8 Conclusions and Future Perspectives

Conclusions

In this project, we have developed a computer program for *de novo* molecule design, the Molecule Evoluator. It is unique among programs for *de* novo molecule design since it combines three features: an atom-based approach, an evolutionary algorithm that can optimize structures, and its interactivity which allows it to profit from the knowledge, intuition and creativity of its user.

The first feature, the atom-based approach, helps the Molecule Evoluator to search all of chemical space, and fine-tune the structures. When using the much more common fragment-based approach, one first faces the problem of whether all synthetically possible and drug-like fragments have been included, and second there is the question whether the reconnection algorithm uses all realistic possibilities of synthesis. Covering chemical space with the atom-based approach is much easier and more natural. Secondly, our exhaustive repertoire of atom-based mutations (several of which lack in other work) allows the molecules to change gradually and adapt themselves to their target, instead of making big jumps in structure which usually result in large loss of fitness and may tend to force the population into premature convergence.

The second defining feature of the Molecule Evoluator is the evolutionary algorithm on which it is based. While several optimization methods exist (like random search, simulated annealing, or just molecular growth), evolutionary algorithms make good use of two features of molecular space. First, that molecules close in structure generally have related biological activity; evolution's concept of heredity, of inheriting good genes from the parents, makes methods which base the new molecules on the previous ones (instead of searching randomly) a good choice. Secondly, both in

literature and in our own experiments we found that perhaps the most defining feature of evolutionary algorithms, namely crossing different solutions, is also advantageous and viable in drug design. Evolutionary algorithms not only use heredity, but also crossover, and therefore automatically use the structure of biological activity space to their advantage.

The third, and perhaps most distinguishing feature of the Molecule Evoluator is its interactiveness. While interactive evolutionary computing (IEC) has been used for quite some time in diverse applications, it had not yet been used for molecule evolution. However, there are good reasons to use interactive evolution in this field. First, there is a lack of good "fitness functions" – methods to calculate "how good" a molecule really is. Programs to estimate ligand binding energy and ease of synthesis are still very unreliable, so evolution without any human intervention will rarely yield good structures. However, human domain knowledge on ease-of-synthesis and pattern-recognition may help. Second, it can be difficult for people to accept the computer "prescribing" molecules out of the blue: most chemists would either like a good reason or some input of their own. The limitations of current automatic structure evaluation would lead to rejection of flawed structures, instead of correction. Third, evolutionary algorithms do not only "exploit" existing knowledge, they also explore new possibilities. Since a computer can quickly generate many possibilities and has other prejudices than a human chemist, interactive evolution can be a valuable idea-generating machine to complement human creativity.

During the development of the Molecule Evoluator, we discovered that the first versions needed to be refined to be acceptable to the chemists using them. In the following paragraphs we describe some of the problems encountered and the modifications implemented in response.

The main problem of the first version of the Molecule Evoluator was that many structures just did not seem possible to synthesize at all. In particular, common substructures like phenyl were almost completely absent, weird substructures like peroxide were common, and many structures had overly complicated rings or disobeyed chemical rules of thumb, like Bredt's law which states that a bridgehead carbon atom cannot have a double bond (unless the rings have a certain minimum size).

To make the molecules more "appealing" and easier to synthesize, we first allowed the ME to not only add atoms to a growing molecule, but also a number of predefined fragments (carboxy, phenyl, cyanide, etc.). Secondly, we mined the NCI database to find the frequencies of the different atom types and 2/3/4-atom substructures. Based on the frequency of, for example, O-O in the NCI database we

modified the chance that an oxygen atom would be connected to another oxygen atom. This automatically enforced chemical rules that state that peroxides are rare, using statistics instead of qualitative human intervention. Third, we implemented a couple of rules, such as Bredt's rule and a prohibition of CH_2-imines. These rules act like filters that prevent molecules that do not obey the given chemical constraints from being shown to the user. Mining the NCI also gave us a catalog of ring structures (see the "Chemical Clichés" chapter) which were also implemented to filter unknown and probably strained rings out. Note that the filters can be activated and deactivated by the user, and that the full idea generation potential remains available if desired.

