the random variations in experimental measurements. An evolutionary algorithm, to be effective, should therefore tolerate this noise and not select too strictly and too greedily. We also stress that the accuracy of the docking score is not too relevant for the comparisons made here. Our aim was not to find novel HIV-RT inhibitors, but to evaluate the two EAs in a realistic setting. Docking procedures possibly assigning different scores over different runs to the same molecule can even be considered to incorporate an extra element of realism since biological experiments also tend to yield ranges of values, and a good optimization algorithm should work despite these irregularities.

The third similarity between the different evolution methods is the relatively high occurrence of crossover. While our settings should have led to about 15% crossover versus 85% mutation for modifying molecules, the optimization tree of highest scoring compound in the atom-based evolution had 4 crossovers versus 9 mutations, 31%, while that of the highest-scoring compound in fragment-based evolution had 7 crossovers versus 10 mutations, 41%. The reason the crossover was set at 15% in the first place was because medicinal chemists evaluating the Molecule Evoluator did not like the large changes in molecules introduced by crossover; the structure of crossover products was often so different from that of its parents that the compounds created by crossover seemed unlikely to have any related biological activity. Apparently, however, crossover does not always make such activity-destroying changes, and is a useful operator for finding good molecules. Therefore, we should consider setting its prevalence higher in follow-up experiments. Interestingly, molecule crossover has also been found useful to create drug molecules in practice (Lazar, 2004).

Surveying our results, we learned that atom-based and fragment-based evolution have lots in common and that both seem approximately equally efficient in this experiment. However, there are also differences. First of all, the evolution trees show more clearly than the fitness plot that atom-based evolution does run more smoothly. It reaches a better molecule than the fragment-based evolution, and it does so in fewer generations. It truly seems to be more efficient, though not by a large amount. In the fragment-based evolution shown in Figure 6.7, big substituents make large jumps around the molecular core. An example would be the $SO_2C_2H_3$ group in the middle column of Figure 6.7, which moves to very different positions of the ring system. If a large substituent is attached at very different positions in parent and offspring molecules, this would imply that either the substituent of the offspring molecule is docked in another part of the binding pocket than it was in the parent molecule or that the offspring's core is docked in another place. Since the dockings of parents and offspring in fragment-based evolution are therefore more likely to be dissimilar, the evolution is probably less efficient. If, for example, in fragment-based evolution only half of the offspring of a good molecule is docked in the way of their parent, it is likely that only this half will have a good or even better score than the parent molecule, and the time spent to evaluate the other half of the offspring is wasted. The smaller changes in atom-based evolution should result in offspring of which a large part will be docked similarly to the parent, say 90%, leading to fewer offspring with low fitness scores due to radically different docking. Quantitatively, one would have to analyze the correlation between the fitness scores of parent and offspring. It would definitely be interesting in future investigations to automate such a "docking comparison" process and get the actual numbers.

The second difference between the atom-based and fragment-based approaches is the result of the optimization. While the molecules found by the atom- and the fragment-based evolution have similar fitnesses, their structures are definitely dissimilar (Figure 6.10). The most striking difference between the best molecules found by the atom-based evolution and those found by the fragment-based evolution is the lack of diversity in the top-5 fragment-based molecules. These molecules all have a sulfur-bound cyclohexadiene group and a ethylenesulfinate group attached to naphthalene or a naphthalene-like ring system. In contrast, the top-5 of atom-based molecules contains four different structural classes. This diversity is beneficial, since it improves the chance that at least one class of molecules is found with suitable biological and physicochemical properties.

The difference in diversity between the molecules optimized by atom- and fragment-based evolution led us to investigate the cause of the similarity of the top-5 molecules created by fragment-based evolution. Was the fragment-based population highly diverse and did it contain several different structural groups, of which one was superior, or were *all* molecules in generation 10 similar, and hence the best molecules too? We measured population diversity by counting the number of different ring systems in each generation of the atom- and the fragment-based evolution. Figure 6.11 plots generation number versus ring system diversity.

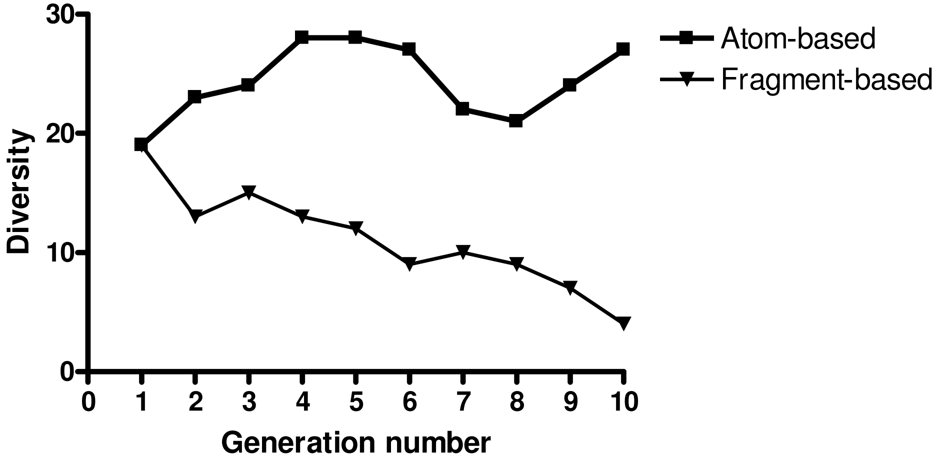


Figure 6.11. The number of different ring systems in the population during the atom-based and the fragment-based evolution. In the last generations, the fragment-based population only contains cyclohexadiene, naphtalene and the 1,2,3,4-tetrahydroisoquinoline system from the best compound n the first generation, while the atom-based population has 3-, 4-, 5-, 6- and 7-rings with a variable number of double bonds, and a number of two-ring systems (5:6, 6:6) and three-ring systems.

The diversity plot shows that while the diversity of the population undergoing atom-based evolution fluctuates, the diversity of the population undergoing fragment-based evolution dwindles fast. This difference probably arises because the atom-based evolution has several options to generate new ring systems: oxidizing/reducing bonds, breaking rings and making rings, as well as mutating, inserting and uninserting atoms. In contrast, the fragment-based evolution currently implemented in the Molecule Evoluator cannot introduce new fragments, only recombine and copy existing fragments. Therefore, it entirely depends for its new ring systems on the 5 *de novo* molecules that are inserted at every generation, and we have seen that these molecules rarely survive in a population of already optimized molecules due to the high selection pressure. Fragments which are less competitive in the current generation can thus go quickly extinct, with no chance of reappearing. The resulting lack in diversity bodes ill for further optimization. While the atom-based evolution may crawl out of its crisis into a new optimum, the fragment-based evolution seems doomed to stagnate in a few more generations.

It is interesting to compare the change in fitness over the generations to the change in diversity over the generations. It turns out that the minimum of the atom-based diversity (generation 8) coincides with the occurrence of the best compound found in the run. Apparently, before generation 8, the evolution is on the trail of a good compound and concentrates the search in that direction, diminishing structural diversity to focus on a few good groups. Afterwards, the obligatory removal of the best molecules pushes the evolution away from the local optimum, and thus back into the high diversity that is necessary for efficient exploration.

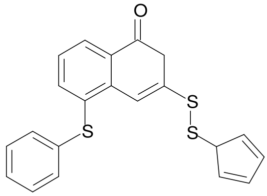
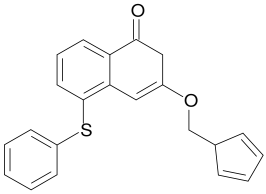
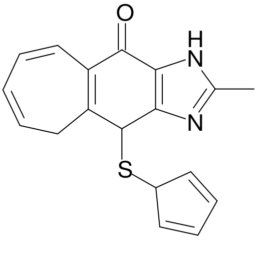
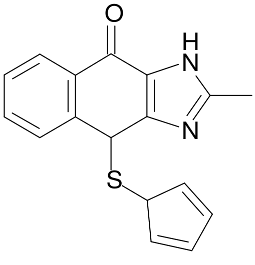
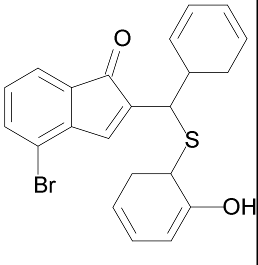
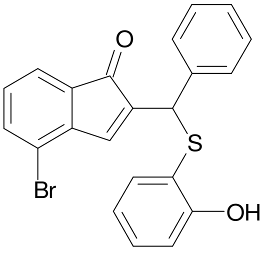
Synthetic feasibility

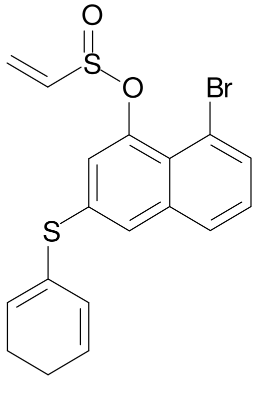
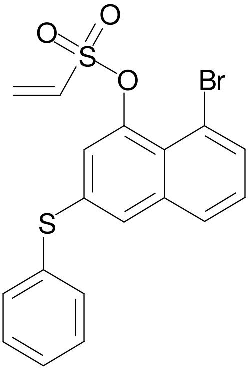
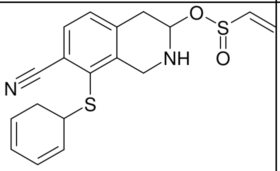
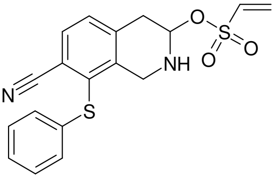
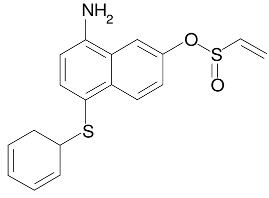
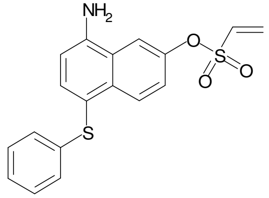
So far we have seen that atom-based evolution seems slightly better than fragment-based evolution regarding the speed of evolution and the fitness of the molecules obtained. However, this does not address the argument of some investigators that fragment-based evolution should be used because the molecules created by it are easier to synthesize. In reality, the difference in ease of synthesis may be not as important as it seems, since both atom- and fragment-based molecules often need to be adapted by experienced medicinal chemists to create molecules which can be synthesized within reasonable time (Douguet, 2000; Vinkers, 2003). Fragment-based molecules might need fewer adaptations than atom-based molecules, but the importance of this difference in ease of synthesis has, to our knowledge, not been investigated yet.

To assess the effects of ease-of-synthesis requirements on the molecules reached by atom-based and fragment-based evolution, we submitted a selection of the top-10 atom-based molecules and top-10 fragment-based molecules to a chemist, who suggested improvements to enhance ease of synthesis. The resulting modified molecules were docked in HIV-transcriptase to determine their fitness values. The results of six of the compounds, selected for diversity in structure, are shown in Table 6.2.

From comparing the structures and scores of the molecules in Table 6.2 we conclude that fragment-based evolution indeed produces superior results respecting ease of synthesis. This is mainly because fragment-based molecules need fewer modifications, which leads to a smaller change in structure, which leads to a smaller change in activity, and since changing an optimized structure is likely to decrease activity, the smaller adaptation needed for the fragment-based molecules is beneficial. From our results, it seems that the position of the modifications also plays a role; if modifications are in or near the core of the molecule, the change in activity is much more dramatic than when a substituent at the “edge” is changed. For example, only modifying the central ring of the 4th best atom-based compound (Figure 6.10) from cyclobutadiene to phenyl made the docking score fall from 66 to 9. In contrast, the decrease in fitness in the fragment-based molecules is generally much smaller, about 10 units, probably because only the substituents are changed.

Table 6.2: Some of the molecules designed by the evolutionary algorithm, their derivatives with improved ease of synthesis suggested by our chemist, and the docking scores of the original and derived molecules.

	Original molecule	Derived molecule	Fitness of original molecule	Fitness of derived molecule
Atom-based evolution			76	39
			70	57
			66	54

Fragment-based evolution			71	63
			70	50
			69	68

Comparison with literature

As last part of this discussion, we compare our work with literature and investigate whether our results are consistent with the findings of others, and whether literature together with our own results can suggest directions for further investigations.

Atom-based evolution. To our knowledge, the earliest publication that described computational evolution of molecules was by Glen and Payne (Payne, 1995). Their mutations and crossover were more or less the same as our atom-based operators, though they included a 2-point crossover (which exchanged the central parts of two molecules) and limited insertions to methylene (only carbon-atoms could be inserted). Douguet et al. (Douguet, 2000) also included the 2-point crossover, but did not use a ring breaking operator, and seemed to focus much on changing larger parts of the molecule at once, which resulted in a hybrid fragment-atom evolution. Globus et al. (Globus, 1999) used only one-point crossover, and no mutations. According to the authors, their algorithm worked quite well, even though it was sometimes impossible to reach a certain target structure because the starting population or an intermediate population did not contain all necessary structural elements (the same can be said of our current fragment-based evolution). The crossover of Nachbar (Nachbar, 1998) also was one-point, and his mutations were similar to our own atom-based mutations (insertion, deletion, atom type mutation, oxydation and reduction of bonds, etc.) However, there were more restrictions on atom change mutations, as insertion and uninsertion were apparently only allowed in rings, and rings could only be broken at one predefined ring bond (though this latter shortcoming was corrected in his later work (Nachbar, 2000)). Brown et al. (Brown, 2004) used mutations similar to ours, though the two crossover operators were quite elaborate and differed considerably from our simple one-point crossover. Finally, the atom-based mutations of Nicolaou et al. (Nicolaou, 2009) also involved modifying atom and bond types and insertion and removal of atoms. However, it seems that cycles were neither formed nor broken in this investigation, possibly because the atom-based mutations were supplemented with fragment-based mutations which could simply insert entire ring systems. Nicolaou's atom-based crossover is similar to that of Globus (and ours), picking one bond to break in two molecules and pasting one of the parts of the first molecule to one of the fragments of the second molecule. The general impression left by comparing the work of others to our atom-based mutation and crossover operators seems to be that our set of atom-based mutations is essentially complete, and at least as versatile as in other work. However, of crossover many variants seem possible, and it should be investigated which types of crossover are most beneficial for *de novo* design, especially considering the large contribution crossover made in our investigation.

Fragment-based evolution. The fragment-based evolutionary approaches we found in literature are those of Schneider et al. (Schneider, 2000), Pegg et al. (Pegg, 2001), Vinkers et al. (Vinkers, 2003) and Dey et al. (Dey, 2008). The evolutionary algorithm of Nicolaou et al. (Nicolaou, 2009) used both atom-based and fragment-based evolution, their atom-based operators were discussed in the preceding paragraph, while their fragment-based operators will be discussed here. Schneider obtained his fragments by using retrosynthetic rules on the WDI database (Daylight, 2005), resulting in over 24,000 fragments. From these fragments, molecules were constructed by applying the same retrosynthetic rules in the reverse direction. The algorithm did not contain crossover, just mutation which replaced a fragment by another fragment of the same type. Dey et al. (Dey, 2008) used a commercially available database of 20,000 pre-selected fragments. Interestingly, the genotypes in their algorithm did not encode for a complete molecule, each gene encoded a docked pose of an unattached fragment; every member of their population was therefore not a molecule, but a collection of unconnected fragments, which were connected later by another optimization algorithm. Mutation and crossover involved changing the value of a gene (making it represent another docked fragment) or exchanging the values of the genes at a certain position between two members of the population. Nicolaou et al. used a smaller database, consisting of 2363 fragments of tested ligands (so somewhat focused on the problem at hand, not a truly general search through molecular space). Fragments could be inserted, removed or exchanged, according to the retrosynthetic rules as defined by RECAP (Lewell, 1998). In stark contrast to Schneider and Dey, Pegg et al. used only a handful of fragments (about 15 in one experiment), which could be attached randomly to each other. The mutations here were replacing fragments and connecting fragments to other points, which is similar to our "rotation". Finally, Vinkers et al. used 32,000 molecules and a collection of 70 reactions which could extend and combine molecules. In the first iterations, molecules could only "grow" by reacting with other compounds, in the later phase of evolution, reactants in any position of the chain could be replaced by other ones from the database.

From the diversity in the fragment-based studies one can conclude that many different approaches to fragment-based evolution are possible, with a varying role of synthesis rules in molecule construction, and fragment databases that vary from under 20 fragments to over 30,000 fragments. One of the few similarities between the fragment-based methods is that they can replace fragments in a molecule with fragments from the database, something our fragment-based evolution cannot do yet, but may be good to implement. Our fragment-based evolution does have an extra feature, however, as it considers each ring atom a potential attachment point; therefore exchange and rotation can create many more combinations from the same fragments than the more synthetically strict methods can. Not being able to get new fragments into the population is a disadvantage for optimization, but is partially compensated in our algorithm by this ability to create more combinations out of a fixed number of fragments. The flexibility in attachment points might reduce the synthetic feasibility of the designs, but could increase the speed and efficiency of the optimization, since the structural changes may be smaller than those of fragment replacement.

Next to comparing the mutation and crossover operators, it may be useful to compare the parameters of evolution of our investigation with those in literature. In the above-mentioned investigations, population sizes ranged from 30 to 1000, and the number of generations allowed ranged from 50 to 1000. Settings were varied to some extent; Douguet et al. tested population sizes from 10 to 80, and found 20 to 40 to be an optimal size. Glen and Payne investigated sizes from 10 to 100, and considered 50 to 100 to be the best size. In general, too small sizes are thought to result in premature convergence, too large sizes produce slower running optimizations, and may "drown" the best compounds by making the probability that they are selected for procreation too small. However, premature convergence can also be caused by high selection pressure, and this is likely to be the case with our atom-based evolution.

Another issue is whether we should or should not use elitism to conserve our best molecules. Both approaches are present in literature: occasionally (Pegg, Vinkers, Dey) elitism has been used, but in other cases (Glen, Nachbar) not. Douguet tested whether elitism was advantageous, and found that it is. Douguet's findings on elitism support our impression that it would be good to allow molecules be copied unmodified into the next generation, in a more reliable way and perhaps a larger scale than is used now.

A final point is that we have used far fewer generations than the other researchers. This was mostly because we wanted to optimize the efficiency of the evolution itself, and expose poor methodology which would have been masked by throwing in huge computational resources. We can only guess what would have happened if evolution had been allowed to go on for more generations. In the current setup, the fragment-based evolution would have converged in a few more generations, the atom-based evolution would have intermittently found new structures with activities comparable to or even better than our current maximum. However, in drug discovery it would have probably been more efficient to use 50 independent populations of 20 runs than it would have been to evaluate one population during 1000 runs. Having many diverse ideas is likely to be better than spending many generations refining a local maximum, which, despite advanced docking scores, would only have a small chance of being a suitable and active lead (Vinkers, 2003). Another argument for having many short runs instead of a few long ones is that using atom-based mutations on the fragment-based initial population tends to guide the evolution into generating increasingly uncommon groups, so ease of synthesis will probably decrease as molecules are "optimized" by the evolutionary algorithm.

