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.

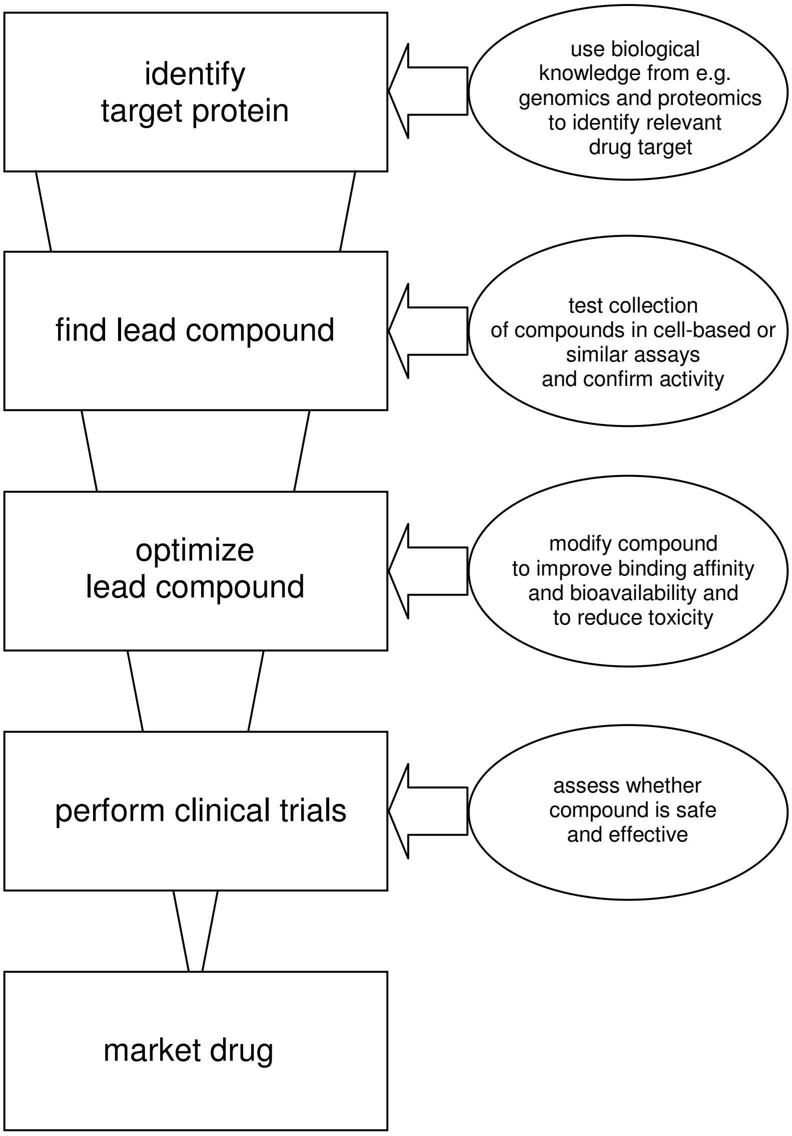
identify target protein
use biological knowledge from e.g. genomics and proteomics to identify relevant drug target
find lead compound
test collection of compounds in cell-based or similar assays and confirm activity
optimize lead compound
modify compound to improve binding affinity and bioavailability and to reduce toxicity
perform clinical trials
assess whether compound is safe and effective
market drug

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 $Y = f(x_1,\ldots,x_n) \rightarrow \min (\max)$ 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 $f(x_1,\ldots,x_n)$ (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.

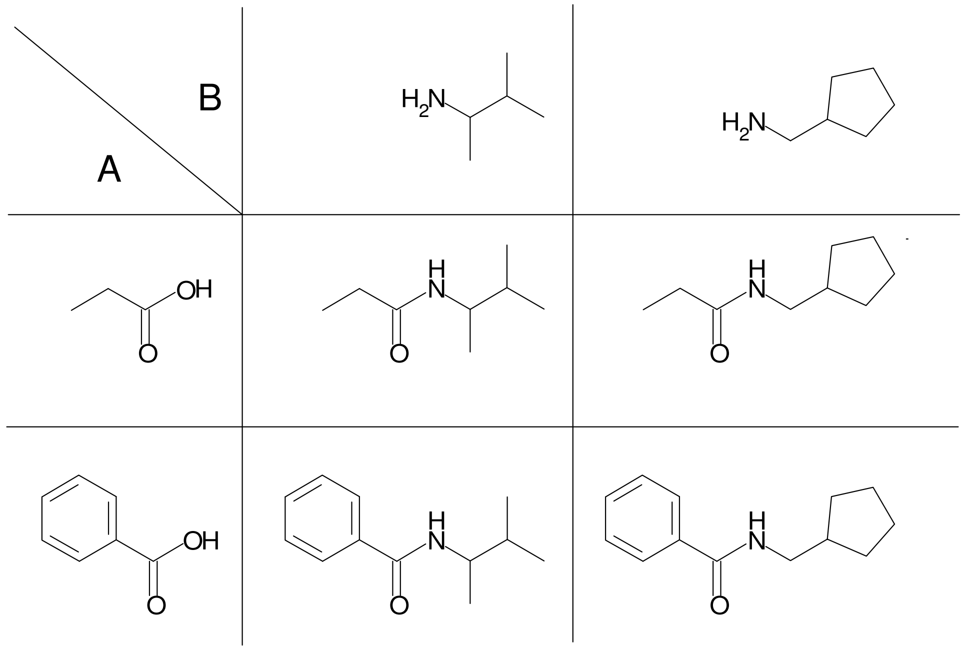

Figure 2.2: A simple combinatorial library.