The second main point for improvement was user control. For example, chemists often had certain ideas about which part of a molecule was important, and should be conserved. Also they preferred that the best molecule from the previous generation was to be saved always (which is not guaranteed in a normal evolutionary algorithm). Thirdly, occasionally a chemist could see an obvious modification of an existing molecule, and wanted to put the adapted molecule in the Molecule Evoluator. And finally, the molecules produced should preferably be drug-like and obey a number of physicochemical restraints.

We made various adaptations to address these points. We added atom and bond fixation, which can conserve any atom (even hydrogen atoms) and can therefore focus evolution on the variable part of the molecule. Secondly, we added elitism, so the chosen molecules of the previous generation appear as the first molecules of the current generation, allowing a chemist to easily see if their offspring improves over them. Thirdly, we added a molecule editing window, which allows the user to adapt ME-generated molecules and feed them back into the evolution, or even to sketch new molecules as input of the evolution. Lastly, we added physicochemical filters to allow the user to determine the allowable physicochemical properties of a molecule, such as the range in which the molecular weight should fall, and the maximum permissible number of hydrogen bond donors.

Finally, we tested the Molecule Evoluator to examine whether the concept was sound and useful.

First we ran a number of small experiments, in which we evolved certain drug molecules from scratch (that is, without editing the molecules, though we fixed certain atoms/bonds to accelerate optimization in a certain direction) to show that we can indeed convert simple substances into drugs. We also were able to reproduce, using experimental fitnesses, an optimization pathway of neuramidase inhibitors, which shows that our mutations and selection work well with good (experimental) fitness

measures.

The value of the idea-generating function was tested by creating 300 random molecules that obeyed certain physicochemical restraints. A panel of chemists chose the most “drug-like” 34 structures, of which a number did not appear as structure or substructure in existing databases, and therefore could be considered potential new molecule templates. Synthesis of eight of the compounds yielded four compounds which showed biological activity in the used essays. It seems that chemists indeed have a valuable intuition, and that the ME can inspire the synthesis of truly new classes of molecules, unknown before yet possible to synthesize.

The Molecule Evoluator has become more advanced over the years, and is at the time of writing (May 2008) commercially available. In the Leiden group of Medicinal Chemistry, where it has been developed, it is now used in each synthesis project as an interactive, idea-generating but responsive aid to get new ideas for structure modifications. Several companies have bought versions, and some are quite happy with it, as is evident from the following quote:

"Both computational chemists and medicinal chemists have explored the Molecule Evoluator and have been excited about the output in terms [of] novel ideas being generated and the potential for further enhancements in the future. The real advantage of the current programme is that it can be iteratively influenced by trained chemists to propose new structures, some of which may look immediately obvious but yet had not been previously suggested. Three of our current GPCR-based projects have benefited in this way."

Software that allows humans and computers to combine their particular strengths is still rare, for drug design the Molecule Evoluator is the only one to our knowledge which is currently commercial. This has two likely causes. First of all, interactive evolution is itself a young field, most programs for designers (of molecules or buildings) are drawing programs which do not give any creative input of their own, as they were designed to be computerized replacements for real drawing boards. Second, most complex software for molecule design has been created for computational chemists and therefore only runs under Unix/Linux workstations and has powerful but complex interfaces. The medicinal chemists, the people who design and modify most of the molecules and are experts on molecules rather than computers, have been left out. Only in the last two or three years software companies are also starting to develop versions for Windows and thus for the “normal” chemists. (See for example the

Software section in the biweekly ‘Chemical and Engineering News’ of the American Chemical Society). But even when a Windows version is available, it will be a long road for most programs to also become user-friendly for people who are not experts in computational chemistry. As of yet, the Molecule Evoluator is quite lonely in the software landscape, but we hope that in the coming years it will be joined by followers and colleagues which bring both the computational and creative potential of computers to the medicinal chemists directly. Software has a vast potential for changing drug design, if we invest effort and creativity in it.