One important point to consider is whether our test system, docking and its score, are suitable as fitness function. For example, Verdonk et al. (Verdonk, 2004) have shown that docking scores tend to increase approximately linearly with the number of heavy atoms in a compound. Chemically, it is reasonable that the average compound with 20 heavy atoms should bind more tightly than the average compound with 10 heavy atoms for the simple reason that having more atoms allows a compound to have more interactions with the target. However, we should be wary if the fitness scores of individual compounds correlate linearly with the number of heavy atoms. After all, this would mean that the fitness score is biologically unrealistic, since the specific types of atoms and the three-dimensional shape of the molecule should influence ligand binding much more than the atom count. We therefore took all of the compounds generated by the evolution and plotted fitness against heavy atom count. For the atom-based evolution, the plot is shown in Figure 6.12. For both atom and fragment-based evolution, the r^2 values of the correlation were less than 0.35, implying that most of the docking score could not be explained by heavy atom count. Our findings imply that the GOLD scoring function is in line with biochemical expectations: bigger molecules in general bind better, but atom count is not the only contribution to binding.

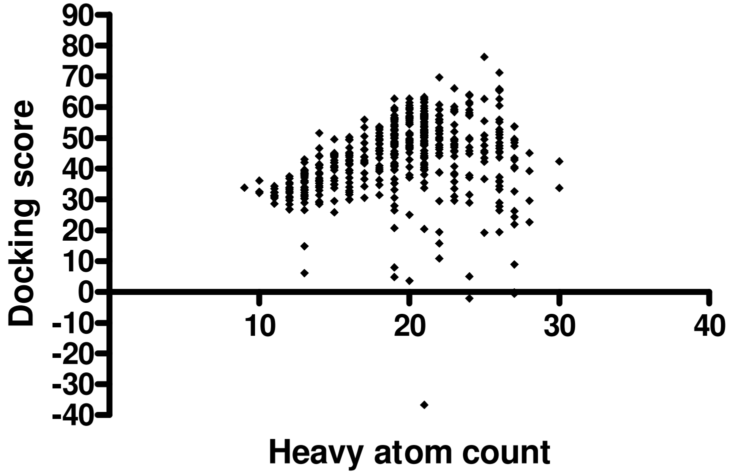


Figure 6.12: The relationship between heavy atom count and docking score for the molecules produced by the atom-based evolution; while a higher heavy atom count allows a higher score, it does not automatically lead to improved fitness.

Future perspectives

One of the main practical obstacles during this investigation was that it required quite some manual intervention to convert the 2D structures generated by the Molecule Evoluator to 3D structures, dock them, and feed the results back into the evolutionary algorithm. While the small scale (about 1000 molecules in total) allowed us to thoroughly analyse the strengths and weaknesses of the competing algorithms and give us some ideas of what we should or could change and improve, it would be impractical to scale our current method up for proper statistical analysis on dozens of runs. For future investigations, it would be certainly worth the time and effort to either create intermediate software or batch-files to automate as many of the intermediate steps as possible.

Next to automating the evolution-docking cycle, it would also be beneficial to 'trace' the evolution of compounds automatically, for example creating a sort of extended file format identifying the parent(s) of each compound and which mutation gave rise to it. This would be similar to the approach Nicolaou et al. (Nicolaou, 2009) use for their graph-based chromosomes, even though it is unclear from their paper whether they indeed used that data to optimize their evolutionary algorithm. In this way, we could automatically collect data detailing the average change in fitness after specific mutations, and let the computer quickly create all evolutionary trees, which could be helpful in analyzing exactly when and why a particular run stagnates or improves.

In addition, there are some factors which may hamper the optimization. One of those is the conversion of the molecules from 2D to 3D. The encoding of molecules in the current version of the Molecule Evoluator does not specify stereochemistry. This means that the conversion of the molecule “genome” to 2D will give rather randomly a *cis* or *trans* stereoisomer for each double bond outside rings, and the subsequent 2D to 3D conversion with MOE will add undefined R/S stereocenters. The Molecule Evoluator and (as far as we could observe) MOE are deterministic in their conversion procedures, which would ensure that a molecule genome would always be converted into one specific stereoisomer, so this will not affect reproducibility. However, offspring, even if modified outside the stereocenters, might have different stereochemistry assigned to them. Such “non-genetic” jumps in fitness would add noise and hamper the optimization. It would therefore be useful to encode stereochemistry in future versions of the Molecule Evoluator. Another concern is that some structures might be in a different tautomeric state in a biological environment (an enol might be a keto group in solution). Since bonds are coded into the genes, the tautomer remains fixed over generations and does not give rise to offspring in a different configuration, in contrast to enantiomerism. Therefore tautomerism will disturb evolution less than enantiomerism. However, a chemist should inspect the molecules with the highest scores for possible tautomerism, and dock the tautomers to explore the effects of the equilibria on the fitness score.

Summary and conclusion

In this investigation we compared the relative usefulness of atom based (small step) evolution and fragment based (large step) evolution for optimizing potential drug molecules. As was expected, atom-based evolution found a better molecule after fewer steps (although the difference was not large), while the fragment-based evolution produced molecules that seemed easier to synthesize. By comparing the two approaches, also a number of other lessons were learned concerning molecule evolution and how to improve the Molecule Evoluator.

Firstly, there is no need to add entirely novel molecules after the first generation. The fitness of the population increases so steeply, that the new molecules do not contribute any useful “genes”.

Secondly, and somewhat surprisingly, crossover can be a valuable operator in molecule optimization; in contrast to “normal” optimization strategies in pharmaceutical research, in which variations are made on one molecule, it may be worthwhile to combine one molecular core and some of the side groups of other high-scoring molecules.

Future modifications to improve the Molecule Evoluator would be to stop adding *de novo* molecules each generation, experiment whether allowing a molecule to appear more than once in a generation would find an even better optimum, and implement automatic gathering of statistical data of all evolved molecules to analyze the contributions of the various possible mutations or crossovers to molecule fitness. Finally, new mutation operators such as “add/delete fragment” will have to be added to the fragment-based evolution to prevent excessive loss of diversity and premature convergence.

Optimizing molecule optimization techniques is still complicated and requires lots of experiments and sufficiently sophisticated fitness functions. However, if drug designers want to explore a greater part of the huge number of potential drug-like molecules (over 10^{60}) or want to use accurate and therefore slow fitness functions for a more reliable *in silico* screening, efficiency in optimization is needed. We hope that studies like this one will not only lead to faster optimization methods, but will also give insight into the general craft of molecule optimization.

Acknowledgement

We thank Johannes Brussee for his help in assessing the synthetic feasibility of the best-scoring compounds, and for suggesting modifications for greater ease of synthesis.

References

(Barreiro, 2007) Barreiro, G.; Guimarães, C. R. W.; Tubert-Brohman, I.; Lyons, T. M.; Tirado-Rives, J.; Jorgensen, W. L. Search for Non-Nucleoside Inhibitors of HIV-1 Reverse Transcriptase Using Chemical Similarity, Molecular Docking, and MM-GB/SA Scoring. *J. Chem. Inf. Model.* **2007**, *47*, 2416-2428.

(Beilstein, 2009) http://www.info.crossfirebeilstein.com/features.shtml.

(Bohacek, 1996) Bohacek R. S.; McMartin C.; Guida, W.C. The Art and Practice of Structure-Based Drug Design: A Molecular Modeling Perspective. *Med. Res. Rev.* **1996**, *16*, 3-50.

(Brady, 2000) Brady, G. P.; Stouten, P. F. W. Fast prediction and visualization of protein binding pockets with PASS. *J. Comput.-Aided Mol. Des.* **2000**, *14*, 383-401.

(Brown, 2004) Brown, N.; McKay, B.; Gilardoni, F.; Gasteiger, J. A Graph-Based Genetic Algorithm and Its Application to the Multiobjective Evolution of Median Molecules. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 1079-1087.

(Carter, 2005) Carter, E.; Ebdon, S.; Neal-Sturgess, C. Optimization of Passenger Car Design for the Mitigation of Pedestrian Head Injury Using a Genetic Algorithm. In *Proceedings of GECCO 2005*; Beyer, H.G. et al., Ed.; ACM: New York, 2005; Vol 2, pp 2113-2120.

(CAS, 2009) http://www.cas.org/expertise/cascontent/ataglance/index.html.

(Dalby, 1992) Dalby, A.; Nourse, J.G.; Hounshell, W.D.; Gushurst, A.K.I.; Grier, D.L.; Leland, B.A.; Laufer, J. Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 244-255.

(Daylight, 2006) World Drug Index, published by Daylight Chemical Information Systems Inc. http://www.daylight.com/products/wdi.html.

(Dey, 2008) Day, F.; Caflisch, A. Fragment-Based de Novo Ligand Design by Multiobjective Evolutionary Optimization. *J. Chem. Inf. Model.* **2008**, *48*, 679-690.

(Douguet, 2000) Douguet, D.; Thoreau, E.; Grassy, G. A genetic algorithm for the automated generation of small organic molecules: Drug design using an evolutionary algorithm. *J. Comput.-Aided Mol. Des.* **2000**, *14*, 449-466.

(Ertl, 2000) Ertl, P.; Rohde, B.; Selzer, P. Fast Calculation of Molecular Polar Surface Area as a Sum of Fragment-Based Contributions and Its Application to the Prediction of Drug Transport Properties. *J. Med. Chem.* **2000**, *43*, 3714-3717.

(Esnouf, 1997) Esnouf, R. M.; Ren, J.; Hopkins, A. L.; Ross, C. K.; Jones, E. Y.; Stammers, D. K.; Stuart, D. I. Unique features in the structure of the complex between HIV-1 reverse transcriptase and the bis(heteroaryl)piperazine (BHAP) U-90152 explain resistance mutations for this nonnucleoside inhibitor. *Proc. Natl. Acad. Sci. USA* **1997**, *94*, 3984-3989.

(Gillet, 1999) Gillet, V. J.; Willett, P.; Bradshaw, J.; and Green, D.V.S. Selecting Combinatorial Libraries to Optimize Diversity and Physical Properties. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 169-177.

(Gillet, 2000) Gillet, V. J. *De Novo* Molecular Design. In *Evolutionary Algorithms in Molecular Design*, Clark D.E., Ed.; John Wiley & Sons, Inc.: New York, 2000; Chapter 4, pp 49-69.

(Globus, 1999) Globus, A.; Lawton, J.; Wipke, T. Automatic molecular design using evolutionary techniques. *Nanotech.* **1999**, *10*, 290-299.

(Hann, 2000) Leach, A. R.; Hann, M. M. The *in silico* world of virtual libraries. *Drug Discovery Today* **2000**, *5*, 326-336.

(Hopfinger, 1996) Hopfinger, A. J.; Holzgrabe, U. Conformational Analysis, Molecular Shape Comparison, and Pharmacophore Identification of Different Allosteric Modulators of Muscarinic Receptors. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 1018-1024.

(Hopkins, 2004) Hopkins, A. L.; Ren, J.; Milton, J.; Hazen, R. J.; Chan, J. H.; Stuart D. I.; Stammers, D. K. Design of Non-Nucleoside Inhibitors of HIV-1 Reverse Transcriptase with Improved Drug Resistance Properties. 1. *J. Med. Chem.* **2004**, *47*, 5912-5922.

(Jones, 1997) Jones, G.; Willett, P.; Glen, R. C.; Leach, A. R.; Taylor, R. Development and Validation of a Genetic Algorithm for Flexible Docking. *J. Mol. Biol.* **1997**, *267*, 727-748.

(Jorgensen, 2006) Jorgensen, W. L.; Ruiz-Caro, J.; Tirado-Rives, J.; Basavapathruni, A.; Anderson, K. S.; Hamilton, A. D. Computer-aided design of non-nucleoside inhibitors of HIV-1 reverse transcriptase. *Bioorganic & Medicinal Chemistry Letters* **2006**, *16*, 663-667.

(Kimura, 1998) Kimura, T.; Hasegawa, K.; Funatsu, K. GA Strategy for Variable Selection in QSAR Studies: GA-Based Region Selection for CoMFA Modeling. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 276-282.

(Klebe, 2006) Klebe, G. Virtual ligand screening: strategies, perspectives and limitations. *Drug Discovery Today* **2006**, *11*, 580-594.

(Koza, 2005) Koza, J. R.; Al-Sakran, S. H.; Jones, L. W. Automated Re-Invention of Six Patented Optical Lens Systems using Genetic Programming. In *Proceedings of GECCO 2005*; Beyer, H.G. et al., Ed.; ACM: New York, 2005; Vol 2, pp 1953-1960.

(Lazar, 2004) Lazar, C.; Kluczyk, A.; Kiyota, T.; Konishi, Y. Drug Evolution Concept in Drug Design: 1. Hybridization Method. *J. Med. Chem.* **2004**, *47*, 6973-6982.

(Lewell, 1998) Lewell, X. Q.; Judd, D. B.; Watson, S. P. and Hann, M. M. RECAP-Retrosynthetic Combinatorial Analysis Procedure: A Powerful New Technique for Identifying Privileged Molecular Fragments with Useful Applications in Combinatorial Chemistry. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 511-522.

(Morris, 1998) Morris, G. M.; Goodsell, D. S.; Halliday, R. S.; Huey, R.; Hart, W. E.; Belew, R. K.; Olson, A. J. Automated Docking Using a Lamarckian Genetic Algorithm and an Empirical Binding Free Energy Function. *J. Comput. Chem.* **1998**, *19*, 1639-1662.

(Nachbar, 1998) Nachbar, R. B. Molecular Evolution: A Hierarchical Representation for Chemical Topology and its Automated Manipulation. In *Proceedings of the Third Annual Genetic Programming Conference;* Madison, Wisconsin, 1998, pp 246-253.

(Nachbar, 2000) Nachbar R. B. Molecular Evolution: Automated manipulation of hierarchical chemical topology and its application to average molecular structures. *Genetic Programming and Evolvable Machines* **2000**, *1*, 57-94.

(NCI, 2000) NCI open database, as found on http://cactus.nci.nih.gov/ncidb2/ download.html, the August 2000 2D file.

(Nicolaou, 2009) Nicolaou, C. A.; Apostolakis, J.; Pattichis, C. S. De Novo Drug Design Using Multiobjective Evolutionary Graphs. *J. Chem. Inf. Model.* **2009**, *49*, 295-307.

(Payne, 1995) Glen, R. C.; Payne, A. W. R. A genetic algorithm for the automated generation of molecules within constraints. *J. Comput.-Aided Mol. Des.* **1995**, *9*, 181-202.

(Pegg, 2001) Pegg, Scott C.-H.; Haresco, J. J.; Kuntz, I. D. A genetic algorithm for structure-based de novo design. *J. Comput.-Aided Mol. Des.* **2001**, *15*, 911-933.

(Pubchem, 2009) http://en.wikipedia.org/wiki/PubChem

(Rees, 2003) Rees, P. Big pharma learns how to love IT. *Scientific Computing World* **2003,** 16-18.

(Ren, 1995) Ren, J.; Esnouf, R.; Garman, E.; Somers, D.; Ross, C.; Kirby, I.; Keeling, J.; Darby, G.; Jones, Y.; Stuart, D.; Stammers, D. High resolution structures of HIV-1 RT from four RT-inhibitor complexes. *Struct. Biol.* **1995**, *2*, 293-302.

(Schneider, 2000) Schneider, G.; Lee, M.-L.; Stahl, M.; Schneider, P. De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks. *J. Comput.-Aided Mol. Des.* **2000**, *14*, 487-494.

(Shen, 2003) Shen, L.; Shen, J.; Luo, X; Cheng, F.; Xu, Y.; Chen, K.; Arnold, E.; Ding, J.; Jiang, H. Steered Molecular Dynamics Simulation on the Binding of NNRTI to HIV-1 RT. *Biophys. J.* **2003**, *84*, 3547-3563.

(Sheridan, 2000) Sheridan, R. P.; SanFeliciano, S. G.; Kearsley, S. K. Designing targeted libraries with genetic algorithms. *J. Mol. Graph. Model.* **2000**, *18*, 320-334.

(Tan, 2005) Tan, T. Z.; Quek, C.; Ng, G. S. Brain-inspired Genetic Complementary Learning for Stock Market Prediction. In *Proceedings of IEEE CEC 2005*; IEEE Press: Piscataway, 2005; Vol 3, pp 2653-2660.

(Titmuss, 1999) Titmuss, S. J.; Keller, P. A.; Griffith, R. Docking Experiments in the Flexible Non-nucleoside Inhibitor Binding Pocket of HIV-1 Reverse Transcriptase. *Bioorganic & Medicinal Chemistry* **1999**, *7*, 1163-1170.

(Verdonk, 2004) Verdonk, M. L.; Berdini, V.; Hartshorn, M. J.; Mooij, W. T. M.; Murray, C. W.; Taylor, R. D.; Watson, P. Virtual Screening Using Protein-Ligand Docking: Avoiding Artificial Enrichment. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 793-806.

(Vinkers, 2003) Vinkers, H. M.; De Jonge, M. R.; Daeyaert, F. F. D.; Heeres, J.; Koymans, L. M. H.; Van Lenthe, J. H.; Lewi, P. J.; Timmerman, H.; Van Aken, K.; Janssen, P. A. J. SYNOPSIS: SYNthesize and OPtimize System in Silico. *J. Med. Chem.* **2003**, *46*, 2765-2773.

(Vinkers, 2004) Vinkers, H. M. *Computational drug design*. PhD thesis Free University, Amsterdam, the Netherlands, 2004.

(Westhead, 1996) Clark, D. E.; Westhead, D. R. Evolutionary algorithms in computer-aided molecular design. *J. Comput.-Aided Mol. Des.* **1996**, *10*, 337-358.

7 Designing Active Template Molecules by Combining Computational *De Novo* Design and Human Chemist's Expertise

Eric-Wubbo Lameijer, Reynier A. Tromp, Ronald F. Spanjersberg, Johannes Brussee, and Adriaan P. IJzerman
Leiden/Amsterdam Center for Drug Research, Division of Medicinal Chemistry, Leiden University, PO Box 9502, 2300 RA Leiden, The Netherlands

This chapter was first published in the Journal of Medicinal Chemistry, reference: Lameijer, E.W.; Tromp, R.A.; Spanjersberg, R.F.; Brussee, J.; IJzerman, A.P. Designing Active Template Molecules by Combining Computational De Novo Design and Human Chemist's Expertise. *J. Med. Chem.*, **2007**, *50* (8), 1925–1932.

Abstract

We used a new software tool for *de novo* design, the "Molecule Evoluator", to generate a number of small molecules. Explicit constraints were a relatively low molecular weight and otherwise limited functionality, for example low numbers of hydrogen bond donors and acceptors, 1 or 2 aromatic rings, and a small number of rotatable bonds. In this way we obtained a collection of scaffold- or template-like molecules rather than fully "decorated" ones. We asked medicinal chemists to evaluate the suggested molecules for ease of synthesis and overall appeal, allowing them to make structural changes to the molecules for these reasons. On the basis of their recommendations we synthesized 8 molecules with an unprecedented (not patented) yet simple structure, which were subsequently tested in a screen of 83 drug targets, mostly G protein-coupled receptors. Four compounds showed affinity for biogenic amine targets (receptor, ion channel and transport protein), reflecting the training of the medicinal chemists involved. Apparently the generation of lead-like solutions helped

the medicinal chemists to select good starting points for future lead optimization, away from existing compound libraries.

