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

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

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

Abstract

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

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

Introduction

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

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

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

The difficulty of finding new drug molecules has pushed the pharmaceutical industry to try to improve its output using both experimental and computational methods. Of these computational methods, the Evolutionary Algorithms (EAs) are widely applied. Not only are they used to gain insight in the structures and functions of genes and proteins (Fogel 2004), which is important for discovering which proteins to target with a drug, they have also been applied to designing the drug molecules

themselves, such as designing compound libraries, docking small molecules into proteins, and finding structure-activity relationships. Good overviews are given in the book edited by Clark (Clark 2000) and in chapter 2 of this thesis. However, one of the most interesting applications is the design of new drug molecules themselves, the so-called *de novo* design.

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

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

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

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

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

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

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

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

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

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

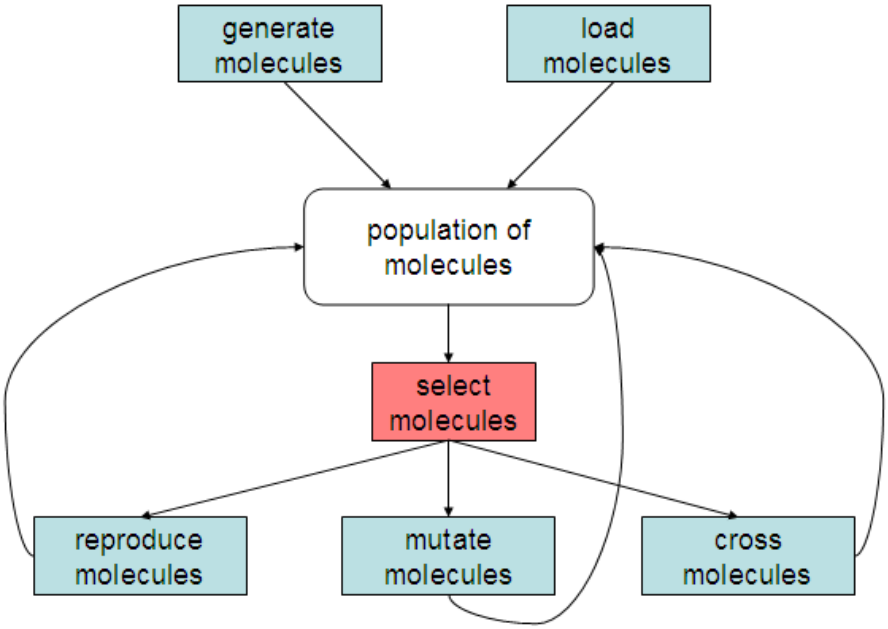


Figure 5.1: Overview of the evolutionary algorithm of the Molecule Evoluator.

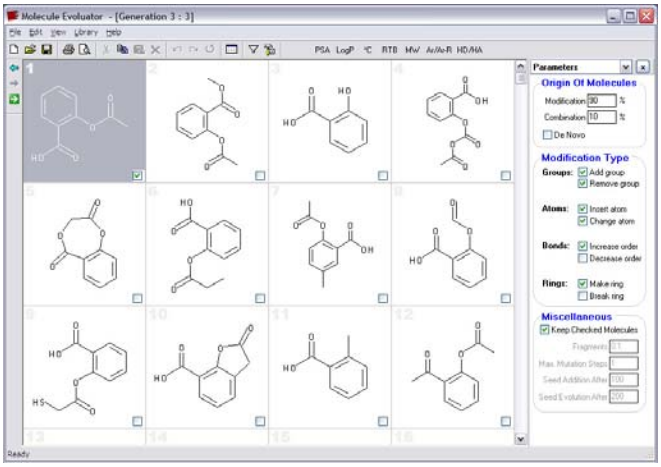


Figure 5.2: The Molecule Evoluator generating variants of acetylsalicylic acid (Aspirin^{TM}), the top left molecule.