Future Perspectives

Prediction is very difficult, especially about the future.
Niels Bohr (1885 – 1962)

Science is never finished. The research described in this thesis may have produced a useful computational tool for the medicinal chemist, and it may have given more insight into chemical databases. However, we nor others have as of yet produced the perfect medicinal chemistry tool, and many problems in drug design remain. The previous chapters of this thesis have covered what we have done in our research. This last part will contain reflections on where to go from here. I hope this chapter will provide ideas and inspiration to researchers and non-researchers alike on what subjects in computational drug design would be worthwhile to investigate, and in which directions we could go.

I will begin with some thoughts on future directions for the Molecule Evoluator, then discuss the possible evolution of evolutionary algorithms in drug design, and will end by zooming out to look at the general role of software in drug development, and talk about some of the ways in which we can increase the ability of software to help us design new drugs.

The future of the Molecule Evoluator

At the time of this writing, we have performed experiments which have shown that the Molecule Evoluator can at the very least help find novel biologically active molecules. We may never know if chemists without the Molecule Evoluator would have been as creative as chemists using the Molecule Evoluator, but it is very likely that computer-generated structures can complement the brainstorming by chemists, which may mainly design variations on the molecules they already know. For that reason, the Molecule Evoluator as it is now may remain useful for a long time to come.

To enhance the usefulness of the Molecule Evoluator further, there are numerous possibilities: improving the speed at which molecules are generated, comparing the effects of different crossover functions, making the user interface even more intuitive, improving the display so that the user can see very quickly which mutations have been generated, offering calculations of more physicochemical properties (for example pK_a) or linking the Molecule Evoluator to third-party software that can calculate those

properties, and numerous other tweaks and enhancements. At the moment there are however three points which I think most promising for future updates: changing the structure generation to yield even better structures, linking the Molecule Evoluator to other software and databases, and experimenting with computational fitness evaluations.

Ease of synthesis: less boring, less impossible, more novel?

From conversations with users, we found that the main factor determining how much they liked the Molecule Evoluator was the ratio of “good” molecules to “bad” molecules. Good molecules are those molecules that are novel and seem relevant or at least can be easily changed into a molecule with a good structure. “Bad” molecules are those molecules which are boring (not very novel), irrelevant, or plainly impossible to synthesize.

The most significant way to enhance the use that chemists get out of the Molecule Evoluator would therefore be increasing the number of good molecules while preferably decreasing the number of bad molecules. With previous adaptations we have already succeeded partially in this, and it is certainly possible to further improve our results with some additional adaptations of the code.

At the moment the best method to improve the ratio of good to bad molecules seems to be to diminish the occurrence of the main types of bad molecules: the 'impossible' molecules, the 'irrelevant' molecules, and the 'boring' molecules.

The 'impossible' molecules are those molecules which cannot be synthesized. We have already reduced their number with chemical filters and giving the user the option to only allow known ring structures; further feedback will undoubtedly allow us to increase the number of filters that can be applied to a molecule. The 'irrelevant' molecules are only created when the Molecule Evoluator cannot find a mutation that works, which mostly occurs in molecules where many atoms or bonds have been 'fixed' by the user. The solution to this is to rewrite the mutation algorithm: at the moment it picks a random atom from the molecule to mutate, which fails if that atom has been fixed by the user – forcing the Evoluator to create a random/irrelevant molecule instead. Rewriting the mutation algorithm so it picks only from the atoms which are *not* fixed will give a much greater mutation success rate and therefore a much lower production of irrelevant molecules.

The final way to decrease the amount of “bad” molecules is to tackle the boring molecules (this is for interactive evolution. An automatic fitness function cannot be bored). A chemist may find certain mutations boring or “not novel”. The aim therefore

is to find out which mutations are generally found interesting, and which are not. Finding out these preferred or unpopular mutations can be done by either directly observing a user or by creating special statistical subroutines to observe which mutations (atom addition, deletion, insertion) produce molecules that are most often selected or not selected, and what kind of additions/deletions/insertions are most interesting. Such an investigation might for example find that adding a methyl group to a benzene ring is “boring”, while adding a hydroxy group to the same ring is “interesting”. The probabilities of those specific mutations can then be adjusted appropriately.