Introduction

Chemical space is vast – the number of potential drug-like molecules has been estimated to be beyond the number of atoms in the universe.[1,2] This is in sharp contrast with the total count of molecules in large compound databases such as CAS, with approximately 25 million references to chemical compounds.[3] Hence, *de novo* design is crucial to cover more of the chemical universe. Computational methods are particularly suitable for this goal, as they can quickly generate and store thousands of putative structures. Currently, there are dozens of *de novo* design programs, many of which have been covered in a recent review.[4] For example, the program CoG (Compound Generator) of Brown et al.[5] constructs molecules based on atoms and fragments that have been given as input to the program, eventually yielding molecules that resemble a number of selected ligands. Other programs construct new molecules based on the structure of the target protein. For example, DycoBlock[6] takes a list of fragments and searches for their optimal position in the active site of the protein. Then it searches for combinations of building blocks that could be linked together to form a new molecule.

We have recently developed a software tool to help medicinal chemists in designing new active structures; we called it “The Molecule Evoluator” (see also chapter 3 of this thesis). The Molecule Evoluator constructs molecules from atoms and a limited number of predefined larger fragments (such as phenyl and carboxylic acid groups). The use of atoms and the ability to attach atoms to any other atom and make rings at all chemically valid positions of a molecule allows an exhaustive search of chemical space and fine–tuning of the molecular structure.

An important difference between the Molecule Evoluator and most other *de novo* design programs is the focus on interaction with the user to produce lead compounds. Instead of generating a large database which is then screened virtually by docking or molecule similarity calculations, it presents a number of molecules to the user, who selects and edits the molecules to make them more lead-like. This cycle of computer generation and user modification can go on for several rounds, hence the name “Molecule Evoluator”. This user involvement was inspired by new approaches in computer science that stress the collaboration between computer and user, such as interactive evolutionary computing.[7] The user is able to use his implicit knowledge, e.g.

of synthetic feasibility, to eliminate structures suggested by the program that are difficult to make in the laboratory. The user may also bring in other areas of expertise, such as domain knowledge for a certain drug target, for example in the form of structure-activity relationships.

The aim of the present study was to determine whether combining computational inspiration with the domain knowledge of a number of medicinal chemists could produce novel, biologically active, lead-like structures. We used the Molecule Evoluator in a more constrained way than the usual cycle, in which the molecules modified by the user are fed back to the computer program to "breed" new molecules. Instead, we just created one database of molecules, the structures of which were refined by the medicinal chemists alone. For that we asked a panel of medicinal chemists to select, comment on and amend a limited number of compounds out of the library, which were subsequently checked for novelty. On the basis of their recommendations a limited number of the chosen and amended compounds, further simplified for reasons of chemical feasibility, was synthesized and tested on an array of drug targets. Half of the compounds synthesized possessed significant activity for biological targets, indicating that our combination of computer-based generation of molecules and chemist-based selection and modification can be useful to develop entirely novel lead structures.

Results

De novo design of template molecules

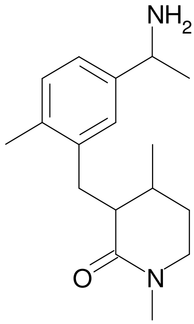
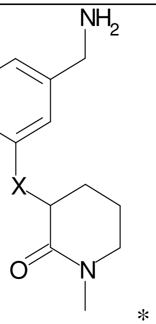
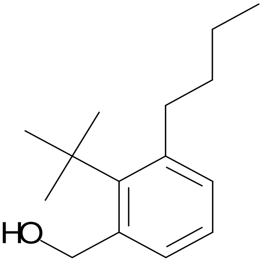
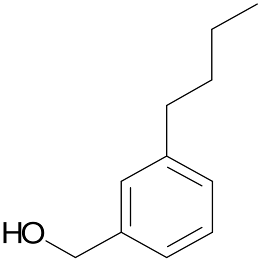
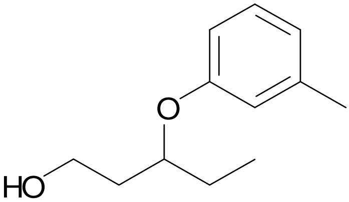
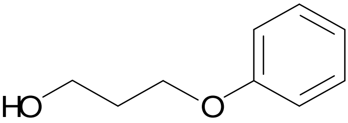
We used the Molecule Evoluator to generate a virtual library of 300 compounds according to a number of restrictions meant to produce template-like rather than drug-like molecules. These limitations, extended on but stricter than Lipinski's "rule of five",[8] were as follows:

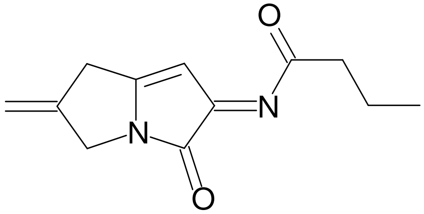
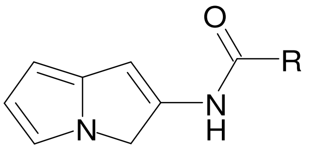
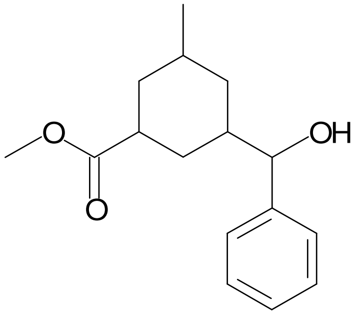
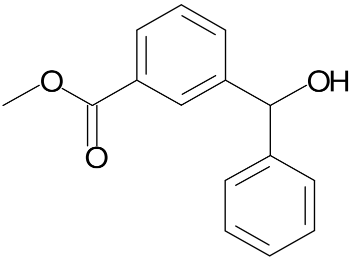
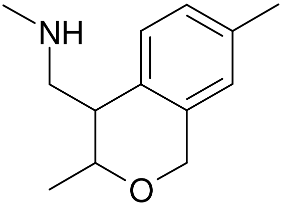
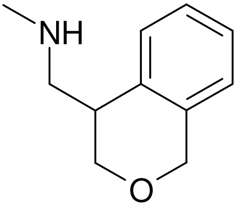
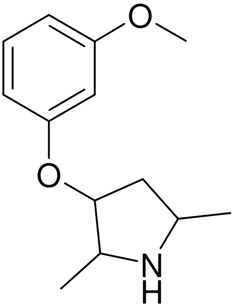
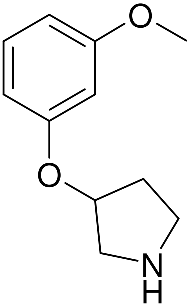
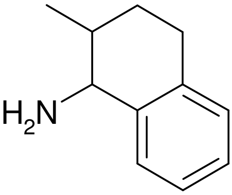
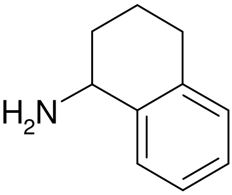
1) At least one and at most two aromatic systems
2) Polar surface area equal to or below 70 Å^2
3) A maximum number of two hydrogen bond donors and four hydrogen bond acceptors
4) Not more than five rotatable bonds

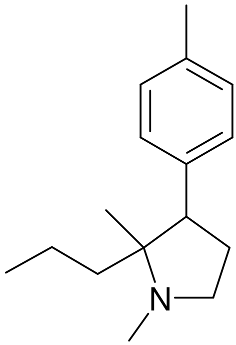
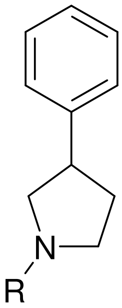
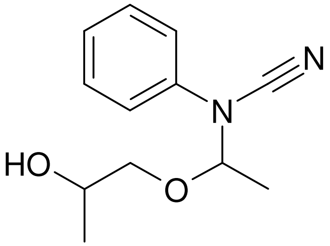
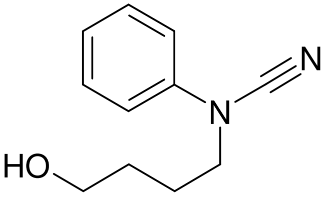
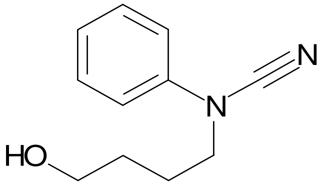
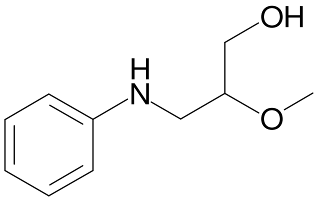
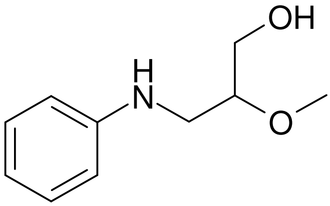
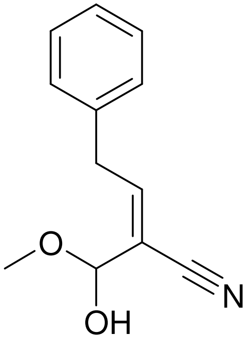
Although we also experimented with molecular weight restrictions we learned that the above four criteria invariably resulted in compounds with molecular weights lower than 400 D, hence lower than "Lipinski's" cut-off of 500 D.

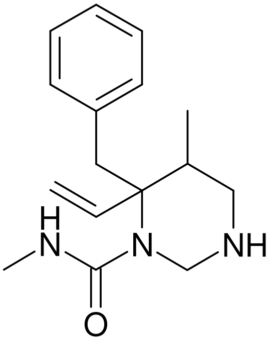
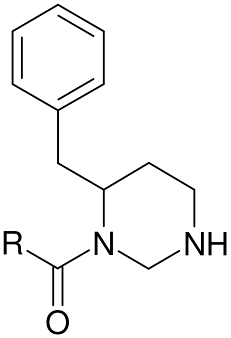
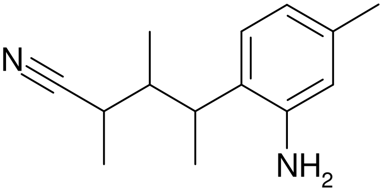
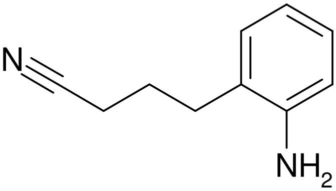
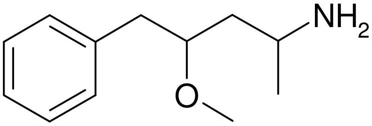
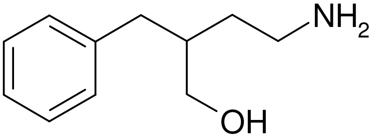
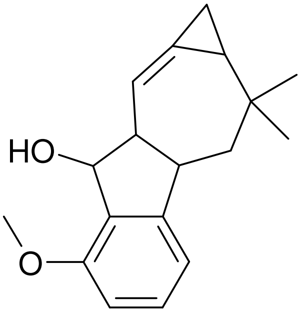
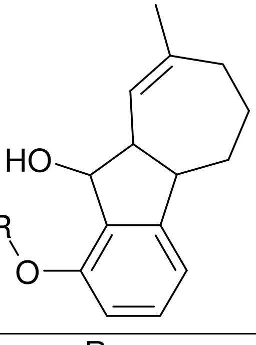
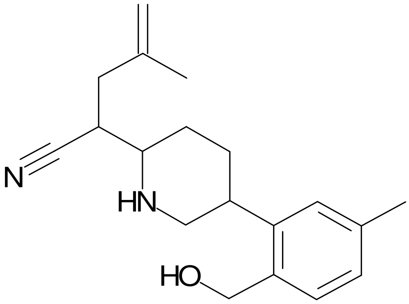
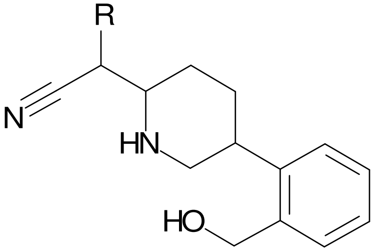
The 300 compounds were presented to a panel of five medicinal chemists with different backgrounds (chemistry of peptides, biogenic amines (2x), nucleosides/ nucleotides, and chiral synthesis). They were asked to select at least 10 compounds to their liking. Specifically, the selected compounds had to look drug-like and synthetically feasible, or at least be amenable to be changed into such compounds by minor modifications. This led to a total of 34 compounds (Table 7.1).

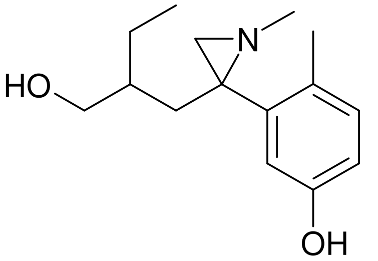
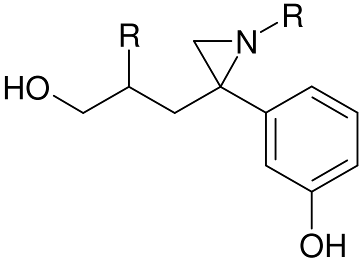
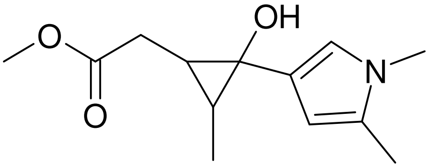
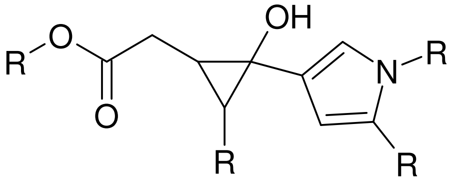
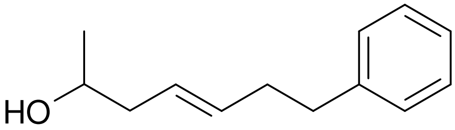
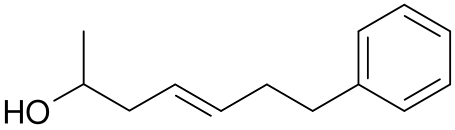
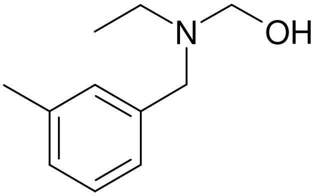
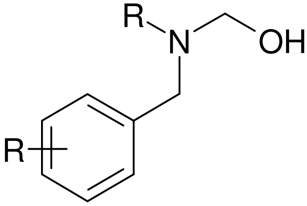
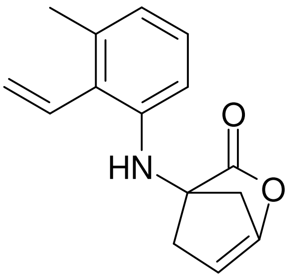
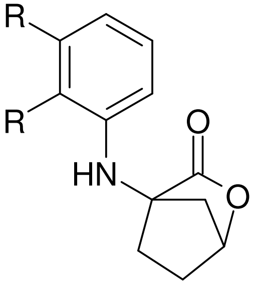
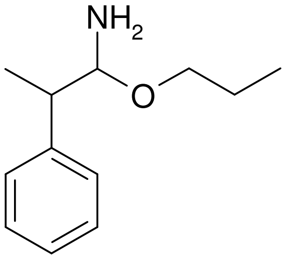
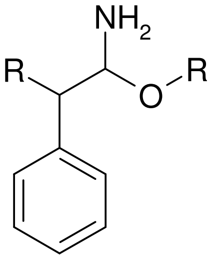
Table 7.1: The 34 compounds selected from the 300-member library generated by the Molecule Evoluator. The left-hand column shows the structures as generated, while the right-hand column lists the structures after initial amendment by the medicinal chemists. Those molecules marked with a star were selected for further amendment and synthesis.

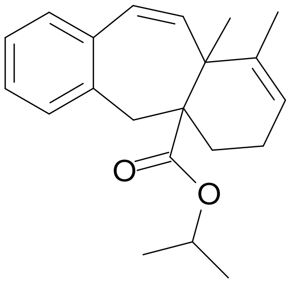
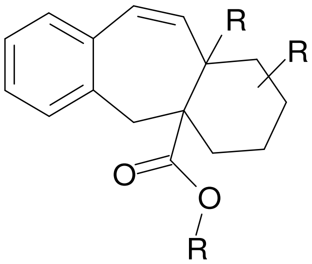
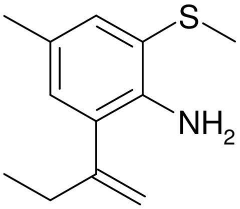
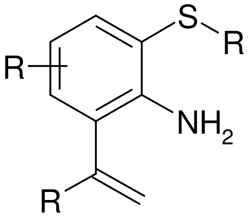
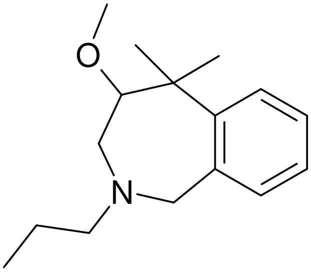
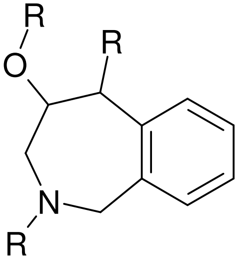
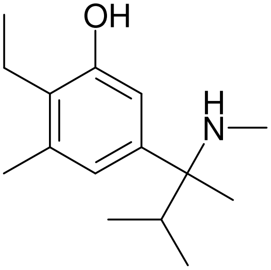
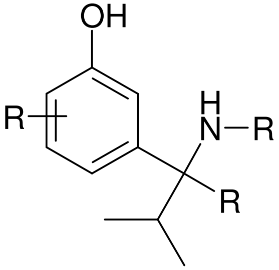
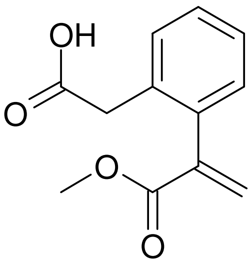
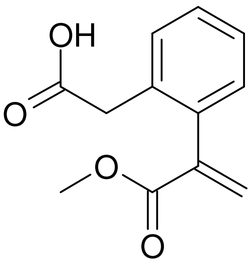
Original molecule	Modified molecule
	 *
	
	

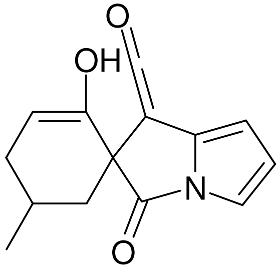
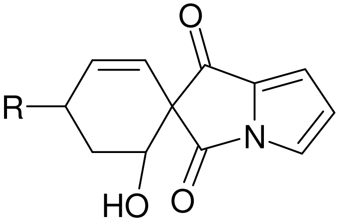
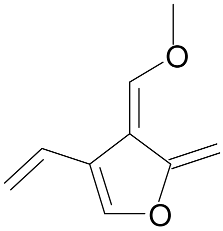
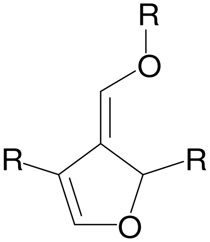
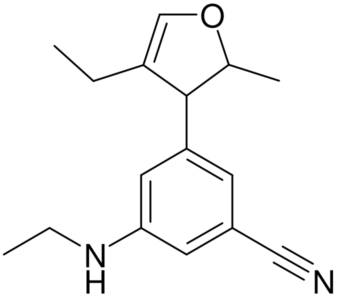
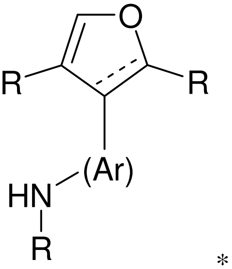
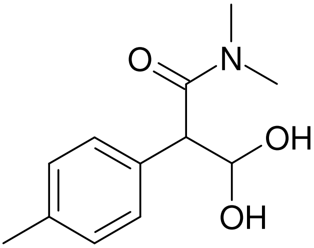
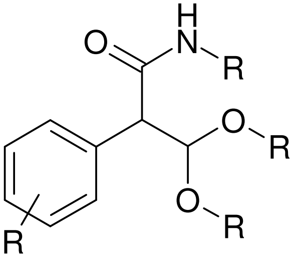
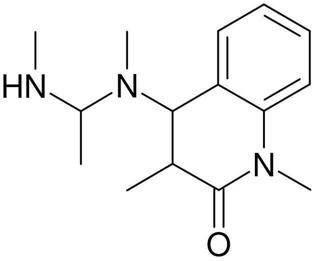
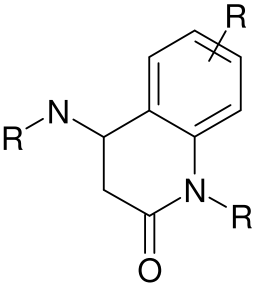
 O N N O	 O R N NH *
 O OH O	 O OH O
 NH O	 NH O
 O O NH	 O O NH
 H_2N	 H_2N