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).

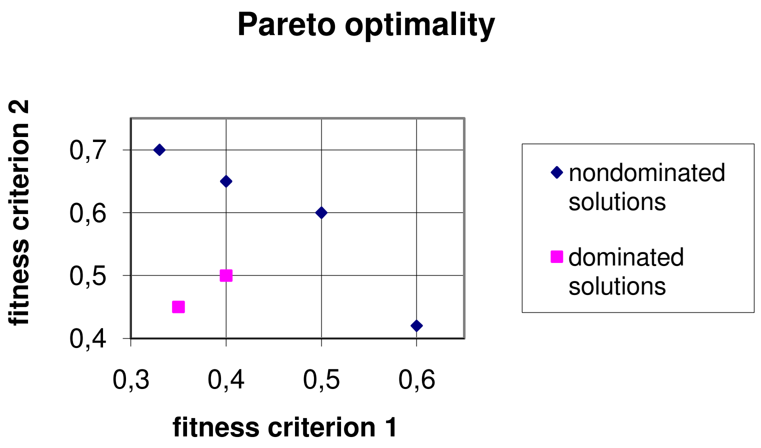

Figure 2.3: Pareto optimality. In this example, both fitness criteria are to be maximized. A solution is dominated if there exists another solution that has equal or better scores on all criteria. for example (0.5 , 0.6) dominates (0.4 , 0.5) because 0.5>0.4 and 0.6>0.5. However, (0.5 , 0.6) does not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

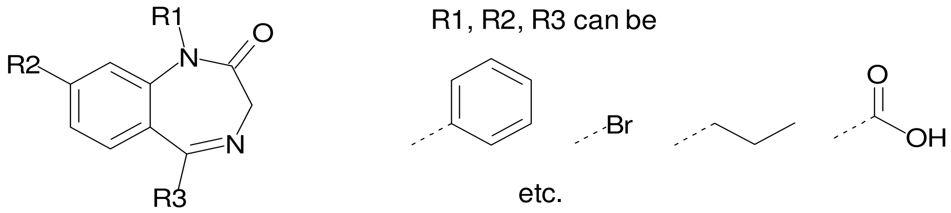

Figure 2.4: Template-based (virtual) library design.

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.

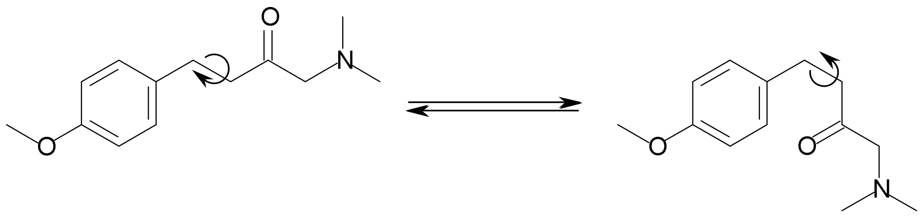

Figure 2.5: Change in conformation by rotation around a bond.

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 1.9×10^8 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*.

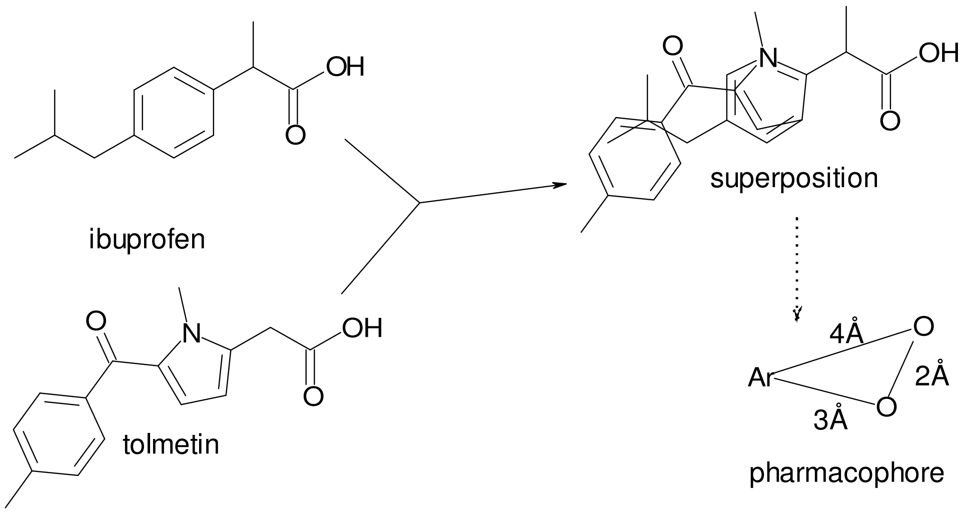

Figure 2.6: Molecule superposition and pharmacophore detection. “Ar” stands for aromatic center, 1 Å is 0.1 nm.

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.

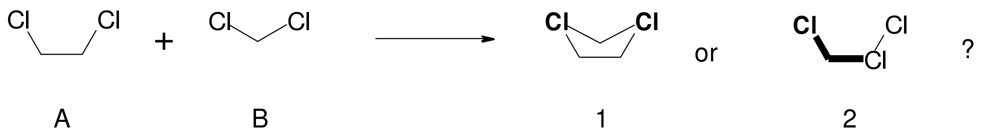

Figure 2.7: Which superposition is best?

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 $<-9.545\text{-HOMO}>^2$, which was combined linearly with the other (derived) descriptors. A typical individual thus may look like $\{C_4,<2.301\text{-}U_t>,(U_t\text{-}2.966)^2,<-9.631\text{-HOMO}>^2\}$. 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 pseudo-receptor 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.