In conclusion, adapting the Molecule Evoluator to change the ratio of good to bad molecules in a beneficial way is certainly possible with user observation and feedback and some reprogramming. Of all the possible options to improve the Evoluator, this optimization may have the strongest impact on user-friendliness and frequency of use, and would therefore be a prime target for implementation.

Linking the Molecule Evoluator to databases

A second area for improving the Molecule Evoluator turned up during our own tests. While trying to find novel biologically active molecules, every compound the chemists found interesting had to be manually looked up in the CAS database. While this database search was by far not as much work as eventually went into synthesizing the truly novel compounds, it taught us that it would be incredibly handy if one could look up the Molecule Evoluator-generated structure or similar structures in the user's favourite chemical databases by just pressing a button. A useful improvement would therefore be a link to databases that would allow chemists to find whether the molecule suggested by the Molecule Evoluator exists in its entirety or as a substructure, and if it exists, how it can be synthesized (or ordered). For large and wealthy institutions, links to commercial databases like Beilstein and CAS may be possible, but more exciting is the opportunity brought by the advent of large public databases like PubMed, PubChem, eMolecules and ChemSpider to give all users of the Molecule Evoluator the chance to have structures automatically checked with literature.

Databases could not only be used for checking structures, but can also help to create structures. If, for example, a chemist is looking for alternatives for a certain ring in a molecule, it would be very useful if he could view a list of the most common ring systems from our 'chemical cliché' database next to the molecule editing window, as that could give many ideas for changes. A similar approach could be taken for substituents, where the chemical clichés fragment database or a specialized database

like a bioisosters database could be used to find replacements that the chemist might not have thought of yet.

The Molecule Evoluator and automated evolution

Interactive evolution has some major advantages over automated evolution, mainly that it can use expert knowledge much more easily than any fitness function designed by computer programmers. However, this requirement for expensive expert time is also a disadvantage, and given the successes of the interactive mode of the Molecule Evoluator, one could consider adding options for computational chemists who want to use automated evolution.

While adding a feature for automated evolution is possible (in fact, it has already been done in one or two individual cases), to make the automated evolution perform optimally one needs to change more than the code for the fitness function. Automatic evolution and interactive evolution, despite their apparent similarity in approach, are as dissimilar (if not more dissimilar) as tennis and table tennis.

The first dissimilarity of automated versus interactive evolution is the absence of user fatigue. This opens up the desirable possibility to create larger generations than the 12-20 which are practicable for user feedback (50-100 molecules seems to be about the optimum size if we consider investigations such as that of Douguet et al.[1], since it may avoid the premature dead ends which endanger small populations and the 'drowning out' of the good genes in very large populations. Larger populations seem to work better when split into 'islands'). Also, automated evolution doesn't need settings that prevent molecules that differ only slighly from their ancestor being created, as a small increase in fitness is useful, whereas such a molecule would strike a human as uncreative and increase user fatigue.

The absence of user selection, however, also has some disadvantages. First of all, automatic evolution makes it necessary to implement a selection function: for if the user isn't selecting the “good” molecules, the programmer has to decide how to select the best molecules for further evolution. Take the best five molecules? Or ten molecules? Use tournament selection? Roulette wheel selection? A second disadvantage, which we discovered during the docking experiment (Chapter 6), is that atom-based evolution when not supervised by humans tends to produce molecules which over time become more and more difficult to synthesize. Therefore, automated evolution needs stricter filters to sufficiently dispose of unwanted structures.

In summary, unlocking the full potential of automated evolution requires changing more parts of the Evoluator than the fitness function – the automatic evolution would

need to be at least partially split from the code for interactive evolution. On the positive side, automated evolution also offers opportunities. Performing automated evolution with a good fitness function may give us a better idea what parameter settings and what mutations or crossover operations are optimal for drug design, and allow us to adjust the Molecule Evoluator accordingly. If we keep studying drug design and evolutionary strategies for drug design, the Molecule Evoluator may one day not only be the best interactive evolutionary algorithm for drug design, but also the best evolution-based program for automatic drug design available.