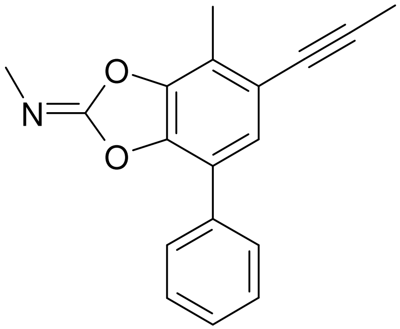
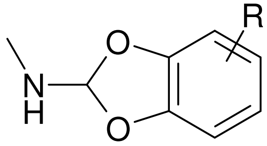
	
	
	
	

H N N NH O	R N NH O *
N NH2	N NH2
NH2 O	NH2 OH
HO O	HO R O
N HN HO	R N HN HO *

Our next step was to inspect the 34 molecules for novelty, ease of synthesis and drug-likeness. Novelty in this case was defined as absence from both the Beilstein and SciFinder databases either as a structure or substructure.[3,9] This process took place in March 2003; we did not check for later occurrence. For ease-of-synthesis we allowed the chemists to modify the suggested structures to reduce the anticipated number of synthetic steps (maximally 3 from a commercially available starting material). Drug-likeness was not only based on the filters that we already applied when the virtual library was generated, but also on the intuition of the individual medicinal chemist. All in all this led to a top-nine of compounds that formed the start for our synthetic program (see Figure 7.1). Two chemists (R.T. and R.S.) were allotted a restricted period of time to try and synthesize these compounds. It was decided to rapidly terminate a project whenever synthetic feasibility in practice was less than anticipated ‘on paper’. This was particularly true for compounds **1f** and **1g**. It was also decided to allow further variations on the nine molecules presented in Figure 7.1 on the basis of experimental findings in the synthetic program. As a consequence the final series of compounds, although much inspired by the very first suggestions, deviated from the original structures. In general, the computer-generated molecules were simplified by eliminating most substituents, while the core structure was retained together with one or two of the most important or interesting substituents. Further variation was produced by making derivatives of the remaining substituents (such as oxidizing a CH_2OH group to COOH). Eventually we prepared and characterized eight compounds as represented in Figure 7.2. Their synthesis is outlined in the chemistry paragraph below and described in full detail in the Experimental Section.

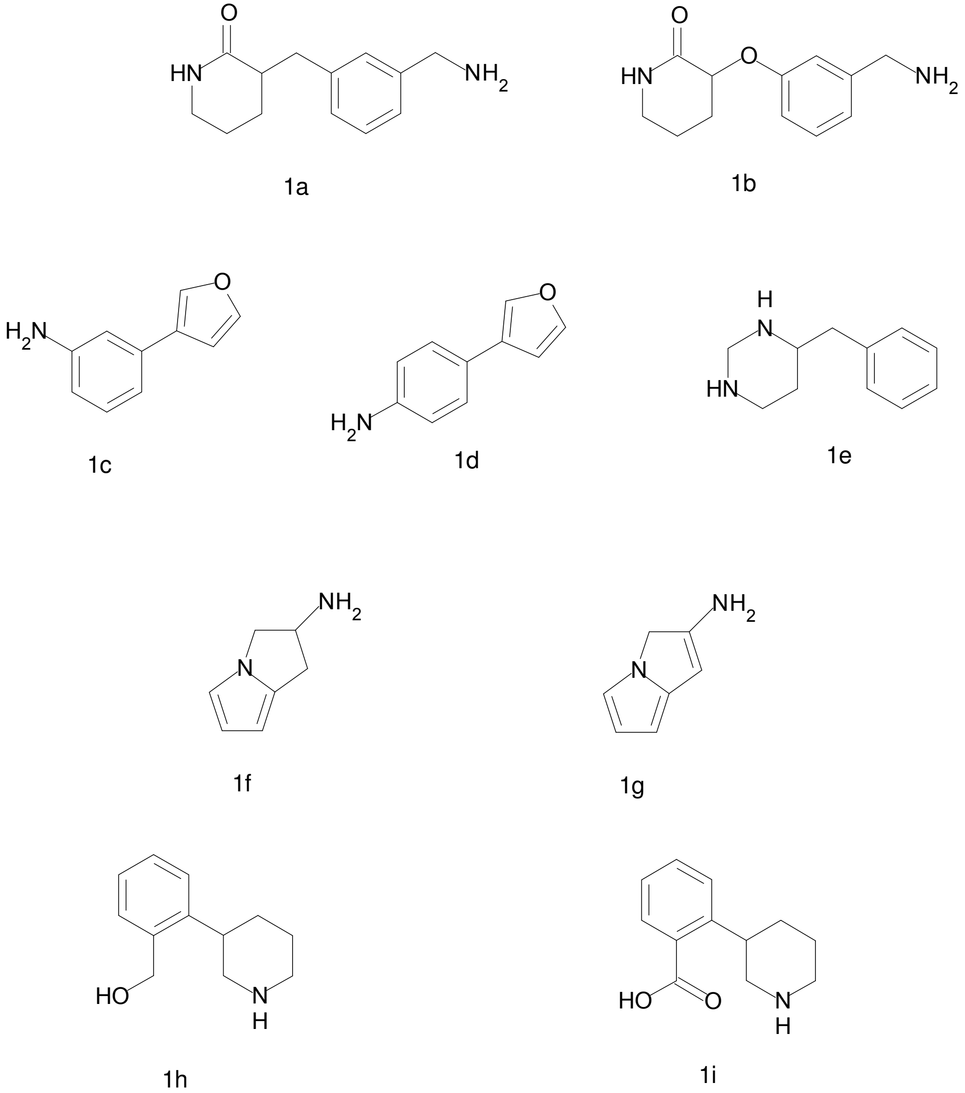
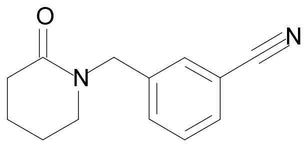
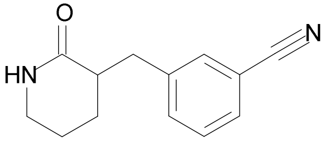


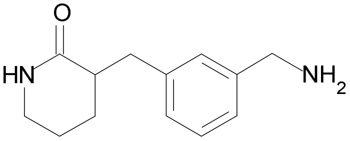
Figure 7.1: The final selection of nine compounds (1a-1i), amended by the panel of medicinal chemists.



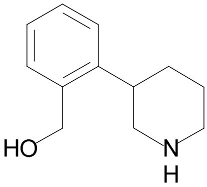
3



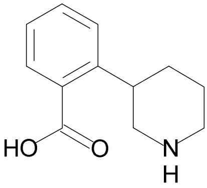
4



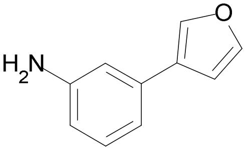
5 (1a)



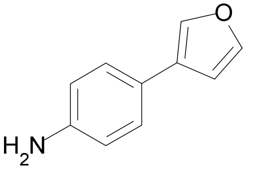
6 (1h)



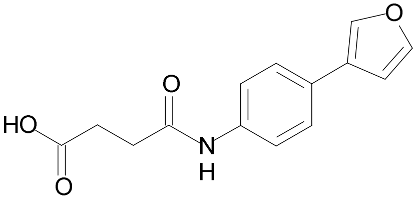
7 (1i)



8 (1c)



9 (1d)

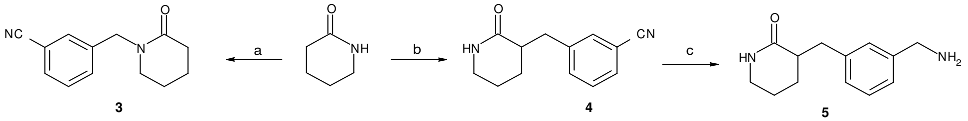


10

Figure 7.2: The eight compounds resulting from the synthetic program (3-10)

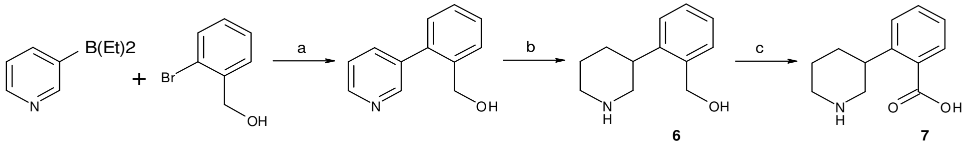
Chemistry

Compound **3** was prepared by substitution of 3-(bromomethyl)benzonitrile with 2-piperidinone which was deprotonated with one equivalent of butyllithium.[10] Synthesis of compound **4** was performed by alkylation at the 3-position of 2-piperidinone via the enolate anion in which the nitrogen atom was temporarily protected with TMS.[11] Hydrogenation of compound **4** with Pd/C as catalyst afforded the benzylamino compound **5** (Scheme 7.1).



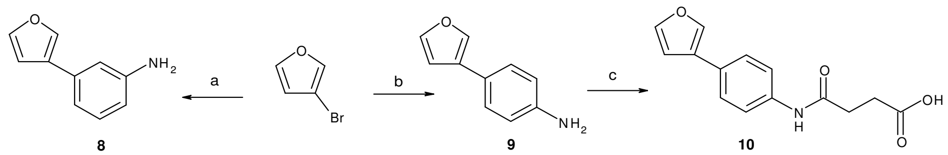
Scheme 7.1: a: 1 eq. *n*-BuLi, 3-(bromomethyl)benzonitrile. b: TMSCl, *n*-BuLi, 3-(bromomethyl)benzonitrile. c: Pd/C 10%, H_2.

Synthesis of 2-(3-piperidyl)-benzyl alcohol (**6**) was done by a two step reaction. First 2-(3-pyridyl)-benzyl alcohol was prepared by a Suzuki reaction of diethyl(3-pyridyl)borane and 2-bromobenzyl alcohol under microwave conditions.[12] The product of this reaction was hydrogenated under acid conditions with PtO_2 as catalyst and provided compound **6**. Benzyl alcohol **6** was oxidized with chromic acid and isolated as zwitterion. Purification was problematic, however preparative HPLC provided pure product **7** (Scheme 7.2).



Scheme 7.2: a: Na_2CO_3, TBAB, $(Ph_3P)_4Pd$, H_2O, MW. b: HCl, PtO_2/H_2. c: Jones' reagent.

The most straightforward way to prepare compounds **8** and **9** was the Suzuki coupling reaction of 3-bromofuran with boron derivatives of 3- and 4-aniline respectively, under microwave conditions. Compound **10** was prepared from **9** by reaction with succinic acid and crystallization from diethyl ether (Scheme 7.3).



Scheme 7.3: a: 3-aminophenyl-boranic acid, Na_2CO_3, TBAB, $(Ph_3P)_4Pd$, MW. b: 4-(4,4,5,5-tetramethyl-1,3,2-dioxaborolan-2-yl)aniline, Na_2CO_3, TBAB, $(Ph_3P)_4Pd$, MW. c: succinic anhydride, 4-methyl-morpholine.

Biology

We tested the 8 compounds in a commercially available screening program. Radioligand binding and enzyme assays, 68 and 15, respectively, were the read-outs to probe the interaction of the individual compounds with this large collection (83) of drug targets. These included G protein-coupled receptors (rhodopsin-like, class A; metabotropic glutamate-like, class C), ion channels (for Na^+, K^+, Ca^{2+}), nuclear hormone receptors (e.g., estrogen, progesterone), transport proteins (e.g. for dopamine, norepinephrine, GABA), and enzymes (several phosphodiesterases, Na^+/K^+-ATPase, etc). All compounds were tested in duplicate at a single concentration of 10 μM. In Table 7.2 the percentage inhibition of specific radioligand binding to the indicated target (with a minimum of 30%) is shown. Negative values indicate an increase in specific binding. This might indicate an allosteric mechanism of enhancement,[13] but this was not investigated further. Four out of eight compounds displayed activity in a number of radioligand binding assays, while none of the compounds appeared active in the enzyme assays. Compounds **5** (imidazoline and muscarinic receptors), **6** (α-adrenergic receptors), and **8** and **9** (norepinephrine transport protein) caused approximately 50% radioligand displacement or more. It should be mentioned here that compound **7**, being inactive at all tested targets, appealed to one of the chemists for a different reason, i.e. it being an unnatural and new amino acid, which will be used for incorporation in modified peptides.

Table 7.2: Percentage inhibition of specific radioligand binding (min. 30%) to the indicated target by 10 µM of test compounds **3-10**. Negative values indicate an increase in specific binding.

Target		**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
CB$_1$				-33					
I$_2$				49	36		41	32	
M$_{1-5}$				50					
NAch					45		42		
NE transp.							80	77	
DA transp.							-62		
5-HT transp.							-31		
kainate								-37	
α_1-adrenerg.					47				
α_2-adrenerg.					62				
NK$_1$					-32				
opiate					35				
5-HT					39				

CB$_1$: cannibinoid receptor 1; I$_2$: imidazoline receptor 2; M$_{1-5}$: muscarinic receptors 1-5 in rat brain; NAch: nicotinic acetylcholine ion channel; NE transp: norepinephrine transport protein; DA transp: dopamine transport protein; 5-HT transp: serotonin transport protein; kainate: glutamate/kainate receptor; NK$_1$: neurokinin receptor 1; opiate: all opioid receptors in rat brain; 5-HT: serotonin receptors in rat brain.

Discussion

For a medicinal chemist, drug-likeness, synthetic feasibility and overall ‘molecule appeal’ are very important criteria in drug design. However, these features are very difficult to quantify, such that good ‘scoring functions’ are often lacking. For instance, computer-assisted organic synthesis was recently reviewed by Todd,[14] who concluded that available software invariably required human intervention to be useful. Similarly, computational approaches to predict ligand binding affinity for a given target protein ("docking") are notoriously inaccurate. Aware of such considerations we decided to rely on the user as evaluator. A user cannot know the binding strength of a given molecule *a priori*, but we reasoned this defect may not be much worse than the inaccuracy of scoring functions. A definite advantage in letting the user choose would be that intensive feedback from a medicinal chemist would make the compounds easier to synthesize, and steer the idea generation away from areas which have already been explored. Furthermore, user feedback could still be combined with experimental results or advanced computed fitness functions if so desired. Considering these advantages we developed a software tool for *de novo* molecule design, called the Molecule Evoluator, which we recently described. It contains a graphical user interface and has options for directly editing the molecule, marking part of a molecule as conserved, and calculating relevant physicochemical parameters.[7]

It should be noted that the Molecule Evoluator mainly uses the atom-based approach to construct molecules, that is, a molecule is built from individual atoms and bonds, though some predefined fragments can be added. A number of other researchers have also constructed molecules in an atom-based way, for example Nachbar[15], Douguet et al.[16] and Brown et al.[5] Others construct molecules from a number of multi-atom fragments, such as Pegg et al.[17], Vinkers et al.[18], and Schneider et al.[19] The main difference between atom-based and fragment-based methods is not so much the size of the fragments used (atom-based methods often also use fragments, and vice versa) but the emphasis placed on synthetic feasibility. Atom-based methods such as ours sample the entire chemical space but also produce molecules of doubtful synthetic feasibility, and fragment-based methods like the one of Vinkers et al.[18] stress synthetic accessibility and therefore sample a much smaller part of chemical space, excluding hard-to-synthesize molecules but also many potential drugs. In the Molecule Evoluator, we have chosen for the flexibility of the atom-based approach, although we are aware of the sensitive issue of synthetic ease and have developed a number of features which allow the user to restrict the variety of molecules produced.[7]

In the present study we generated 300 molecules according to the criteria specified in the Results section. These criteria are well below the classic 'rule-of-five'[9] to largely yield template or scaffold-like molecules only. For example, the number of hydrogen bond donors was confined to a value of two, rather than five. Repeating the experiment would yield a largely different library of molecules due to the random-number generator in our algorithm. While by setting the criteria identical to our original experiment the average physicochemical properties of such molecules would be similar to those in the original library, the enormous number of molecules possible with a molecular weight between, say, 150 and 300, would ensure that there would be barely any molecules in common between the two libraries. Changing the parameters would force the algorithm to sample another part of chemical space, however the physicochemical properties of the lead structures might not change as greatly as the parameters since the chemists generally adapt the molecules to the complexity of their taste.

The 300 molecules were shown to a panel of medicinal chemists. They examined them for drug-likeness, synthetic feasibility, and overall appeal as mentioned above, and identified their preferences. It should be noted that human judgment is not unequivocal. In a study by Takaoka and coworkers, five chemists judged a collection of almost 4000 molecules in a Japanese corporate database for their drug-likeness and ease of synthesis. Their scores showed considerable variation.[20] A similar inconsistency was noted among 13 medicinal chemists at a US-based company when asked to reject compounds with undesirable properties from one or more lists of 2000 compounds each.[21] Apparently unanimity among medicinal chemists is not self-evident. On a more positive note, their diversity in opinion may in fact constitute an important and discriminative asset for a research group. While our computational generation of the library benefited from human intervention, the chemists themselves also found that the computational generation of molecules added value. They appreciated the many choices possible, which emphasizes that it is easier to *recognize* a “good” structure than to *invent* one.

We did not give the chemists explicit instructions on which molecules should be chosen or rejected, other than that the molecules should seem lead-like and not too difficult to synthesize. Analyzing their choices in retrospect, it became clear that the chemists did use some general “implicit” rules for molecule choice. Molecules without heteroatoms (or with only one heteroatom if that was a nitrogen) were almost always rejected, as were molecules with more than two ring systems, cyclophanes (having a bridged benzene ring), molecules with odd or unwanted groups like halogen atoms or nitro groups, and molecules with many alkyl substituents. However, if anything, these rules seemed more like a weighing of attractive and inattractive features than a black-or-white approval or elimination. For example, one molecule with only one heteroatom, a nitrogen, was nevertheless selected, probably because the nitrogen was in a two-ring system instead of somewhere in a substituent. Occasionally the chemists disagreed about the appropriateness of a certain selection. So instead of general rules one could say that the chemists used general guidelines, which were weighed according to individual experience and taste.

