The Molecule Evoluator: an Interactive Evolutionary Algorithm for Designing Drug Molecules

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

[1]Leiden/Amsterdam Center for Drug Research
Leiden University, Einsteinweg 55
Leiden, the Netherlands
+31-(0)71-5274660 ,
+31-(0)71-5274651
e.lameijer@chem.leidenuniv.nl,
ijzerman@chem.leidenuniv.nl

[2]Leiden Institute of Advanced Computer Science
Leiden University, Niels Bohrweg 1
Leiden, the Netherlands
+31-(0)71-52746517057,
+31-(0)71-52746517108
joost@liacs.nl,
baeck@liacs.nl

[3]NuTech Solutions
Martin-Schmeisser-Weg 15
44227 Dortmund, Germany
+49-(0)231-72546310
baeck@nutechsolutions.de

ABSTRACT

To help chemists design new drugs, we created a tool that uses interactive evolution to design drug molecules, the "Molecule Evoluator". In contrast to most other evolutionary *de novo* design programs, the molecule representation and the set of mutations enable it to both search the chemical space of all drug like molecules extensively and to fine-tune molecular structures to the problem at hand. Additionally, we use interaction with the user as a fitness function, which is new in evolutionary algorithms in drug design. This interactivity allows the Molecule Evoluator to use the domain knowledge of the chemist to estimate the ease of synthesis and the biological activity of the compound. This knowledge can guide the optimization process and thereby improve its results. Chemists of our department using the Molecule Evoluator were able to find six novel and synthesizable druglike core structures, indicating that the Molecule Evoluator can be used as a tool to enhance the chemist's creativity.

Categories and Subject Descriptors

J.2 [**Physical Sciences and Engineering**]: *chemistry,* J.3 [**Life and Medical Sciences**]: *health*.

General Terms

Algorithms, Design, Experimentation, Human Factors.

Keywords

Drug design, molecule, interactive evolution.

1. INTRODUCTION

In today's world, pharmaceuticals have a major impact on both public health and the economy. The pharmaceutical industry is a very large industry, with global sales of 491.8 billion dollars in 2003, estimated to grow to 496.6 billion dollars in 2004 [5].

Despite the large sales and the past and current successes in curing and alleviating diseases, the pharmaceutical industry is still looking for newer and better drugs. Most bacterial diseases can be cured and the life span of people with cardiac illnesses is greatly improved by cholesterol-lowering and blood-pressure normalizing medication. Yet viral diseases such as AIDS, mental diseases such as Alzheimer's disease, and diseases in which body cells themselves turn traitor, such as cancer, are still difficult or even impossible to fight, and even in the case of AIDS the virus can only be kept at bay at the price of serious side effects.

However, drug design continues to be a slow and expensive process in which it takes 10-15 years and about 800 million dollars to bring a drug to the market [7]. This is because it is difficult to find compounds which obey the strict criteria of safety and effectivity. It is therefore estimated that only one in about 5000 screened candidate compounds reaches the drug market [16].

To aid drug design, researchers in the pharmaceutical industry have been turning to computational methods. Among these methods are evolutionary algorithms, which have been applied as optimizers in several areas of drug design, such as designing compound libraries and finding structure-activity relationships. Evolutionary algorithms have played an especially interesting role in *de novo* design, the design of new compounds that could be suitable as a drug.

There have been several experiments in *de novo* drug design using evolutionary algorithms [8, 9, 10, 14, 15, 17, 19]. However, reviewing these experiments leads to the conclusion that despite the power inherent to global optimization methods, evolutionary algorithms are still difficult to apply to molecule design.

One problem in applying evolutionary algorithms to molecules is that molecules are graphs which have to obey certain chemical rules. Converting a molecule into a bitstring or fixed-size vector of numbers as used in conventional evolutionary algorithms will therefore probably result in mutations and crossover producing invalid molecules, which would need complex repair algorithms to correct.

The second major problem in applying evolutionary algorithms in drug design is obtaining a useful fitness function. While currently the most common methods use a similarity index to a reference compound [8, 9, 10, 17] or calculations of binding strength to a target [15, 19], only few examples have been published of successful applications of evolutionary algorithms to find new structures. Only Schneider [17] claimed success since his algorithm found a compound with a similar kind of activity as the lead compound, be it a 1000-fold less potent.

GECCO'05, June 25-29, 2005, Washington, DC, USA.

The main reasons for this only limited success seem to be that the compounds designed by computer are often not very easy to synthesize in the laboratory and that the fitness functions used are such poor approximations of structure-activity relationships that the noise in the fitness landscape makes true optimization impossible.

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

We believe that this approach can be a useful one. On the one hand the power of the computer can be used to perform a quick yet elaborate search of chemical space and suggest ideas which might have been overlooked by the mental blocks and prejudices of the chemist. On the other hand, the creativity and pattern-recognition capacities of the human chemist can ensure ease of synthesis and incorporate biochemical knowledge.

Based on this molecule representation and interactive fitness we designed a tool called the "Molecule Evoluator" to help medicinal chemists in drug design.

The overview of the remainder of the paper is as follows: we first discuss the representation of the molecules and the evolutionary operators, subsequently we will discuss how the user's choices affect the fitness of the molecules. We will then shortly describe the graphical user interface of the tool, together with the result of some of our experiments in interactive drug design.

2. MOLECULE REPRESENTATION

A molecule can be considered to be a connected graph consisting of one or more atoms (vertices) connected by bonds (undirected edges). One of the main rules of chemistry specifies that each type of atom has a particular number of bonds: for example, hydrogen atoms can only have one bond, while a carbon atom must have four. In some cases this is more complicated: atoms such as sulfur can have several valence states, of which bivalent sulfur and hexavalent sulfur are the most common. In our genotype we handle this by creating a separate symbol for each valence state, so that divalent sulfur atoms are denoted by "S" and hexavalent sulfur atoms by "Sh". So whether a graph represents a valid molecule does not merely depend on the structure of the graph, but also on the identity of each particular node, as can be seen in figure 1.