General perspectives on evolutionary algorithms in drug design

Useful as they are, evolutionary algorithms aren't the “cute new kid” in computational drug design anymore. They were immensely popular in the late 1990's, but interest waned as it grew more and more difficult to think of yet-unpublished applications and it turned out that evolutionary algorithms, like all methods before them, were not the “cure-all, one-size-fits-all”-solution that drug designers have been seeking for so long. I have discussed my view of the future of Evolutionary algorithms in 2005 in my review on evolutionary algorithms in drug design (Chapter 2), in which I discussed various possible developments such as creating standardized test databases for chemical problems and evaluating newer types of evolutionary algorithms. Currently, evolutionary algorithms are already unobtrusively integrated as standard tools for experiments, for example for descriptor selection in a virtual screening experiment[2]. For new problems too, evolutionary algorithms are becoming easier to try out as flexible optimization methods, due to the development of EA toolkits such as the GA Playground, OAT, Lil-gp, and ECJ[3]. However, in my mind two ideas seem most important: adding further domain knowledge to evolutionary algorithms, and the use of evolutionary algorithms in novel or at least uncommon ways in drug design, such as modelling and data mining.

Adding knowledge to evolutionary algorithms

The main bottleneck in successful application of evolutionary algorithms in drug design is that finding the best solution (or even a very good solution) often takes more computer time than is available. This is both because of the “high dimensionality” of many drug design problems (many parameters need be optimized simultaneously) and because fitness functions often take much time to be calculated. While the increasing

computer speeds may help here, it may even be more important to perform the optimization itself in more intelligent ways.

Evolutionary algorithms can be improved by testing and comparing different models (differing in population size, selection pressure, etcetera), but more important will be the collaboration between computer scientists and experts in the problem domain, such as drug design. Currently, relatively simple evolutionary algorithms are often used since they shorten programming time – unfortunately, these same algorithms are afterwards too easily carried over into the commercial version where the 'saving' of programming time is paid back with interest as the inefficient algorithm is run thousands of times. Only with knowledge of the problem itself can one develop rough fitness functions which can eliminate patently bad candidates before they are subjected to a more accurate fitness calculation, create meaningful mutations that turn good solutions into other good solutions instead of impossible ones (for example, producing a carbon atom with five bonds), split the solutions as much as possible in semi-independent sub-systems for faster optimization, and make optimal use of the knowledge obtained by the evolution so far (for example, learning that a certain type of atom at a particular position produces very good scores). It may be difficult for computer scientists and drug designers to understand each other and communicate one's knowledge and aims clearly to the other party, but in the end the algorithms that will survive and turn out to be most useful in a given problem domain will not be the newest generic computer science methods, but well-chosen basic methods, carefully optimized to suit the problem at hand.

Future applications of evolutionary algorithms

Evolutionary algorithms are currently experimentally or routinely used in most phases of drug discovery. The most important of their current applications are their contributions to the “core” business of finding new leads by computer, namely by library design, *de novo* design, and virtual screening. Of these, “virtual screening” (evaluation by computer) of a potential lead is hardest by far, since proper computational evaluation needs to answer six questions: 1) is the target important in the disease, 2) does the candidate molecule interact strongly enough with the target, 3) can the molecule get to the place of action, 4) how and how fast is the molecule metabolized, 5) are the molecule and/or its metabolites toxic, and if so, how much, and 6) is the molecule excreted slowly enough.

While docking tries to answer question 2, and Lipinski's rules and calculating the polar surface area of the molecule help us somewhat with 3 (barring active transport),

reliably predicting any of these relevant properties requires good-enough predictive models, and even a much-researched subject such as docking can definitely be improved greatly yet. The main challenge of computational drug design is therefore to develop better models for the interaction of a molecule with the human body.