The compounds that were suggested (Figure 7.1) and eventually synthesized (Figure 7.2) all had a relatively small number of hydrogen bond donors and/or acceptors, next to their low molecular weight, as a logical consequence of the strict criteria imposed. They largely adhere to a recently proposed "rule-of-three" for fragment-based lead discovery, in which molecular weight is <300, the number of hydrogen bond donors is ≤3, the number of hydrogen bond acceptors is ≤3 and the calculated logP value is ≤3[22], and can be considered leads[23] or fragments rather than potential drugs. In this view fragments should have features that, when combined, still adhere to Lipinski's "rule-of-five". The differences between "rule-of-three" and "rule-of-five" allow a further "decoration" of our compounds. At the same time, fragments tend to have very low affinity for a given target, in view of the limited options for interaction.[22] Surprisingly, quite a few of our compounds displayed affinities that allowed them to be recognized in conventional radioligand binding assays, as opposed to more sophisticated and demanding NMR- or X-ray-based screening that is generally applied in fragment-based approaches.

It appeared that most of our ligands intervened with targets for biogenic amines (e.g., adrenergic, muscarinic and serotonin receptors, norepinephrine transport protein, nicotinic acetylcholine ion channel). Interestingly, the background, education and training of some of our medicinal chemists involved in the selection of the compounds had been focused on this important ligand class, suggesting that medicinal chemists can indeed develop a "feel" for a certain target or family of targets.

The chemical structures of the suggested molecules as well as those synthesized are simple, or, as some medicinal chemists put it, "quite boring". Apparently, chemical space is vast, but also nearby, i.e. entirely novel structures can be far from exotic. It suggests that medicinal chemists when asked tend to prefer more uncommon structures. Interestingly, it has been shown on a number of occasions that currently available drugs in fact have low diversity.[24,25] In a recent analysis of the NCI database harboring over 250,000 molecules tested for biological activity we learned that in it 80% of all ring systems found in molecules belonged to one out of the 66 "top" ring systems – which was only 0.5% of the total variety in ring systems in the database. The same analysis taught us that a phenyl ring was present in almost half of the compounds, whereas the next most prevalent (pyridine) ring occurred in less than 3% of the molecules,[27] "quite boring" indeed. The reason may be that exotic ring systems and substituents have undesirable synthetic or biological properties. It emphasizes that our method of template development, which puts "ordinary" parts in novel combinations, may actually be quite suitable for drug design.

Conclusion

Computational generation of novel molecules, as implemented in the Molecule Evoluator, appeared useful in *de novo* template and scaffold design. It helped a panel of medicinal chemists in generating, amending and selecting a number of ‘simple’ yet novel chemical entities. A number of low-molecular weight compounds was eventually synthesized and tested on a diverse panel of drug targets. Some of the compounds proved to be active, mainly on targets for biogenic amines, in line with the background and expertise of some of the medicinal chemists. It seems that nearby chemical space still offers substantial room for drug design, and that simple structures can be very attractive.

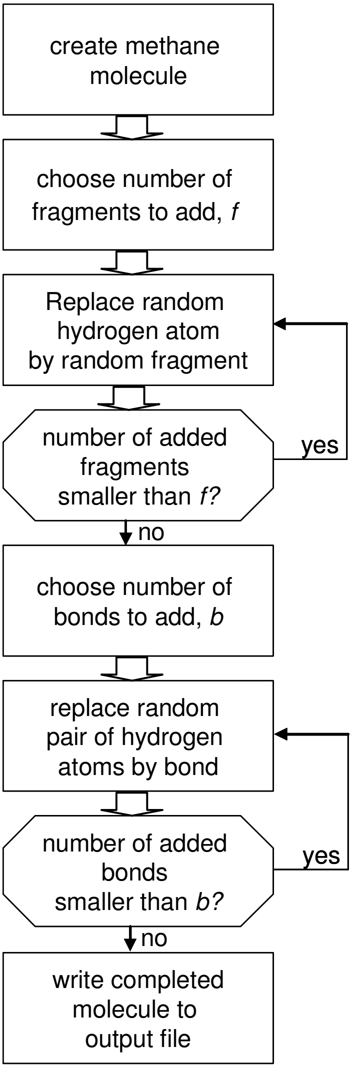
Experimental

De novo design algorithm

The 300 molecules were generated by taking a methane molecule, and growing the molecule for a number of iterations by attaching atoms to it at random positions, and adding double bonds and rings. The algorithm is shown in Figure 7.3.

If a molecule did not obey pre-set criteria (at least one and at most two aromatic systems, polar surface area (calculated according to Ertl et al.[26]) equal to or below 70 Å^2, a maximum number of two hydrogen bond donors and four hydrogen bond acceptors, not more than five rotatable bonds) it was discarded and a new molecule was generated, until we had 300 molecules with the desired physicochemical properties.

Figure 7.3: Flowchart of the *de novo* design algorithm. A molecule is generated by adding a random number of fragments (varying from 1 to 16) to a methane molecule, and subsequently adding bonds, thereby creating double bonds and rings. The exact number of rings and double bonds is determined by a weighted probability table, as is the ring size (so a 5-membered ring is more frequent than an 8-membered ring, like in normal chemical databases).


create methane molecule
choose number of fragments to add, *f*
Replace random hydrogen atom by random fragment
number of added fragments smaller than *f?*
yes
no
choose number of bonds to add, *b*
replace random pair of hydrogen atoms by bond
number of added bonds smaller than *b?*
yes
no
write completed molecule to output file

Chemistry

Microwave reactions were performed in an Emrys^{TM} Optimizer (Biotage AB). Wattage was automatically adjusted so as to maintain the desired temperature. Column chromatography was performed on Baker Silica Gel (0.063-0.200 mm). For TLC analysis, Schleicher and Schuell F1500/LS 254 silica plates were used. Spots were visualised with ultraviolet light. ^{1}H NMR and ^{13}C NMR were recorded with a Bruker AC 200 spectrometer at room temperature. Tetramethylsilane was used as internal standard; δ in ppm, *J* in Hz. Melting points were determined with a Büchi melting point apparatus and are uncorrected. High Resolution Mass spectroscopy was performed on a PE-Sciex API Qstar instrument. Elemental analyses were within 0.4% of the theoretical values.

1-(3-Cyanobenzyl)-piperidin-2-one (3)

A solution of 2-piperidinone (5 mmol) in THF (25 mL) was stirred for 1 h at 0 °C before 1 eq. of *n*-BuLi (5 mmol, 3.2 mL of a 1.6 M solution in hexane) were added dropwise. After stirring for another hour at 0 °C 1 eq. of 3-(bromomethyl)benzonitrile (5 mmol) was added rapidly. The mixture was allowed to warm slowly to room temperature and stirred overnight. After quenching by adding 15 mL of brine, the solvent layers were separated. To the aqueous layer was added 20 mL of water. After extraction of the water layer with CH_2Cl_2 the combined organic layers were dried (Na_2SO_4), filtered and the solvents evaporated. The product was purified by column chromatography (eluent: CH_2Cl_2/MeOH, 99/1→97.5/2.5 v/v). Yield: 24%. White solid. M.p.: 53-55 °C. Anal. ($C_{13}H_{14}N_2O$) C, H, N.

3-(3-Cyanobenzyl)-piperidin-2-one (4)

To a solution of 2-piperidinone (10 mmol) in THF (15 mL) was added at -78 °C 1 eq. of *n*-BuLi (10 mmol; 6.3 mL of a 1.6 M solution in hexane). After stirring for 15 minutes at -78 °C 1.1 eq. of TMSCl was added and the solution was allowed to warm to room temperature and left to stir for 45 min. The resulting solution was added at -78 °C to a solution of 11 mmol of 1,1,1,3,3,3-hexamethyldisilazane and 11 mmol of *n*-BuLi (6.9 mL of a 1.6 M solution in hexane) in 20 mL of THF. After stirring for 15 min 3-(bromomethyl)benzonitrile (11 mmol) was added and the mixture was allowed to warm slowly to -25°C, before the reaction was quenched by adding an aqueous NH_4Cl (sat.) solution. After extraction with diethyl ether the combined organic layers were washed with a saturated NH_4Cl (aq.) solution, a saturated $NaHCO_3$ (aq.) solution, dried ($MgSO_4$) and the solvents removed by evaporation. The product was purified by column chromatography (eluent: CH_2Cl_2/MeOH, 99/1→98/2 v/v). Yield: 47%. White crystals. M.p.: 95-96 °C. Anal. ($C_{13}H_{14}N_2O$) C, H, N.

3-(3-Benzylamino)-piperidin-2-one (5)

Compound **4** (2 mmol) was dissolved in methanol, and 2 mmol of concentrated HCl and 100 mg of Pd/C 10% were added. The mixture was hydrogenated at 3 atm for 3 h. After the catalyst was filtered off and the methanol was evaporated the residue was dissolved in water and the pH was adjusted to 4. This solution was washed with ether and the water layer was adjusted with 0.1 M NaOH to pH 12. The free amine was extracted with CH_2Cl_2, dried (Na_2SO_4) and the solvent evaporated. White powder. Yield: 31%. M.p.: 114-116 °C. Anal. ($C_{13}H_{18}N_2O$) C, H, N.

2-(3-Pyridyl)-benzyl alcohol

A suspension of 2-bromobenzyl alcohol (1 mmol), and diethyl(3-pyridyl)borane (1 mmol), Na_2CO_3 (3.8 mmol), TBAB (1 mmol), and $(Ph_3P)_4Pd$ (3%) in 2.5 mL of water was heated in a microwave for 12 min at 150 °C. The product was extracted with ethyl acetate. The combined organic layers were dried ($MgSO_4$), filtered and the solvent was evaporated. The product was purified by flash column chromatography. Eluent column: CH_2Cl_2/MeOH, 99/1→96/4 v/v. Yield: 79%. Oil.

2-(3-Piperidyl)-benzyl alcohol (6)

A mixture of 5.77 mmol of 2-(3-pyridyl)-benzyl alcohol, HCl (5.77 mmol) and PtO_2 (0.38 mmol) in 46 mL of absolute ethanol was placed in a Parr apparatus under H_2 (3 atm) for 3 days. The catalyst was filtered off and the solvent evaporated. After addition of water to the residue the pH was adjusted to 12 and the product was extracted with ethyl acetate. The combined organic layers were dried ($MgSO_4$) and the solvent was evaporated. Recrystallisation from ethyl acetate provided the pure product. Yield: 27%. White needles. M.p.: 135 °C. Anal. ($C_{12}H_{17}NO$) C, H, N.

2-(3-Piperidyl)-benzylic acid (7)

Compound **6** (2 mmol) was dissolved in 50 mL of acetone. Jones' reagent (chromic acid) was added slowly until the orange colour persisted. The pH of the mixture was adjusted to 7 with 1 M NaOH and the product was extracted with ethyl acetate. The combined organic layers were dried ($MgSO_4$), filtered and the solvent was evaporated. The product was purified by preparative HPLC. Anal. ($C_{12}H_{15}NO_2$) C, H, N.

3-(3'-Furyl)-aniline (8)

A suspension of 3-bromofuran (1 mmol), 3-aminophenyl-boranic acid (1 mmol), Na_2CO_3 (3.8 mmol), tetrabutylammonium bromide (1 mmol), and $(Ph_3P)_4Pd$ in 2.5 mL of water was heated in a microwave for 12 min at 150 °C. The product was extracted with ethyl acetate. The combined organic layers were dried ($MgSO_4$), filtered and the solvent was evaporated. The product was purified by flash column chromatography. Eluent: CH_2Cl_2. Yield: 74%. Yellowish solid. M.p.: 73-74 °C. Anal. ($C_{10}H_9NO$) C, H, N.

4-(3'-Furyl)-aniline (9)

A suspension of 3-bromofuran (1 mmol), 4-(4,4,5,5-tetramethyl-1,3,2-dioxaborolan-2-yl)aniline (1 mmol), Na_2CO_3 (3.8 mmol), tetrabutylammonium bromide (1 mmol), and $(Ph_3P)_4Pd$ in 2.5 mL of water was heated in a microwave for 12 min at 150 °C. The product was extracted with ethyl acetate. The combined organic layers were dried ($MgSO_4$), filtered and the solvent was evaporated. The product was purified by flash column chromatography. Eluent: CH_2Cl_2. Yield: 78%. Yellow solid. M.p.: 92-93 °C. Anal. ($C_{10}H_9NO$) C, H, N.

4-Oxo-4-[4-(3'-furyl)-phenylamino]-butanoic acid (10)

To a solution of 0.63 mmol 4-(3'-furyl)-aniline (**8**) in 10.5 mL of CH_2Cl_2 were added succinic anhydride (0.63 mmol) and 4-methylmorpholine (0.63 mmol). After stirring for 4.5 h the mixture was filtered, the residue washed with CH_2Cl_2 and the filtrate evaporated to dryness. The product was purified by chromatography (eluent: CH_2Cl_2/MeOH, 9/1 v/v). Yield: 27%. Yellow solid. M.p.: 198 °C (dec.). Anal. ($C_{14}H_{13}NO_4$•$0.3CH_3OH$) C, H, N.

Biology

The final compounds (Figure 7.2) were tested at one concentration (10 μM) in duplicate in the Diversity Profile program, including 68 receptors and 15 enzymes, at Cerep (Paris, France).

Software

For the template design we used the Molecule Evoluator software package (Cidrux Pharminformatics, Haarlem, the Netherlands, www.cidrux.com).

Acknowledgements

The authors thank Lisa Chang, Gijs van der Marel and Marc Overhand for their involvement in the template selection process. Gruenenthal GmbH is gratefully acknowledged for their financial contribution, as is Biopartner for support from its First Stage Grant program.

References

[1] Bohacek, R.S.; McMartin, C; Guida, W.C. The art and practice of structure-based drug design: a molecular modelling perspective. *Med. Res. Rev.* **1996**, *16*, 3–50.

[2] Dobson, C.M.; Chemical space and biology. *Nature* **2004**, *432*, 824-828.

[3] as taken from http://www.cas.org/chemplus/chemplus1.html and http://www.cas.org/ SCIFINDER/ataglance.html (accessed on April 26, 2006)

[4] Schneider, G.; Fechner, U. Computer-based *de novo* design of drug-like molecules. *Nature Rev. Drug Disc.* **2005**, *4*, 649-663.

[5] Brown, N.; McKay, B.; Gilardoni, F.; Gasteiger, J. A Graph-Based Genetic Algorithm and Its Application to the Multiobjective Evolution of Median Molecules. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 1079-1087.

[6] Zhu, J.; Yu, H.; Fan, H.; Liu H.; Shi Y. Design of new selective inhibitors of cyclooxygenase-2 by dynamic assembly of molecular building blocks. *J. Comput. Aid. Mol. Des.* **2001**, *15*, 447-463.

[7] Takagi, H. Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. In: *Proceedings of the IEEE*, **2001**, *89*, 1275-1296.

[8] Lipinski, C.A.; Lombardo, F.; Dominy, B.W.; Feeney, P.J. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv. Drug Deliv. Rev.* **1997**, *23*, 3-25.

[9] http://www.mdl.com/products/knowledge/crossfire_beilstein/

[10] Cossy, J.; de Filippis, A.; Gomez Pardo, D. Palladium-catalyzed intermolecular α-arylation of N-protected 2-piperidinones. *Org. Lett.* **2003**, *5*, 3037-3039.

[11] Brimble, M.A.; Trzoss, M. A double alkylation-ring closing metathesis approach to spiroimines. *Tetrahedron* **2004**, *60*, 5613-5622.

[12] Miyaura, N.; Suzuki, A. Palladium catalyzed cross-coupling reaction of organoboron compounds. *Chem. Rev.* **1995**, *95*, 2457-2483.

[13] Christopoulos, A.; Kenakin, T. G protein-coupled receptor allosterism and complexing. *Pharmacol. Rev.* **2002**, *54*, 323-374.

[14] Todd, M. Computer-aided organic synthesis. *Chem. Soc. Rev.* **2005**, *34*, 247-266.

[15] Nachbar, R. Molecular Evolution: A Hierarchical Representation for Chemical Topology and its Automated Manipulation. In *Proceedings of the Third Annual Genetic Programming Conference;* Madison, Wisconsin, **1998**, 246-253.

[16] Douguet, D.; Thoreau, E.; Grassy, G. A genetic algorithm for the automated generation of small organic molecules: Drug design using an evolutionary algorithm. *J. Comput.-Aided Mol. Des.* **2000**, *14*, 449-466.

[17] Pegg, S.C.; Haresco, J.; Kuntz, I. A genetic algorithm for structure-based de novo design. *J. Comput.-Aided Mol. Des.* **2001**, *15*, 911-933.

[18] Vinkers, H.; De Jonge, M.; Daeyaert, F.; Heeres, J.; Koymans, L.; Van Lenthe, J.; Lewi, P.; Timmerman, H.; Van Aken, K.; Janssen, P. SYNOPSIS: SYNthesize and OPtimize System in Silico. *J. Med. Chem.* **2003**, *46*, 2765-2773.

[19] Schneider, G.; Lee, M.; Stahl, M.; Schneider, P. De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks. *J. Comput.-Aided Mol. Des.* **2000**, *14*, 487-494.

[20] Takaoka, Y.; Endo, Y.; Yamanobe, S.; Kakinuma, H.; Okubo, T.; Shimazaki, Y.; Ota, T.; Sumiya, S.; Yoshikawa, K. Development of a method for evaluation of drug-likeness and ease of synthesis using a data set in which compounds are assigned scores based on chemists' intuition. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 1269-1275.

[21] Lajiness, M.S.; Maggiora, G.M.; Shanmugasundaram, V. Assessment of consistency of medicinal chemists in reviewing sets of compounds. J. Med. Chem. **2004**, *47*, 4891-4896.

[22] Congreve, M.; Carr, R.; Murray, C.; Jhoti, H. A "rule of three" for fragment-based lead discovery? *Drug Discovery Today* **2003**, *8*, 876-877.

[23] Oprea, T.I. Is there a difference between leads and drugs? A historical perspective. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 1308–1315.

[24] Bemis, G.W.; Murcko, M.A. The properties of known drugs. 1. Molecular frameworks. *J. Med. Chem.* **1996**, *39*, 2887-2893.

[25] Bemis, G.W.; Murcko, M.A. Properties of known drugs. 2. Side Chains. *J. Med. Chem.* **1999**, *42*, 5095-5099.

[26] Ertl, P.; Rohde, B.; Selzer, P. Fast Calculation of Molecular Polar Surface Area as a Sum of Fragment-Based Contributions and Its Application to the Prediction of Drug Transport Properties. *J. Med. Chem.*, **2000**, *43*, 3714-3717.

8 Conclusions and Future Perspectives

Conclusions

In this project, we have developed a computer program for *de novo* molecule design, the Molecule Evoluator. It is unique among programs for *de* novo molecule design since it combines three features: an atom-based approach, an evolutionary algorithm that can optimize structures, and its interactivity which allows it to profit from the knowledge, intuition and creativity of its user.

The first feature, the atom-based approach, helps the Molecule Evoluator to search all of chemical space, and fine-tune the structures. When using the much more common fragment-based approach, one first faces the problem of whether all synthetically possible and drug-like fragments have been included, and second there is the question whether the reconnection algorithm uses all realistic possibilities of synthesis. Covering chemical space with the atom-based approach is much easier and more natural. Secondly, our exhaustive repertoire of atom-based mutations (several of which lack in other work) allows the molecules to change gradually and adapt themselves to their target, instead of making big jumps in structure which usually result in large loss of fitness and may tend to force the population into premature convergence.