Data Mining Approach

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

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

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

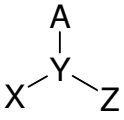
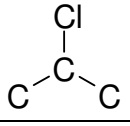
This gave us the idea to use a large database of drug molecules (the World Drug Index) to find the frequencies of all occurring substructures and adapt the mutation operators so that the mutants will be more drug-like and easier to synthesize than when using "uninformed" mutations.

We used the 2002 edition of the World Drug Index (Daylight 2005), containing approximately 32000 drugs and other pharmacologically active compounds. An in-

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

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

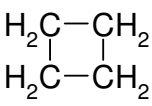
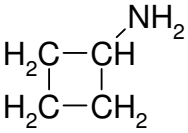
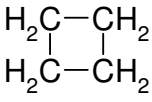
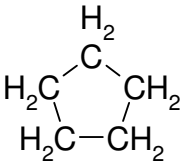
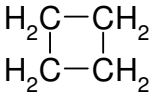
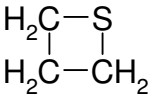
Table 5.1: The substructure patterns mined from the World Drug Index. The letters denote any possible atom type, the '-' characters all possible bond types, to wit single, double and triple bonds.

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

Adaptation of the Mutation Operators

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

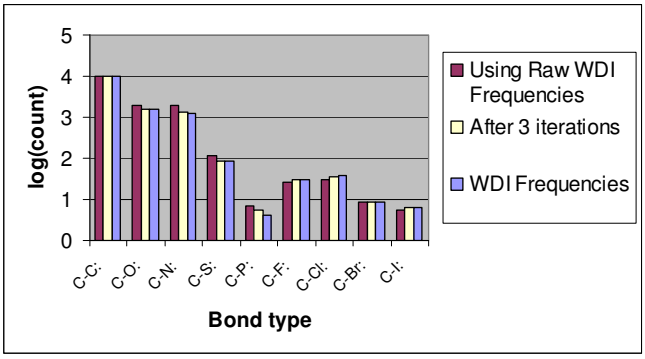


Figure 5.3: Iterative refinement results in the substructure counts getting closer and closer to those of the World Drug Index (all counts are scaled to make the count of CC-bonds 10000). Since the y-scale is logarithmic, some of the gains in accuracy seem smaller than they are: the number of C-N bonds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

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

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

Experiments

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

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

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

However, as it has been estimated that there are over 10^{60} molecules possible (Bohacek 1996), it seemed unreasonable to demand that our new algorithm would transform all existing NCI molecules into other existing NCI molecules. However, we can reduce the search space by splitting the molecules into fragments (figure 5.4). The

chance that a certain fragment is mutated into another known fragment is likely to be much larger than the theoretical molecule to molecule "mutation success ratio" of $2{,}5 \cdot 10^5/10^{60}$, since the smaller size of the fragments will greatly reduce the search space.

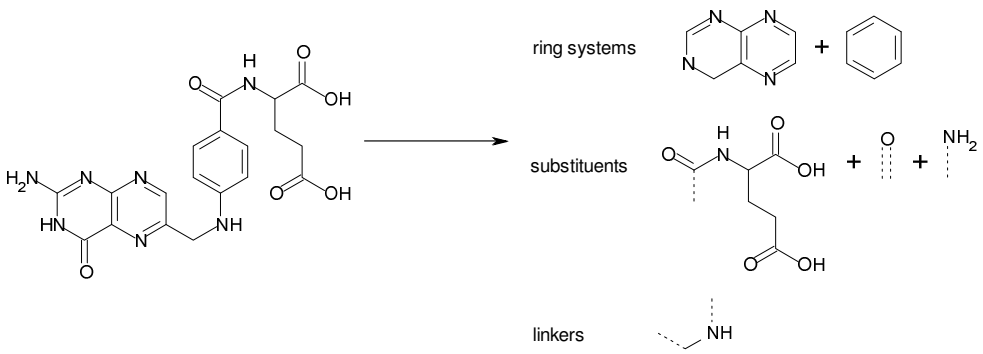


Figure 5.4: Splitting an example molecule, folic acid, into fragments.

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

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

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

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

Discussion

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

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

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

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

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

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

Conclusions

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

Acknowledgments

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

References

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Appendix: Comparing the old with the new mutation operators

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

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