Creating models generally starts with collecting large amounts of raw data (for example, molecules and their intestinal absorption), calculating descriptors (properties of the input molecules, such as the weight or the polar surface area of each molecule) and picking a computational model (linear regression, neural network, support vector machine) that links those descriptors to the measured property. With the unavoidable measurement errors in experimental data, perfect mathemathical relationships are generally not possible, but evolutionary algorithms could help in parameter selection (as they have done for QSAR), and even with computational model selection. In some relatively simple cases this is already possible, such as evolutionary algorithms producing mathematical equations out of pictures of moving systems[4]. In the end, evolutionary algorithms could become more independent and work on more complex problems, becoming untiring generators and testers of hypotheses. Even more than today, future evolutionary algorithms may supplement human brain power in making better models to predict how a particular molecule will fare as a drug.

The future of computational medicinal chemistry

While medicinal chemistry changes, its basic challenges stay the same. People will continue to have diseases and want to get rid of them, and unless genetic modification of living humans becomes easy, reliable and safe (which seems very unlikely to happen this century) we will in most cases need to fight the diseases with drug molecules. These will remain difficult to find since we do not always know the mechanism of the disease or the best protein to target, and even if we know those we may struggle to find the molecule that interacts effectively with that target, can get to the place of action, and does not have unacceptable side effects or toxicity.

Most current sub-fields of computational medicinal chemistry, such as docking and prediction of ease of synthesis will probably grow and improve over time, though that is likely to be a slow and laborious process. The algorithms may never be perfect, but they may become so good as to be too useful to ignore. The three areas which interest me most, however, are still much earlier in their development: automated data analysis,

simulations and interactive software.

Automated data analysis

In theory, knowing the DNA encoding for a protein should be enough to calculate the three-dimensional structure of the protein and find molecules which bind to it, by applying quantum mechanics and molecular dynamics. Similarly, an optimum-yield synthetic route for a new molecule could be found by using retrosynthesis and quantum mechanical calculations on a library of available reagents. In practice, however, we need lots of data both as primary input (you can't understand an organism without knowing at least its entire DNA), and as a substitute for calculations which would be theoretically possible but far too computationally intensive to be practical. If you would need either 50 years of computer time to correctly calculate the affinity of a lead compound to a protein using quantum mechanics and thermodynamics, or one hour of biochemical testing, the latter is far preferable. Therefore drug design still needs lots of experimental data to be efficient.

These experimental data are increasingly becoming available through scientific journals creating online versions, electronic lab journals collecting the primary data generated by researchers, and the growth of public databases such as PubChem and Wikipedia. This increase of data is promising, but current search methods have difficulties exploiting it: it can be incredibly hard to find the particular piece of data one is looking for. For example, in my stints as a science journalist I have spent much time being frustrated and giving up searches when Google couldn't locate a proper answer to even elementary scientific questions such as which hormones affect the release of GnRH from the hypothalamus. This information undoubtedly exists in reviews and/or books, and a more elaborate searching through reviews could probably uncover it, but still the (time) cost of finding particular information is much higher than would be technologically necessary. Making information easier to find can reduce both the time and the cost of any research project.

A large part of making better use of existing data will be economical and organizational: somebody has to pay for the servers to house the data and the transformation of (scientific) texts and tables into a computer-readable format; in many cases the 'owner' of the data (such as a scientific journal) will also want financial compensation. But what are the technological aspects?

The first part of technology will be disambiguation and transforming text into something the computer can more easily relate to search queries; for example a text like “the melting point of benzene is +5.5 ^{o}C” would fit better in a computer database

as “object=”benzene”::==reference=”www.merckindex.com/benzene”=>property = “melting point”=>value=”278.5”=>unit=”K”. Ontologies/synonym lists and lists of standard terminology will remain necessary to transform raw data into something that is as standardized and unambiguous as programming language statements (for example, instead of “melting point” my version of the Merck index occasionally uses the less standard “solidif”).