The second defining feature of the Molecule Evoluator is the evolutionary algorithm on which it is based. While several optimization methods exist (like random search, simulated annealing, or just molecular growth), evolutionary algorithms make good use of two features of molecular space. First, that molecules close in structure generally have related biological activity; evolution's concept of heredity, of inheriting good genes from the parents, makes methods which base the new molecules on the previous ones (instead of searching randomly) a good choice. Secondly, both in literature and in our own experiments we found that perhaps the most defining feature of evolutionary algorithms, namely crossing different solutions, is also advantageous and viable in drug design. Evolutionary algorithms not only use heredity, but also crossover, and therefore automatically use the structure of biological activity space to their advantage.

The third, and perhaps most distinguishing feature of the Molecule Evoluator is its interactiveness. While interactive evolutionary computing (IEC) has been used for quite some time in diverse applications, it had not yet been used for molecule evolution. However, there are good reasons to use interactive evolution in this field. First, there is a lack of good "fitness functions" – methods to calculate "how good" a molecule really is. Programs to estimate ligand binding energy and ease of synthesis are still very unreliable, so evolution without any human intervention will rarely yield good structures. However, human domain knowledge on ease-of-synthesis and pattern-recognition may help. Second, it can be difficult for people to accept the computer "prescribing" molecules out of the blue: most chemists would either like a good reason or some input of their own. The limitations of current automatic structure evaluation would lead to rejection of flawed structures, instead of correction. Third, evolutionary algorithms do not only "exploit" existing knowledge, they also explore new possibilities. Since a computer can quickly generate many possibilities and has other prejudices than a human chemist, interactive evolution can be a valuable idea-generating machine to complement human creativity.

During the development of the Molecule Evoluator, we discovered that the first versions needed to be refined to be acceptable to the chemists using them. In the following paragraphs we describe some of the problems encountered and the modifications implemented in response.

The main problem of the first version of the Molecule Evoluator was that many structures just did not seem possible to synthesize at all. In particular, common substructures like phenyl were almost completely absent, weird substructures like peroxide were common, and many structures had overly complicated rings or disobeyed chemical rules of thumb, like Bredt's law which states that a bridgehead carbon atom cannot have a double bond (unless the rings have a certain minimum size).

To make the molecules more "appealing" and easier to synthesize, we first allowed the ME to not only add atoms to a growing molecule, but also a number of predefined fragments (carboxy, phenyl, cyanide, etc.). Secondly, we mined the NCI database to find the frequencies of the different atom types and 2/3/4-atom substructures. Based on the frequency of, for example, O-O in the NCI database we modified the chance that an oxygen atom would be connected to another oxygen atom. This automatically enforced chemical rules that state that peroxides are rare, using statistics instead of qualitative human intervention. Third, we implemented a couple of rules, such as Bredt's rule and a prohibition of CH_2-imines. These rules act like filters that prevent molecules that do not obey the given chemical constraints from being shown to the user. Mining the NCI also gave us a catalog of ring structures (see the "Chemical Clichés" chapter) which were also implemented to filter unknown and probably strained rings out. Note that the filters can be activated and deactivated by the user, and that the full idea generation potential remains available if desired.

The second main point for improvement was user control. For example, chemists often had certain ideas about which part of a molecule was important, and should be conserved. Also they preferred that the best molecule from the previous generation was to be saved always (which is not guaranteed in a normal evolutionary algorithm). Thirdly, occasionally a chemist could see an obvious modification of an existing molecule, and wanted to put the adapted molecule in the Molecule Evoluator. And finally, the molecules produced should preferably be drug-like and obey a number of physicochemical restraints.

We made various adaptations to address these points. We added atom and bond fixation, which can conserve any atom (even hydrogen atoms) and can therefore focus evolution on the variable part of the molecule. Secondly, we added elitism, so the chosen molecules of the previous generation appear as the first molecules of the current generation, allowing a chemist to easily see if their offspring improves over them. Thirdly, we added a molecule editing window, which allows the user to adapt ME-generated molecules and feed them back into the evolution, or even to sketch new molecules as input of the evolution. Lastly, we added physicochemical filters to allow the user to determine the allowable physicochemical properties of a molecule, such as the range in which the molecular weight should fall, and the maximum permissible number of hydrogen bond donors.

Finally, we tested the Molecule Evoluator to examine whether the concept was sound and useful.

First we ran a number of small experiments, in which we evolved certain drug molecules from scratch (that is, without editing the molecules, though we fixed certain atoms/bonds to accelerate optimization in a certain direction) to show that we can indeed convert simple substances into drugs. We also were able to reproduce, using experimental fitnesses, an optimization pathway of neuramidase inhibitors, which shows that our mutations and selection work well with good (experimental) fitness measures.

The value of the idea-generating function was tested by creating 300 random molecules that obeyed certain physicochemical restraints. A panel of chemists chose the most “drug-like” 34 structures, of which a number did not appear as structure or substructure in existing databases, and therefore could be considered potential new molecule templates. Synthesis of eight of the compounds yielded four compounds which showed biological activity in the used essays. It seems that chemists indeed have a valuable intuition, and that the ME can inspire the synthesis of truly new classes of molecules, unknown before yet possible to synthesize.

The Molecule Evoluator has become more advanced over the years, and is at the time of writing (May 2008) commercially available. In the Leiden group of Medicinal Chemistry, where it has been developed, it is now used in each synthesis project as an interactive, idea-generating but responsive aid to get new ideas for structure modifications. Several companies have bought versions, and some are quite happy with it, as is evident from the following quote:

"Both computational chemists and medicinal chemists have explored the Molecule Evoluator and have been excited about the output in terms [of] novel ideas being generated and the potential for further enhancements in the future. The real advantage of the current programme is that it can be iteratively influenced by trained chemists to propose new structures, some of which may look immediately obvious but yet had not been previously suggested. Three of our current GPCR-based projects have benefited in this way."

Software that allows humans and computers to combine their particular strengths is still rare, for drug design the Molecule Evoluator is the only one to our knowledge which is currently commercial. This has two likely causes. First of all, interactive evolution is itself a young field, most programs for designers (of molecules or buildings) are drawing programs which do not give any creative input of their own, as they were designed to be computerized replacements for real drawing boards. Second, most complex software for molecule design has been created for computational chemists and therefore only runs under Unix/Linux workstations and has powerful but complex interfaces. The medicinal chemists, the people who design and modify most of the molecules and are experts on molecules rather than computers, have been left out. Only in the last two or three years software companies are also starting to develop versions for Windows and thus for the “normal” chemists. (See for example the

Software section in the biweekly ‘Chemical and Engineering News’ of the American Chemical Society). But even when a Windows version is available, it will be a long road for most programs to also become user-friendly for people who are not experts in computational chemistry. As of yet, the Molecule Evoluator is quite lonely in the software landscape, but we hope that in the coming years it will be joined by followers and colleagues which bring both the computational and creative potential of computers to the medicinal chemists directly. Software has a vast potential for changing drug design, if we invest effort and creativity in it.

Future Perspectives

> *Prediction is very difficult, especially about the future.*
> Niels Bohr (1885 – 1962)

Science is never finished. The research described in this thesis may have produced a useful computational tool for the medicinal chemist, and it may have given more insight into chemical databases. However, we nor others have as of yet produced the perfect medicinal chemistry tool, and many problems in drug design remain. The previous chapters of this thesis have covered what we have done in our research. This last part will contain reflections on where to go from here. I hope this chapter will provide ideas and inspiration to researchers and non-researchers alike on what subjects in computational drug design would be worthwhile to investigate, and in which directions we could go.

I will begin with some thoughts on future directions for the Molecule Evoluator, then discuss the possible evolution of evolutionary algorithms in drug design, and will end by zooming out to look at the general role of software in drug development, and talk about some of the ways in which we can increase the ability of software to help us design new drugs.

The future of the Molecule Evoluator

At the time of this writing, we have performed experiments which have shown that the Molecule Evoluator can at the very least help find novel biologically active molecules. We may never know if chemists without the Molecule Evoluator would have been as creative as chemists using the Molecule Evoluator, but it is very likely that computer-generated structures can complement the brainstorming by chemists, which may mainly design variations on the molecules they already know. For that reason, the Molecule Evoluator as it is now may remain useful for a long time to come.

To enhance the usefulness of the Molecule Evoluator further, there are numerous possibilities: improving the speed at which molecules are generated, comparing the effects of different crossover functions, making the user interface even more intuitive, improving the display so that the user can see very quickly which mutations have been generated, offering calculations of more physicochemical properties (for example pK_a) or linking the Molecule Evoluator to third-party software that can calculate those properties, and numerous other tweaks and enhancements. At the moment there are however three points which I think most promising for future updates: changing the structure generation to yield even better structures, linking the Molecule Evoluator to other software and databases, and experimenting with computational fitness evaluations.

Ease of synthesis: less boring, less impossible, more novel?

From conversations with users, we found that the main factor determining how much they liked the Molecule Evoluator was the ratio of “good” molecules to “bad” molecules. Good molecules are those molecules that are novel and seem relevant or at least can be easily changed into a molecule with a good structure. “Bad” molecules are those molecules which are boring (not very novel), irrelevant, or plainly impossible to synthesize.

The most significant way to enhance the use that chemists get out of the Molecule Evoluator would therefore be increasing the number of good molecules while preferably decreasing the number of bad molecules. With previous adaptations we have already succeeded partially in this, and it is certainly possible to further improve our results with some additional adaptations of the code.

At the moment the best method to improve the ratio of good to bad molecules seems to be to diminish the occurrence of the main types of bad molecules: the 'impossible' molecules, the 'irrelevant' molecules, and the 'boring' molecules.

The 'impossible' molecules are those molecules which cannot be synthesized. We have already reduced their number with chemical filters and giving the user the option to only allow known ring structures; further feedback will undoubtedly allow us to increase the number of filters that can be applied to a molecule. The 'irrelevant' molecules are only created when the Molecule Evoluator cannot find a mutation that works, which mostly occurs in molecules where many atoms or bonds have been 'fixed' by the user. The solution to this is to rewrite the mutation algorithm: at the moment it picks a random atom from the molecule to mutate, which fails if that atom has been fixed by the user – forcing the Evoluator to create a random/irrelevant molecule instead. Rewriting the mutation algorithm so it picks only from the atoms which are *not* fixed will give a much greater mutation success rate and therefore a much lower production of irrelevant molecules.

The final way to decrease the amount of “bad” molecules is to tackle the boring molecules (this is for interactive evolution. An automatic fitness function cannot be bored). A chemist may find certain mutations boring or “not novel”. The aim therefore is to find out which mutations are generally found interesting, and which are not. Finding out these preferred or unpopular mutations can be done by either directly observing a user or by creating special statistical subroutines to observe which mutations (atom addition, deletion, insertion) produce molecules that are most often selected or not selected, and what kind of additions/deletions/insertions are most interesting. Such an investigation might for example find that adding a methyl group to a benzene ring is “boring”, while adding a hydroxy group to the same ring is “interesting”. The probabilities of those specific mutations can then be adjusted appropriately.

In conclusion, adapting the Molecule Evoluator to change the ratio of good to bad molecules in a beneficial way is certainly possible with user observation and feedback and some reprogramming. Of all the possible options to improve the Evoluator, this optimization may have the strongest impact on user-friendliness and frequency of use, and would therefore be a prime target for implementation.

Linking the Molecule Evoluator to databases

A second area for improving the Molecule Evoluator turned up during our own tests. While trying to find novel biologically active molecules, every compound the chemists found interesting had to be manually looked up in the CAS database. While this database search was by far not as much work as eventually went into synthesizing the truly novel compounds, it taught us that it would be incredibly handy if one could look up the Molecule Evoluator-generated structure or similar structures in the user's favourite chemical databases by just pressing a button. A useful improvement would therefore be a link to databases that would allow chemists to find whether the molecule suggested by the Molecule Evoluator exists in its entirety or as a substructure, and if it exists, how it can be synthesized (or ordered). For large and wealthy institutions, links to commercial databases like Beilstein and CAS may be possible, but more exciting is the opportunity brought by the advent of large public databases like PubMed, PubChem, eMolecules and ChemSpider to give all users of the Molecule Evoluator the chance to have structures automatically checked with literature.

Databases could not only be used for checking structures, but can also help to create structures. If, for example, a chemist is looking for alternatives for a certain ring in a molecule, it would be very useful if he could view a list of the most common ring systems from our 'chemical cliché' database next to the molecule editing window, as that could give many ideas for changes. A similar approach could be taken for substituents, where the chemical clichés fragment database or a specialized database like a bioisosters database could be used to find replacements that the chemist might not have thought of yet.

The Molecule Evoluator and automated evolution

Interactive evolution has some major advantages over automated evolution, mainly that it can use expert knowledge much more easily than any fitness function designed by computer programmers. However, this requirement for expensive expert time is also a disadvantage, and given the successes of the interactive mode of the Molecule Evoluator, one could consider adding options for computational chemists who want to use automated evolution.

While adding a feature for automated evolution is possible (in fact, it has already been done in one or two individual cases), to make the automated evolution perform optimally one needs to change more than the code for the fitness function. Automatic evolution and interactive evolution, despite their apparent similarity in approach, are as dissimilar (if not more dissimilar) as tennis and table tennis.

The first dissimilarity of automated versus interactive evolution is the absence of user fatigue. This opens up the desirable possibility to create larger generations than the 12-20 which are practicable for user feedback (50-100 molecules seems to be about the optimum size if we consider investigations such as that of Douguet et al.[1], since it may avoid the premature dead ends which endanger small populations and the 'drowning out' of the good genes in very large populations. Larger populations seem to work better when split into 'islands'). Also, automated evolution doesn't need settings that prevent molecules that differ only slighly from their ancestor being created, as a small increase in fitness is useful, whereas such a molecule would strike a human as uncreative and increase user fatigue.

The absence of user selection, however, also has some disadvantages. First of all, automatic evolution makes it necessary to implement a selection function: for if the user isn't selecting the “good” molecules, the programmer has to decide how to select the best molecules for further evolution. Take the best five molecules? Or ten molecules? Use tournament selection? Roulette wheel selection? A second disadvantage, which we discovered during the docking experiment (Chapter 6), is that atom-based evolution when not supervised by humans tends to produce molecules which over time become more and more difficult to synthesize. Therefore, automated evolution needs stricter filters to sufficiently dispose of unwanted structures.

In summary, unlocking the full potential of automated evolution requires changing more parts of the Evoluator than the fitness function – the automatic evolution would need to be at least partially split from the code for interactive evolution. On the positive side, automated evolution also offers opportunities. Performing automated evolution with a good fitness function may give us a better idea what parameter settings and what mutations or crossover operations are optimal for drug design, and allow us to adjust the Molecule Evoluator accordingly. If we keep studying drug design and evolutionary strategies for drug design, the Molecule Evoluator may one day not only be the best interactive evolutionary algorithm for drug design, but also the best evolution-based program for automatic drug design available.

General perspectives on evolutionary algorithms in drug design

Useful as they are, evolutionary algorithms aren't the “cute new kid” in computational drug design anymore. They were immensely popular in the late 1990's, but interest waned as it grew more and more difficult to think of yet-unpublished applications and it turned out that evolutionary algorithms, like all methods before them, were not the “cure-all, one-size-fits-all”-solution that drug designers have been seeking for so long. I have discussed my view of the future of Evolutionary algorithms in 2005 in my review on evolutionary algorithms in drug design (Chapter 2), in which I discussed various possible developments such as creating standardized test databases for chemical problems and evaluating newer types of evolutionary algorithms. Currently, evolutionary algorithms are already unobtrusively integrated as standard tools for experiments, for example for descriptor selection in a virtual screening experiment[2]. For new problems too, evolutionary algorithms are becoming easier to try out as flexible optimization methods, due to the development of EA toolkits such as the GA Playground, OAT, Lil-gp, and ECJ[3]. However, in my mind two ideas seem most important: adding further domain knowledge to evolutionary algorithms, and the use of evolutionary algorithms in novel or at least uncommon ways in drug design, such as modelling and data mining.

Adding knowledge to evolutionary algorithms

The main bottleneck in successful application of evolutionary algorithms in drug design is that finding the best solution (or even a very good solution) often takes more computer time than is available. This is both because of the “high dimensionality” of many drug design problems (many parameters need be optimized simultaneously) and because fitness functions often take much time to be calculated. While the increasing computer speeds may help here, it may even be more important to perform the optimization itself in more intelligent ways.

Evolutionary algorithms can be improved by testing and comparing different models (differing in population size, selection pressure, etcetera), but more important will be the collaboration between computer scientists and experts in the problem domain, such as drug design. Currently, relatively simple evolutionary algorithms are often used since they shorten programming time – unfortunately, these same algorithms are afterwards too easily carried over into the commercial version where the 'saving' of programming time is paid back with interest as the inefficient algorithm is run thousands of times. Only with knowledge of the problem itself can one develop rough fitness functions which can eliminate patently bad candidates before they are subjected to a more accurate fitness calculation, create meaningful mutations that turn good solutions into other good solutions instead of impossible ones (for example, producing a carbon atom with five bonds), split the solutions as much as possible in semi-independent sub-systems for faster optimization, and make optimal use of the knowledge obtained by the evolution so far (for example, learning that a certain type of atom at a particular position produces very good scores). It may be difficult for computer scientists and drug designers to understand each other and communicate one's knowledge and aims clearly to the other party, but in the end the algorithms that will survive and turn out to be most useful in a given problem domain will not be the newest generic computer science methods, but well-chosen basic methods, carefully optimized to suit the problem at hand.

Future applications of evolutionary algorithms

Evolutionary algorithms are currently experimentally or routinely used in most phases of drug discovery. The most important of their current applications are their contributions to the “core” business of finding new leads by computer, namely by library design, *de novo* design, and virtual screening. Of these, “virtual screening” (evaluation by computer) of a potential lead is hardest by far, since proper computational evaluation needs to answer six questions: 1) is the target important in the disease, 2) does the candidate molecule interact strongly enough with the target, 3) can the molecule get to the place of action, 4) how and how fast is the molecule metabolized, 5) are the molecule and/or its metabolites toxic, and if so, how much, and 6) is the molecule excreted slowly enough.

While docking tries to answer question 2, and Lipinski's rules and calculating the polar surface area of the molecule help us somewhat with 3 (barring active transport), reliably predicting any of these relevant properties requires good-enough predictive models, and even a much-researched subject such as docking can definitely be improved greatly yet. The main challenge of computational drug design is therefore to develop better models for the interaction of a molecule with the human body.

Creating models generally starts with collecting large amounts of raw data (for example, molecules and their intestinal absorption), calculating descriptors (properties of the input molecules, such as the weight or the polar surface area of each molecule) and picking a computational model (linear regression, neural network, support vector machine) that links those descriptors to the measured property. With the unavoidable measurement errors in experimental data, perfect mathemathical relationships are generally not possible, but evolutionary algorithms could help in parameter selection (as they have done for QSAR), and even with computational model selection. In some relatively simple cases this is already possible, such as evolutionary algorithms producing mathematical equations out of pictures of moving systems[4]. In the end, evolutionary algorithms could become more independent and work on more complex problems, becoming untiring generators and testers of hypotheses. Even more than today, future evolutionary algorithms may supplement human brain power in making better models to predict how a particular molecule will fare as a drug.

The future of computational medicinal chemistry

While medicinal chemistry changes, its basic challenges stay the same. People will continue to have diseases and want to get rid of them, and unless genetic modification of living humans becomes easy, reliable and safe (which seems very unlikely to happen this century) we will in most cases need to fight the diseases with drug molecules. These will remain difficult to find since we do not always know the mechanism of the disease or the best protein to target, and even if we know those we may struggle to find the molecule that interacts effectively with that target, can get to the place of action, and does not have unacceptable side effects or toxicity.

Most current sub-fields of computational medicinal chemistry, such as docking and prediction of ease of synthesis will probably grow and improve over time, though that is likely to be a slow and laborious process. The algorithms may never be perfect, but they may become so good as to be too useful to ignore. The three areas which interest me most, however, are still much earlier in their development: automated data analysis, simulations and interactive software.

Automated data analysis

In theory, knowing the DNA encoding for a protein should be enough to calculate the three-dimensional structure of the protein and find molecules which bind to it, by applying quantum mechanics and molecular dynamics. Similarly, an optimum-yield synthetic route for a new molecule could be found by using retrosynthesis and quantum mechanical calculations on a library of available reagents. In practice, however, we need lots of data both as primary input (you can't understand an organism without knowing at least its entire DNA), and as a substitute for calculations which would be theoretically possible but far too computationally intensive to be practical. If you would need either 50 years of computer time to correctly calculate the affinity of a lead compound to a protein using quantum mechanics and thermodynamics, or one hour of biochemical testing, the latter is far preferable. Therefore drug design still needs lots of experimental data to be efficient.

These experimental data are increasingly becoming available through scientific journals creating online versions, electronic lab journals collecting the primary data generated by researchers, and the growth of public databases such as PubChem and Wikipedia. This increase of data is promising, but current search methods have difficulties exploiting it: it can be incredibly hard to find the particular piece of data one is looking for. For example, in my stints as a science journalist I have spent much time being frustrated and giving up searches when Google couldn't locate a proper answer to even elementary scientific questions such as which hormones affect the release of GnRH from the hypothalamus. This information undoubtedly exists in reviews and/or books, and a more elaborate searching through reviews could probably uncover it, but still the (time) cost of finding particular information is much higher than would be technologically necessary. Making information easier to find can reduce both the time and the cost of any research project.

A large part of making better use of existing data will be economical and organizational: somebody has to pay for the servers to house the data and the transformation of (scientific) texts and tables into a computer-readable format; in many cases the 'owner' of the data (such as a scientific journal) will also want financial compensation. But what are the technological aspects?

The first part of technology will be disambiguation and transforming text into something the computer can more easily relate to search queries; for example a text like “the melting point of benzene is +5.5 °C” would fit better in a computer database as “object=”benzene”::==reference=”www.merckindex.com/benzene”=>property = “melting point”=>value=”278.5”=>unit=”K”. Ontologies/synonym lists and lists of standard terminology will remain necessary to transform raw data into something that is as standardized and unambiguous as programming language statements (for example, instead of “melting point” my version of the Merck index occasionally uses the less standard “solidif”).

The second and probably greater challenge will be to make search methods less “fussy” and more “fuzzy”. For example, when designing a synthesis route for a compound, it may be that the compound has never been synthesized before, or that it has been synthesized, but via a rather inefficient route. Finding the synthesis route for similar compounds can help in both kinds of cases. However, despite Tanimoto coeffients and fingerprint calculations, it is unlikely that there is an universal measure of molecular similarity (similar molecular weight? Similar size? Similar melting point? Similar biological effects?). Domain experts and ideally data mining should be able to discover what the similarities are in compounds synthesized via the same route, for example, re-discovering that alcohols can be made out of available halo-alkanes using a Grignard-reaction, but also in which cases Grignard fails or is not used at all. This will be a complex piece of data mining where programmers and domain experts will have to work together, but having a system that can learn from existing data and adapt its knowledge when encountering new information would be extremely useful to build.

There are already several individuals and groups striving to improve data management in science from the 'let's do the same as always, only electronically' stage to transforming global scientific knowledge into a truly useful search engine, for example the Scientific Publishing Taskforce which is developing methods to make scientific publications computer-readable (http://esw.w3.org/topic/HCLS/Scientific PublishingTaskForce), attempts to unify data from many scientific sources into a single encyclopedia/wiki (http://www.wikipathways.org/index.php/WikiPathways) and a more fuzzy version of PubMed, eTBlast (http://invention.swmed.edu/etblast/ index.shtml). While the diversity of initiatives indicates that this field is far from ripe yet and standards need to be developed to unify the different projects, going into the direction of converting data into standardized formats and making search engines more intelligent will in my opinion greatly improve data availability and increase literature-search efficiency, and drug design can only profit from that.

Simulation

The paragraph on future applications of evolutionary algorithms focused on model development. Models, however, are not limited to linear regression models or neural network models or decision trees only. One of the most interesting directions that modeling is going is the simulation of cells, tissues and even of entire human bodies. Models for the heart are already available, as are models for blood flow and some cancers[5]. Evolutionary algorithms already help find good parameters for conventional models (such as neural networks), likewise they may help fine-tune simulations by supplying good values for missing parameters. All in all, though, evolutionary algorithms would only be a small part of the total work on simulations – many computational methods and algorithms are likely to be necessary, from logic programming and cellular automata for rough qualitative models to differential equations working on small simulated 'boxes' of cellular cytoplasm and cell membrane for the most advanced models.

While it may be difficult to create good simulations, it would be amazingly useful if we would have more reliable models for what happens in the various tissues in case of disease, and what the effects of various interventions would likely be. In contrast to biological experiments, it is very easy to knock a gene out in the computer or let a protein be deactivated fully without the need to discover a strong and selective protein antagonist first. Even better, good computer models could act as “living encyclopedias” of current biological knowledge, linking the many findings of biological research into one easily accessible and correctly interconnected whole, growing more and more accurate the more we learn about biology. Creating excellent simulations of biological systems may be one of the most difficult projects that could support drug design, but its huge advantages in both allowing us to test compounds reliably without endangering human volunteers and understanding the true mechanisms and complexities of disease will also probably make it the most useful of all possible software if we succeed.

Interactivity

On the surface, interactive evolution may look like nothing more than a speeding up of the usual process of the medicinal chemist stating the target and the constraints to the computational chemist, who programs these into the computer, which produces new molecules. It is indeed an advantage that interactive evolution can shorten and speed up this cycle, but the truly qualitative difference is that interactive evolution can use one resource which automated evolution can not take advantage of: the subconscious knowledge and expertise of the medicinal chemists.

As I stated in the introduction of this thesis, major progress could be made by letting people and computers work together more productively, complementing each other's strengths. While delegating simple tasks to computers has of course been done since the dawn of computing, for example by humans creating the formulas for the spreadsheet and the computer doing the calculations, only more recently computer programmers have deliberately started to try use human brain power. Interactive evolutionary algorithms are one example of this, another is the program C3vision which makes an EEG of the user's brain activity while (s)he is quickly browsing images, and can detect when the user sees something interesting much faster than the user him/herself realizes it, thereby speeding up scanning of images tenfold[6]. The human processing capabilities can also be tapped via games, for example 'fold-it' (http://fold.it) which lets humans predict protein folding quite successfully, as predictions by fold-it players won seven prizes at the protein structure prediction contest CASP8, outperforming purely computational methods, and in one case even outperforming professional scientists.

What will the future be of such interactivity between man and computer? For now, humans seem to excel in combinatorial problems (such as finding the right folding for a protein, where calculating the energy score is probably rather fast) and problems which require knowledge which is hard to make explicit ('hunches'). Docking could undoubtedly be performed similarly to fold-it, at least when the docking scoring functions have become more accurate. And perhaps humans could be trained to get toxicity “intuition” by showing different structures and making them guess whether a compound is mutagenic or not, perhaps recognizing patterns which may be difficult to find by computer.

It may be that human-computer interactivity will one day be surpassed by computers which either can process data in a human-like way or are so fast that they won't need the speedup provided by human knowledge anymore. But such advances are only likely to happen in the very far future, if at all. For now, the more we learn about computers and problems, the more we find that computers are not yet the answer to all our problems and that human experts are unexpectedly potent problem solvers. Making human-computer interaction work well is a field of research in itself, but the better we become at it, the more powerful our capabilities in science and drug research will become, leaving 'computer-only' or 'human-only' techniques far behind.

In closing

Computational drug design has come far since the first QSAR programs and molecule databases, and the Molecule Evoluator is certainly not the end point of its ongoing evolution. There have been quite some successes already (for example with virtual screening), but we can still do better. In my opinion, software developers should strive to make drug design software grow in three directions: deeper, wider, and closer.

Deeper software will be software that more elaborately uses core scientific knowledge such as quantum mechanics, molecular dynamics and thermodynamics to improve the accuracy of predicting ligand affinity, compound metabolism and other phenomena. The main challenges here are integrating the diverse formulas and principles of chemistry and physics (for example, ligand affinity prediction also needs accurate prediction of the energy of (de)solvation of the ligand), and speeding up the calculations so that accurate enough results are obtained in the computer time available. *Wider* software, which focuses on combining and comparing data, may be able to help us where calculations are yet too slow – by recognizing patterns in experiments we may in some cases be able to predict important properties from 'data-based' models where 'computation-based' models are as of yet too time-consuming. Software casting a wider net over our current scientific knowledge may also help us find connections between different subjects, by integrating the data on chemistry, biology and genetics into one organized whole. Finally, developing *closer* software means that we should strive to make software more accessible to 'lay' users, so it can be used more easily by scientists who are not experts in computer programming. For some software products, this would involve creating Windows versions, for almost all programmers it would mean focusing much more on user-friendliness and the user's goals. Consciously seeking to develop the most intelligent task division between computer and user will also greatly increase the user's power to tackle scientific and drug design problems.

Developing deeper, wider and closer software may never bring 'perfect' computational drug design as was perhaps once envisioned. It is quite likely that it will create computer programs which will be totally unexpected, and tackle problems we may not even know about yet. Only one thing is certain: by working steadfastly on finding new ideas and evolving our software, we will be able to increase our understanding of biology and drug design to a depth we would now deem incredible. May that understanding be used effectively for the progress of science, and for the health of humanity.

References

[1] Douguet, D.; Thoreau, E.; Grassy, G. A genetic algorithm for the automated generation of small organic molecules: Drug design using an evolutionary algorithm. *Journal of Computer-Aided Molecular Design* **2000**, *14*, 449-466.

[2] Barreiro, G.; Guimarães, C. R. W.; Tubert-Brohman, I.; Lyons, T. M.; Tirado-Rives, J.; Jorgensen, W. L. Search for Non-Nucleoside Inhibitors of HIV-1 Reverse Transcriptase Using Chemical Similarity, Molecular Docking, and MM-GB/SA Scoring. *J. Chem. Inf. Model.* **2007**, *47*, 2416-2428.

[3] Wilson, G. C.; McIntyre, A.; Heywood, M. I. Resource Review: Three Open Source Systems for Evolving Programs-Lilgp, ECJ and Grammatical Evolution. *Genetic Programming and Evolvable Machines* **2004**, *5*, 103-105.

[4] Schmidt, M.; Lipson, H. Distilling Free-Form Natural Laws from Experimental Data. *Science* **2009**, *324*, 81-85.

[5] Gavaghan, D.; Garny, A.; Maini, P. K.; Kohl, P. Mathematical models in physiology. *Phil. Trans. R. Soc. A* **2006***, 364,* 1099-1106.

[6] Sandhana, L. Man and machine vision in perfect harmony. *New Scientist* **2006**, *2559*, p30.

Summary

Designing drugs is hard. For many illnesses there are no effective drugs available (for example, for multiple sclerosis), in other cases existing drugs are not always effective enough (cancer). And even when effective treatment is available (for example for HIV), the drugs may have side effects that investigators and patients alike want to diminish. Chapter 1 introduces this thesis, and explains why it is so hard to design drugs. It also discusses how computers could help in the drug design process, as well as the advantages of effective collaboration between human drug designers and computers. Chapter 1 also explains some of the terms that are used throughout this thesis, such as data mining and docking, and it closes with an overview of the rest of the chapters.

Chapter 2 discusses one of the core methods used in our investigations: evolutionary algorithms. It describes the basic theory of evolutionary algorithms, and gives an overview of the different uses of evolutionary algorithms in drug design, varying from library design to QSAR and docking. It then compares these applications of evolutionary algorithms, and concludes with suggestions for future applications and further research.

The other main computational method we used, data mining, is discussed in Chapter 3. In data mining, one attempts to find patterns or rules in databases. In this chapter, we investigated a database of molecules, the NCI database, which, at the time of our investigation, contained 250,251 molecules. We were especially interested in how diverse this database was, since having a collection of diverse molecular structures leads to a greater chance of finding potential new drug molecules. We split all molecules in the NCI database into ring fragments and non-ring fragments, and determined the frequency of these fragments, as well as the frequency with which fragments occurred together in molecules. The molecules in the NCI database turned out to be remarkably undiverse: while there were over 10,000 different ring structures, the ten most frequent ring systems (<0,1% of the total ring diversity) accounted for 62% of all the ring systems found in the molecules. On the other hand, thousands of ring systems occurred in only one or two molecules. We also found that many substructures occurred much more often with each other than would be expected by chance, forming 'superfragments', and, conversely, that there were also substructures which seemed to avoid each other. The data mining thus yielded lists of rare fragments and fragment combinations, which could be used to design new molecules. Such new molecules, when added to existing libraries, would increase its molecular diversity and thereby increase the chance of finding new drugs.

In Chapter 4 we return to evolutionary algorithms, in particular to the challenge of designing new drug molecules. When we started our study, there had been many investigations using evolutionary algorithms to design candidate drug molecules. However, all these approaches suffered from the problem that it is not yet possible to calculate a suitable fitness function, namely how good a certain molecule would be as a drug. This lack of quality fitness functions made optimization unreliable at best and meaningless at worst. Therefore, we decided to use a different approach, that of an interactive evolutionary algorithm. Interactive evolutionary algorithms use feedback from the user as fitness criterion; so in our case, chemists had to select the molecules they liked (which means, the molecules that seemed novel, drug-like and possible to synthesize without too many problems). In this way we could use the chemists' experience and chemical intuition, which would have been very difficult to 'extract' from their brains and program a computer with. We called the resulting interactive evolutionary algorithm the 'Molecule Evoluator'. Chapter 4 describes the design of the Molecule Evoluator, as well as the coding of the molecules, the different mutations, and the extra features that were needed to make the program usable for chemists and allow optimal use of their creativity.

One of the problems in the first versions of the Molecule Evoluator was that it created many molecules which either were unstable or would be difficult to synthesize. This annoyed chemists, leading them to abandon the optimization process prematurely. To increase the chance that molecules created by the Molecule Evoluator would be chemically sensible, we applied the technique discussed in Chapter 3, data mining. We mined the World Drug Index, a database that, at the time of our investigation, contained the structures of 32,000 drug molecules. In this collection of drug molecules, we determined the relative frequencies of small substructures containing one to four atoms, and used these data to adapt some of the Molecule Evoluator's mutations. We found that this indeed increased the chance that the Molecule Evoluator produced realistic molecules, while at the same time not forbidding substructures which actually occur in drugs. The exact procedures of the data mining, the adaptation of the mutation functions and the methods used to test whether the new mutations improved over the old ones are described in Chapter 5.

Our literature study for Chapter 2 encountered many evolutionary algorithms for designing drug molecules. However, it was difficult to compare the quality and effectiveness of these methods because they had not been compared to each other. This is unfortunate, since proper validation of molecules, either experimentally or by advanced computational methods, costs significant amounts of time and money.

Therefore it is important to make one's evolutionary algorithm as efficient as possible: an evolutionary algorithm which needs to evaluate only 1000 molecules to reach a certain improvement in biological activity would be much preferable over an evolutionary algorithm that needs 100,000 to reach a similar result. But what kind of evolutionary algorithm would be most efficient? Chapter 6 describes our investigation into the importance of one of the main factors in evolutionary algorithms for molecule design, namely whether evolution is atom-based (molecules are changed by adding, changing or removing single atoms), or fragment-based (where mutations add, change or remove multi-atom fragments). We evaluated the molecules by docking them into the reverse transcriptase enzyme of HIV. Docking was chosen as a fitness function here since it approximated the binding of molecules to a real protein with its irregular, three-dimensional cavity. While the experiment did not show clear superiority of either atom-based evolution or fragment-based evolution, a closer study of the optimization processes yielded several insights on how we could improve our evolutionary algorithms. The details of the experiment, as well as the suggestions for future improvement of evolutionary algorithms for drug design, can be found in Chapter 6.

In theory, the Molecule Evoluator we described in Chapter 4 should be able to help chemists design new, biologically active molecules. However, does this also happen in practice? Chapter 7 describes an experiment in which we let the Molecule Evoluator generate 300 structures, of which chemists chose 34 for further investigation. By consulting chemical databases, we determined which of those molecules were novel. Subsequently, the chemists chose a number of the novel molecules and synthesized eight of them. These eight compounds were tested on 83 proteins, and four turned out to be biologically active in some way, amongst others on the α-adrenergic receptor. This result indicated that the collaboration between chemist and computer using the Molecule Evoluator can indeed create novel and biologically active compounds, which may be good leads for new drugs. Chapter 7 gives the detailed experimental protocol, the structures of the molecules synthesized, and the biological test results.

This thesis concludes with Chapter 8. The first part of this chapter contains the general conclusions of the investigations described in this thesis. The second part of Chapter 8 is dedicated to future perspectives: thoughts on the further development of the Molecule Evoluator and on the future role of evolutionary algorithms in drug design. It concludes with my vision on the directions that software for drug design could or should take in the future.

Samenvatting

Het ontwerpen van geneesmiddelen is moeilijk. Voor vele ziektes (zoals multiple sclerose) bestaan er nog geen geneesmiddelen, zijn de huidige geneesmiddelen onvoldoende om alle gevallen te genezen (kanker), of hebben bestaande geneesmiddelen bijwerkingen die we zouden willen verminderen (bijvoorbeeld bij de behandeling van HIV). Hoofdstuk 1 leidt dit proefschrift in en bespreekt waarom het moeilijk is geneesmiddelen te ontwerpen, hoe computers ons zouden kunnen helpen bij het ontwerp van geneesmiddelen, en het belang van goede samenwerking tussen computer en geneesmiddelonderzoeker. Ook bespreekt hoofstuk 1 enkele kernbegrippen die in de rest van dit proefschrift terugkomen, zoals *datamining* en *docking*. Tenslotte bevat het een overzicht van de andere hoofdstukken.

De computermethode die we het meest gebruikt hebben, die van de evolutionaire algoritmen, wordt besproken in hoofdstuk 2. Hoofdstuk 2 geeft naast een inleiding van de theorie achter evolutionaire algoritmen een overzicht van de verschillende toepassingen van evolutionaire algoritmen in geneesmiddelontwerp. Deze toepassingen zijn onder te verdelen in het voorspellen van de eigenschappen van moleculen (zoals het afleiden van relaties tussen de structuur van een molecuul en zijn activiteit) en het ontwerpen van geheel nieuwe moleculen. De verschillende toepassingen die in de literatuur bekend zijn worden besproken en met elkaar vergeleken, alvorens mogelijke ontwikkelingen worden geschetst en suggesties worden gegeven voor verder onderzoek en mogelijke toekomstige toepassingen.