The second and probably greater challenge will be to make search methods less “fussy” and more “fuzzy”. For example, when designing a synthesis route for a compound, it may be that the compound has never been synthesized before, or that it has been synthesized, but via a rather inefficient route. Finding the synthesis route for similar compounds can help in both kinds of cases. However, despite Tanimoto coeffients and fingerprint calculations, it is unlikely that there is an universal measure of molecular similarity (similar molecular weight? Similar size? Similar melting point? Similar biological effects?). Domain experts and ideally data mining should be able to discover what the similarities are in compounds synthesized via the same route, for example, re-discovering that alcohols can be made out of available halo-alkanes using a Grignard-reaction, but also in which cases Grignard fails or is not used at all. This will be a complex piece of data mining where programmers and domain experts will have to work together, but having a system that can learn from existing data and adapt its knowledge when encountering new information would be extremely useful to build.

There are already several individuals and groups striving to improve data management in science from the 'let's do the same as always, only electronically' stage to transforming global scientific knowledge into a truly useful search engine, for example the Scientific Publishing Taskforce which is developing methods to make scientific publications computer-readable (http://esw.w3.org/topic/HCLS/ScientificPublishingTaskForce), attempts to unify data from many scientific sources into a single encyclopedia/wiki (http://www.wikipathways.org/index.php/WikiPathways) and a more fuzzy version of PubMed, eTBlast (http://invention.swmed.edu/etblast/index.shtml). While the diversity of initiatives indicates that this field is far from ripe yet and standards need to be developed to unify the different projects, going into the direction of converting data into standardized formats and making search engines more intelligent will in my opinion greatly improve data availability and increase literature-search efficiency, and drug design can only profit from that.

Simulation

The paragraph on future applications of evolutionary algorithms focused on model development. Models, however, are not limited to linear regression models or neural network models or decision trees only. One of the most interesting directions that modeling is going is the simulation of cells, tissues and even of entire human bodies. Models for the heart are already available, as are models for blood flow and some cancers[5]. Evolutionary algorithms already help find good parameters for conventional models (such as neural networks), likewise they may help fine-tune simulations by supplying good values for missing parameters. All in all, though, evolutionary algorithms would only be a small part of the total work on simulations – many computational methods and algorithms are likely to be necessary, from logic programming and cellular automata for rough qualitative models to differential equations working on small simulated 'boxes' of cellular cytoplasm and cell membrane for the most advanced models.

While it may be difficult to create good simulations, it would be amazingly useful if we would have more reliable models for what happens in the various tissues in case of disease, and what the effects of various interventions would likely be. In contrast to biological experiments, it is very easy to knock a gene out in the computer or let a protein be deactivated fully without the need to discover a strong and selective protein antagonist first. Even better, good computer models could act as “living encyclopedias” of current biological knowledge, linking the many findings of biological research into one easily accessible and correctly interconnected whole, growing more and more accurate the more we learn about biology. Creating excellent simulations of biological systems may be one of the most difficult projects that could support drug design, but its huge advantages in both allowing us to test compounds reliably without endangering human volunteers and understanding the true mechanisms and complexities of disease will also probably make it the most useful of all possible software if we succeed.

Interactivity

On the surface, interactive evolution may look like nothing more than a speeding up of the usual process of the medicinal chemist stating the target and the constraints to the computational chemist, who programs these into the computer, which produces new molecules. It is indeed an advantage that interactive evolution can shorten and speed up this cycle, but the truly qualitative difference is that interactive evolution can use one resource which automated evolution can not take advantage of: the subconscious knowledge and expertise of the medicinal chemists.

As I stated in the introduction of this thesis, major progress could be made by letting people and computers work together more productively, complementing each other's strengths. While delegating simple tasks to computers has of course been done since the dawn of computing, for example by humans creating the formulas for the spreadsheet and the computer doing the calculations, only more recently computer programmers have deliberately started to try use human brain power. Interactive evolutionary algorithms are one example of this, another is the program C3vision which makes an EEG of the user's brain activity while (s)he is quickly browsing images, and can detect when the user sees something interesting much faster than the user him/herself realizes it, thereby speeding up scanning of images tenfold[6]. The human processing capabilities can also be tapped via games, for example 'fold-it' (http://fold.it) which lets humans predict protein folding quite successfully, as predictions by fold-it players won seven prizes at the protein structure prediction contest CASP8, outperforming purely computational methods, and in one case even outperforming professional scientists.