Hoofdstuk 3 beschrijft het onderzoek dat ik gedaan heb met een tweede computermethode, *datamining* ('data delven'). Bij datamining probeert men met behulp van een computer patronen te ontdekken in een grote database. Hoofdstuk 3 beschrijft het vinden en analyseren van patronen in de NCI database, een database van chemische verbindingen die ten tijde van ons onderzoek iets meer dan 250,000 moleculen bevatte. Van deze database wilden we de diversiteit bestuderen, omdat hoe diverser een set moleculen is, hoe groter de kans is dat het uittesten ervan een goed kandidaat-geneesmiddel oplevert. We splitsten alle moleculen op in ringen en niet-ring-structuren, en ontdekten dat de database enigszins 'saai' was. Hoewel er duizenden verschillende ringstructuren en niet-ringstructuren in voorkwamen, vormden de tien meest voorkomende ringstructuren, die nog geen 0,1% van de totale ringdiversiteit vertegenwoordigden, 62% van alle ringen, terwijl er duizenden ringstructuren waren die maar in één of twee moleculen werden gebruikt. Ook bleek dat bepaalde structuren veel vaker dan statistisch verwacht samen voorkwamen en 'superfragmenten' vormden.

Aan de andere kant leken andere structuren elkaar te 'vermijden'. Dit dataminen leverde lijsten op met zeldzame fragmenten en ongebruikelijke fragmentcombinaties, die door chemici gebruikt zouden kunnen worden om nieuwe moleculen te ontwerpen. Zulke nieuwe moleculen, toegevoegd aan de bestaande molecuulbibliotheken, zouden de kans op het vinden van nieuwe geneesmiddelen een stuk groter kunnen maken.

In hoofdstuk 4 kom ik terug op de andere methode die we gebruikt hebben: evolutionaire algoritmen. Hoewel er al vele evolutionaire algoritmen bestonden om nieuwe moleculen te ontwerpen (hoofdstuk 2 geeft daarvan een overzicht), hadden al die methoden het probleem dat wetenschappers (nog) niet goed kunnen berekenen hoe 'goed' een molecuul zou zijn als mogelijk geneesmiddel. Daarom besloten we de kwaliteit van de moleculen op een andere manier te bepalen, namelijk met een interactief evolutionair algoritme. Dat betekent dat de gemaakte moleculen niet door de computer worden beoordeeld, maar door chemici. Zo konden we de ervaring en chemische intuïtie van de chemici gebruiken, die anders moeilijk te achterhalen zijn. Het resulterende programma noemden we de “Molecule Evoluator”. Hoofdstuk 4 bespreekt de details van hoe we de moleculen codeerden, de mutaties waarmee de moleculen veranderd werden, en de aanpassingen die we moesten doen om het interactieve evolutionaire algoritme bruikbaar te maken voor chemici, en de chemici in staat te stellen hun creativiteit optimaal te gebruiken.

Een van de problemen in de beginfase van de Molecule Evoluator was dat het programma nog te vaak moleculen produceerde die moeilijk te synthetiseren waren, of zo instabiel waren dat ze uit elkaar zouden vallen. Om dat op te lossen hebben we de hoofdmethode uit hoofdstuk 3, datamining, toegepast. We gebruikten allereerst datamining om te achterhalen hoe vaak bepaalde groepen van 1, 2, 3 en 4 atomen voorkwamen in de World Drug Index (een geneesmiddelendatabase die ten tijde van ons dataminen ongeveer 32000 verbindingen bevatte), en gebruikten die gegevens om de mutaties zó aan te passen dat bepaalde zeldzame substructuren (zoals O-F) veel minder vaak gemaakt werden. We testten de methode vervolgens uit op de fragmenten die we in de NCI hadden gevonden (hoofdstuk 3), en het bleek dat de door datamining aangepaste mutaties inderdaad vaker een ander bestaand fragment produceerden dan de oorspronkelijke mutaties. Hoofdstuk 5 geeft de gedetailleerde informatie over het dataminen, beschrijft hoe de mutaties precies werden aangepast, en bespreekt de precieze resultaten.

Een van de lacunes die naar voren kwam uit het literatuuronderzoek voor hoofdstuk 2 was dat evolutionaire algoritmen in geneesmiddelontwerp doorgaans niet met elkaar vergeleken werden. Zeker in ons veld, het ontwerp van nieuwe geneesmiddelmoleculen, zijn er vele methodes, maar het was niet duidelijk welke methode de beste is, ofwel welke methode het effectiefst geneesmiddelmoleculen kan optimaliseren. Maar dat is wel een belangrijke vraag, omdat een goede evaluatie van een kandidaat-geneesmiddel veel tijd en geld kost, of dat nou gebeurt met een geavanceerd computermodel of met experimenten. Een evolutionair algoritme dat maar 1000 moleculen hoeft te produceren voor een bepaalde verbetering, is daardoor veel nuttiger/economischer dan een algoritme dat voor dezelfde prestatie 10,000 moleculen nodig heeft. In hoofdstuk 6 onderzoeken we één van de belangrijkste kwesties bij evolutionaire algoritmen voor geneesmiddelontwerp: is evolutie die gedaan wordt met kleine stapjes (atoom voor atoom veranderen) effectiever dan evolutie die grote sprongen maakt (dus hele stukken molecuul ineens verplaatst of verandert)? De kwaliteit van elk molecuul evalueerden we met een docking-programma, dat het molecuul in het HIV-enzym reverse transcriptase paste en de interactie-energie berekende, om een benadering te krijgen van hoe de evolutie zich zou gedragen als we de moleculen echt zouden synthetiseren en testen. Atoom-gebaseerde en fragment-gebaseerde evolutie leken het ongeveer even goed te doen, wel leerden we een aantal lessen waarmee we onze oorspronkelijke algoritmen kunnen verbeteren. Het verloop van het experiment staat samen met onze discussie en aanbevelingen in hoofdstuk 6.

Het door ons ontwikkelde evolutionaire algoritme zou samen met de resultaten van het dataminen in staat moeten zijn chemici te helpen bij het ontwerpen van nieuwe moleculen. Maar werkt het ook in de praktijk? Om dat uit te testen lieten we de Molecule Evoluator 300 moleculen genereren, die door chemici werden beoordeeld en aangepast. Uit dit proces kwamen 34 'interessante' moleculen, die we opzochten in de databases van Beilstein en de ACS om te kijken of deze moleculen of varianten erop al gemaakt waren. Van de moleculen die zowel nieuw waren als er 'geneesmiddelachtig' uitzagen, kozen de chemici een aantal uit om te synthetiseren. Uiteindelijk leverde dit acht kleine, geheel nieuwe moleculen op. Deze werden op 83 eiwitten uitgetest. Vier verbindingen bleken biologisch actief te zijn, onder andere op de α-adrenerge receptoren. Hoofdstuk 7 bespreekt dit experiment, inclusief de structuren en de syntheses. Het vinden van nieuwe, biologisch actieve verbindingen toonde aan dat onze methodes nuttig kunnen zijn voor het ontwerpen van nieuwe kandidaat-geneesmiddelen: de Molecule Evoluator kan samen met de expertise van chemici geheel nieuwe en toch werkzame stoffen ontwerpen.

Dit proefschrift sluit af met hoofdstuk 8. Het eerste deel hiervan beschrijft de conclusies die ik uit het in dit proefschrift beschreven onderzoek getrokken heb. Het tweede deel van hoofdstuk 8 bevat mijn gedachten over de verdere uitbreidingen en toepassingen voor de Molecule Evoluator en evolutionaire algoritmen in geneesmiddelontwerp, en een aantal mogelijkheden voor toekomstige software voor het helpen ontwerpen van geneesmiddelen.

Curriculum Vitae

Eric-Wubbo Lameijer werd geboren op 11 augustus 1976 in Hilversum. In diezelfde stad volgde hij een gymnasiumopleiding aan het Alberdingk Thijm College, welke hij in 1994 voltooide. In hetzelfde jaar won hij de Nationale Chemie Olympiade en werd als lid van het Nederlandse team uitgezonden naar de Internationale Chemie Olympiade in Oslo, waar hij een bronzen medaille behaalde. Daarna ging hij scheikunde studeren aan de Vrije Universiteit in Amsterdam. Zijn hoofdstage, het ontwerp van een substraatmodel voor het geneesmiddel-metaboliserende enzym cytochroom P450 2D6, deed hij in de groep van Nico Vermeulen onder begeleiding van Jennifer Venhorst. Daarnaast volgde hij stages in de chemische informatica en in de computationele organische chemie. In het jaar 2000 studeerde hij *cum laude* af in de farmacochemie, en ontving hij ook de Taeke Bultsma Award voor zijn afstudeerscriptie over antisense geneesmiddelen. Van 2001 tot 2005 deed hij promotieonderzoek aan de Universiteit Leiden, waar hij in het kader van het bioscience-initiatief onderzoek deed naar de toepassingsmogelijkheden van evolutionaire algoritmen en datamining voor het ontwerp van nieuwe geneesmiddelen. Dit onderzoek heeft onder andere geleid tot een vermelding in het blad *Chemistry World* onder de titel 'To boldly go where no chemist has gone before', en het computerprogramma 'Molecule Evoluator', dat tegenwoordig commercieel verkrijgbaar is. Sinds 2005 heeft Eric-Wubbo onder andere gewerkt als programmeur en als freelance wetenschapsjournalist.

Curriculum Vitae (English)

Eric-Wubbo Lameijer was born on August 11th, 1976 in the city of Hilversum. He also received his highschool education there, at the Alberdingk Thijm College, obtaining his diploma in 1994. In the same year, Eric-Wubbo won the Dutch National Chemistry Olympiad and was sent with three other highschool students to represent the Netherlands at the International Chemistry Olympiad in Oslo, where he won a bronze medal. Subsequently, he studied chemistry at the Free University in Amsterdam. The project of his major was creating a substrate model for the cytochrome P450 2D6 enzyme. This research was performed in the group of professor Nico Vermeulen, under supervision of Jennifer Venhorst. He also had internships in the areas of chemical informatics and theoretical organic chemistry. He graduated with honors (*cum laude*) in

2000, and in the same year received the Taeke Bultsma award for his master's thesis on antisense drugs. From 2001 to 2005 Eric-Wubbo performed his PhD studies at Leiden University under the supervision of Ad IJzerman, Joost Kok and Thomas Bäck, on applying evolutionary algorithms and data mining to drug design. This research has led to several publications, one of which inspired the magazine *Chemistry World* to publish an article entitled “to boldly go where no chemist has gone before”. It has also produced a software package for drug design, the Molecule Evoluator, which has since been commercialized. Since 2005 Eric-Wubbo has worked as a programmer and as a freelance science journalist.

List of publications

Lameijer, E.W.; Bäck, T.; Kok, J.N.; IJzerman, A.P. Evolutionary algorithms in drug design. *Natural Computing*, **2005**, *4*, 177-243.

Lameijer, E.W.; IJzerman, A.P.; Kok, J.N. The molecule evoluator: an interactive evolutionary algorithm for designing drug molecules. *GECCO 2005,* 1969-1976.

Lameijer, E.W.; IJzerman, A.P.; Kok, J.N. Using data mining to improve mutation in a tool for molecular evolution. *Congress on Evolutionary Computation 2005*, 314-321.

Lameijer, E.W.; Kok, J.N.; Bäck, T.; IJzerman, A.P. The Molecule Evoluator. An Interactive Evolutionary Algorithm for the Design of Drug-Like Molecules. *J. Chem. Inf. Model.*, **2006**, *46*(2), 545-552.

Lameijer, E.W.; Kok, J.N.; Bäck, T.; IJzerman, A.P. Mining a Chemical Database for Fragment Co-occurrence: Discovery of "Chemical Clichés". *J. Chem. Inf. Model.*, **2006**, *46* (2), 553–562.

Lameijer, E.W.; Tromp, R.A.; Spanjersberg, R.F.; Brussee, J.; IJzerman, A.P. Designing Active Template Molecules by Combining Computational De Novo Design and Human Chemist's Expertise. *J. Med. Chem.*, **2007**, *50* (8), 1925–1932.

Ye, K.; Lameijer, E.W.; Beukers, M.W.; IJzerman, A.P. A two-entropies analysis to identify functional positions in the transmembrane region of class A G protein-coupled receptors. *Proteins*, **2006**, *63*(4), 1018-1030.

The research described in this thesis has also been the subject of an article in "Chemistry World",
Evans, J. To boldly go where no chemist has gone before. *Chemistry World*, **2006**, February 8th**.**

Epilogue

There exists a stereotype of a lone scientist toiling in a laboratory, making great discoveries that will amaze the world because of his or her great mind seeing the Truth. However, pursuing a PhD makes one aware how wrong that image is, especially on the 'lone' part. While it may be possible to make great discoveries with nothing but your own observations and the knowledge of your predecessors to aid you, my research at least would never have been possible without ample intellectual, material and emotional support by many different people.

First of all, this thesis would never have come into being without the encouragement and support of my three supervisors, Ad IJzerman, Joost Kok, and Thomas Bäck. Next to obtaining the funds that made it possible for me to pursue my PhD without needing a second job to cover my expenses, they guided me in exploring their fields of interest, from medicinal chemistry (Ad IJzerman) to data mining (Joost Kok) and evolutionary algorithms (Thomas Bäck). Furthermore, they were always willing to listen to me whenever I was stuck at something, and mentored me through the process of getting my first papers in publishable form.

Further support and stimulation during my Leiden period was provided by the contacts and discussions with my colleagues both inside and outside the division of Medicinal Chemistry. Most of all I enjoyed the discussions within the 'computer group' with Jeroen Kazius, whose ominous clothes have never concealed his big heart, Kai Ye, who combines a serene spirit with a quick and innovative mind, and last but not least algorithm whizz, architecture buff and good friend Siegfried Nijssen. I fondly remember the talks with these three on subjects ranging from computer science to natural science, from surviving professors to supervising students, and from life as a PhD-student to Life as a whole.

But not only 'computer people' were important for my research. Crucial for developing the Molecule Evoluator were the discussions with and critiques by the medicinal chemists, Johannes (Hans) Brussee, Reynier Tromp, Aniko Göblyös, Jaco van Veldhoven, Lisa Chang and Ron Spanjersberg, who were willing to help me transform the idea of an interactive evolutionary algorithm to something that chemists could use. Neither will I forget the great atmosphere in the medicinal chemistry group contributed by the biology league/band/posse, Margot, Miriam, Ann, Elisabeth, Rianne, Henk, and their students innumerable. My special thanks in the social category go to Thea Mulder and Laura Heitman for co-organizing laboratory events, and to the memory of Jacobien von Frijtag Drabbe Künzel, organizer, cook and person *par* *excellence*, who, despite our very different interests and personalities, always made me feel a welcome and worthwhile person.

Outside our group, I was honored to work with the “Chris team” from Amsterdam, comprising of the constantly busy but ever supportive young scientists Chris Oostenbrink and Chris de Graaf. Chapter 6 is the result of this collaboration. I also thank the companies Grünenthal and Organon/Schering Plough and their medicinal and computational chemists for their willingness to take the time and effort to give feedback on the first versions of the Molecule Evoluator. Within Leiden University I am grateful for the help of Jelle Goeman, a young mathematician with the rare and precious gift of being able to explain statistics clearly to non-statisticians. Jelle was a great aid in getting chapter 3 of this thesis published. Also within Leiden University I am grateful for the coaching by and conversations with ‘P&O’-man Kick Moors, who encouraged me to continue even when the going was rough, and helped me with perhaps the most tricky part of science for an “exact” scientist: managing the emotional aspects of research. Finally, as my 'farthest collaborator' I thank Wanjin Tang, my listening ear in distant Sweden and later in the even more distant Texas, who sometimes made me see a situation differently with her Chinese wisdoms, or made me smile with her photographs of humorously prepared food. I am grateful for our LDF (long distance friendship) during these last six years.

I also thank six special persons for helping in the construction of the Molecule Evoluator, for teaching me to teach, and for providing me with perhaps the most stimulating and inspiring part of my work in Leiden: my students. Computer scientists Tijmen and Kim challenged me to extend my communication style to non-chemists, which was valuable training. But especially I thank my chemical and biopharmaceutical students, whom I had the pleasure and privilege of supervising for a longer time: Eelke van der Horst, who would never let so-called 'midnight' stop his efforts to solving a programming problem (and has later single-handedly dragged the Molecule Evoluator into the age of Windows), Jurre Kamphorst, who tackled (and still tackles) research with the same discipline and professionalism as he applied to his rowing, Daan Acohen, who sometimes seemed to prefer molecule revolution to molecule evolution yet nevertheless made an essential contribution to the Molecule Evoluator and chapter 6 by designing and implementing fragment-based evolution, and finally Guoxiang Liu, my student from farthest away, for whom computer science in Leiden may have been a very strange and stressful environment, but whose perseverance and willingness to tackle research problems far out of his comfort zone I greatly admire. I enjoyed my contacts with them, and I hope that they will become successful and happy in whatever field they choose.

One can however not give to others what one has not been given first. While the faults in this thesis are of course wholly my responsibility, for its qualities it is greatly indebted to the people who guided my growth as a scientist. Next to my three Leiden supervisors, these are the teachers and mentors I had before, who edged me towards formulating more sharply and thinking more critically. Most prominent among these were Jennifer Venhorst, Rosa Bulo, Andreas Ehlers and Jan Commandeur. I am grateful for their guidance. Yet one person among my educators stands out: my high school teacher, Olaf Budde. Whereas my other mentors taught me *how* to do science, “Mr Budde” (also informally known as the 'jazz man') did something even deeper and more important: teaching me *why* to do science. He was the living embodiment that chemistry is not merely 'interesting', but that it can be creative, fun and wonderful. He was a true teacher of the joy of chemistry, and has always inspired me on my path.

Lastly, I thank my family for their unwavering support. I thank my sister Nanette and her partner Ron for their frequent hospitality, excellent cooking and being there for me when I walked through my personal valleys of PhD-doubt. I also thank them for the initial designs for the cover of this thesis. I thank my mother for always giving me a home in Hilversum whenever I was ill or needed relaxation from work, and for the almost unlimited amounts of postcards she has sent me to cheer me up and encourage me with my projects, next to taking care that I would exercise properly by walking in healthy woods, and would not lack the essential nutrients provided by chocolate and Limburg flans. Finally, I thank my father, who from my earliest youth helped ignite a love for science and learning in me. When I was little, he read the biographies of famous chemists to me at bedtime. He pointed me towards my first chemistry book, and has provided me with many volumes of old but fascinating chemistry literature even before my official first chemistry lesson in high school. Throughout the years, he shared his love of knowledge, science, and pharmacology with me, and he was my mentor for my first presentations. But above all he was to me an example of integrity and compassion, living the principle that knowledge can be interesting and useful, but that a person cannot be complete without compassion and willingness to help others, and the courage to do what he or she knows is right. It is the greatest regret of my PhD-period that I could not finish my thesis in time for him to witness it, but I know that he would be proud of me.