What will the future be of such interactivity between man and computer? For now, humans seem to excel in combinatorial problems (such as finding the right folding for a protein, where calculating the energy score is probably rather fast) and problems which require knowledge which is hard to make explicit ('hunches'). Docking could undoubtedly be performed similarly to fold-it, at least when the docking scoring functions have become more accurate. And perhaps humans could be trained to get toxicity “intuition” by showing different structures and making them guess whether a compound is mutagenic or not, perhaps recognizing patterns which may be difficult to find by computer.

It may be that human-computer interactivity will one day be surpassed by computers which either can process data in a human-like way or are so fast that they won't need the speedup provided by human knowledge anymore. But such advances are only likely to happen in the very far future, if at all. For now, the more we learn about computers and problems, the more we find that computers are not yet the answer to all our problems and that human experts are unexpectedly potent problem solvers. Making human-computer interaction work well is a field of research in itself, but the better we become at it, the more powerful our capabilities in science and drug research will become, leaving 'computer-only' or 'human-only' techniques far behind.

In closing

Computational drug design has come far since the first QSAR programs and molecule databases, and the Molecule Evoluator is certainly not the end point of its ongoing evolution. There have been quite some successes already (for example with virtual screening), but we can still do better. In my opinion, software developers should strive to make drug design software grow in three directions: deeper, wider, and closer.

Deeper software will be software that more elaborately uses core scientific knowledge such as quantum mechanics, molecular dynamics and thermodynamics to improve the accuracy of predicting ligand affinity, compound metabolism and other phenomena. The main challenges here are integrating the diverse formulas and principles of chemistry and physics (for example, ligand affinity prediction also needs accurate prediction of the energy of (de)solvation of the ligand), and speeding up the calculations so that accurate enough results are obtained in the computer time available. *Wider* software, which focuses on combining and comparing data, may be able to help us where calculations are yet too slow – by recognizing patterns in experiments we may in some cases be able to predict important properties from 'data-based' models where 'computation-based' models are as of yet too time-consuming. Software casting a wider net over our current scientific knowledge may also help us find connections between different subjects, by integrating the data on chemistry, biology and genetics into one organized whole. Finally, developing *closer* software means that we should strive to make software more accessible to 'lay' users, so it can be used more easily by scientists who are not experts in computer programming. For some software products, this would involve creating Windows versions, for almost all programmers it would mean focusing much more on user-friendliness and the user's goals. Consciously seeking to develop the most intelligent task division between computer and user will also greatly increase the user's power to tackle scientific and drug design problems.

Developing deeper, wider and closer software may never bring 'perfect' computational drug design as was perhaps once envisioned. It is quite likely that it will create computer programs which will be totally unexpected, and tackle problems we may not even know about yet. Only one thing is certain: by working steadfastly on finding new ideas and evolving our software, we will be able to increase our understanding of biology and drug design to a depth we would now deem incredible. May that understanding be used effectively for the progress of science, and for the health of humanity.

References

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

[2] Barreiro, G.; Guimarães, C. R. W.; Tubert-Brohman, I.; Lyons, T. M.; Tirado-Rives, J.; Jorgensen, W. L. Search for Non-Nucleoside Inhibitors of HIV-1 Reverse Transcriptase Using Chemical Similarity, Molecular Docking, and MM-GB/SA Scoring. *J. Chem. Inf. Model.* **2007**, *47*, 2416-2428.

[3] Wilson, G. C.; McIntyre, A.; Heywood, M. I. Resource Review: Three Open Source Systems for Evolving Programs-Lilgp, ECJ and Grammatical Evolution. *Genetic Programming and Evolvable Machines* **2004**, *5*, 103-105.

[4] Schmidt, M.; Lipson, H. Distilling Free-Form Natural Laws from Experimental Data. *Science* **2009**, *324*, 81-85.

[5] Gavaghan, D.; Garny, A.; Maini, P. K.; Kohl, P. Mathematical models in physiology. *Phil. Trans. R. Soc. A* **2006***, 364,* 1099-1106.

[6] Sandhana, L. Man and machine vision in perfect harmony. *New Scientist* **2006**, *2559*, p30.