Chemists have created various ways to represent molecules on the computer. Nowadays, the two most common representations are the MOL-file format of MDL [6] which is an adjacency list representation of the molecule, and the SMILES-notation [20], a line notation/string that is human-readable and can easily be transformed into a 2D-structure by a chemist. SMILES encodes the nonlinear parts of a molecule with brackets to indicate branches and numbers to label rings, as is illustrated in figure 2. The hydrogen atoms are by default not incorporated into the SMILES notation, since their presence can be deduced from the rules of chemical valence.

Figure 1. Comparison of a valid molecule with an invalid molecule: Since C must have four, hydrogen one and oxygen two bonds, the first molecule is valid (it is commonly known as formaldehyde). The second molecule cannot exist: one H and the O have too many bonds, the N (which should have three) has too few bonds.

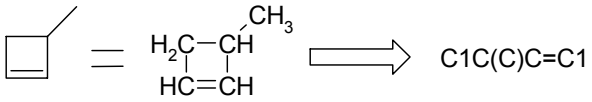


Figure 2. Example of the SMILES representation of a molecule. One bond of each ring is chosen at random (in this case, the bond from the top left to the bottom left atom) and is designated by a unique number. Both atoms participating in ringbonds get the bond number(s) immediately after them in the line notation. Branches are indicated by brackets.

When mutating and crossing molecules, however, neither of those representations is ideal. Since chemical valence rules explicitly state how many bonds any atom has, both the most common 2D-MOL-file format and the SMILES notation take the hydrogen atoms and the bonds to which they are attached for granted. We should therefore calculate for each atom whether it has the right number of hydrogen atoms to be mutated in some way. To avoid these recalculations, we decided to explicitly add the hydrogen atoms to the representation. Also, since a graph representation can be more difficult than a string representation to cross with another graph or to mutate, we decided to use a SMILES-like structure as our main molecule representation. Making the hydrogens explicit we get a bracket-rich, expanded SMILES (figure 3) to which we can apply mutation by relatively simple algorithms. We coin this notation "TreeSMILES".

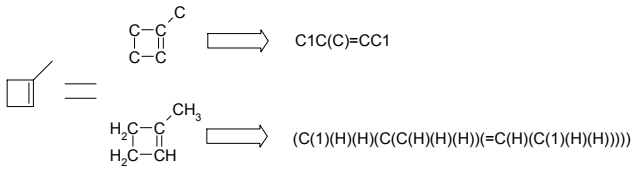


Figure 3. SMILES versus TreeSMILES. By making the hydrogens explicit the notation becomes less compact and less human-readable. However, a computer can now more easily see which positions can undergo a particular mutation.

3. CROSSOVER AND MUTATION

Crossover is implemented like crossover in standard genetic programming [2]: subtrees of two different molecules are selected and swapped. The only complication in this case is that the trees represent graphs and can contain cycles, and subgraphs are not allowed to contain incomplete cycles. So when a subtree at a random "root" atom is selected, the subtree is first checked for unmatched ring bonds, and if these are present the current root is discarded and another subtree is selected for crossover.

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

The implemented mutations, graphically shown in table 1, are as follows:

1) Add atom/group: this replaces a hydrogen atom in the molecule with a non-hydrogen atom or a larger chemical group such as a phenyl group. In the case of atom addition, the remaining bonds of the added atom are filled with hydrogens.
2) Insert atom: adds an atom by inserting the new atom (which should have a valence of two or higher) into a bond. The remaining bonds of the new atom are fulfilled by adding hydrogens to it.
3) Delete atom: this removes an atom that is attached to only one non-hydrogen atom (with a single bond) by first deleting the hydrogen atoms attached to it, and renaming the atom to a hydrogen atom.
4) Uninsert atom: This removes an atom that has exactly two non-hydrogen neighbours. It removes the atom and its hydrogens and subsequently creates a bond between its two neighbouring non-hydrogen atoms.
5) Increase bond order: if two atoms which are bonded to each other both have at least one hydrogen atom, those hydrogen atoms are removed and an extra bond is created between the atoms (the bond order is increased from single to double, or from double to triple).

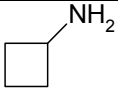
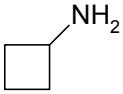
Mutation name	Initial structure	Final structure	Initial TreeSmiles	Final TreeSMILES
Add atom			...(C(H)(*H*)(C…	...(C(H)(**N(H)(H)**)(C…
Insert atom			...(C(H)(H)(C...)...	...(C(H)(H)**(N(H)**(C...)**)**...
Delete atom			..(C(H)(*N(H)(H)*)(C...	...(C(H)(**H**)(C...
Uninsert atom			...(C(H)(H)*(N(H)*(C...)*)*...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

Table 1. Schematic overview of the different mutations in the Molecule Evoluator. Most leave the TreeSMILES string fairly intact and can be performed by string editing. The only exception is the "break ring" mutation, which can substantially rearrange the TreeSMILES.

6) Create ring: similar to increase bond order, but works between two atoms which are not bonded to each other. These atoms are connected using a single bond (the two hydrogen atoms are changed into ring indices).

7) Decrease bond order: if there is a double or triple bond between two atoms, its bond order is decreased by one and a hydrogen atom is attached to each of the two atoms.

8) Break ring: the "break ring" mutation chooses a single bond in a ring, breaks that bond and adds hydrogen atoms to correct the valences. The algorithm should be able to break any bond in the ring, which is not easy to do with a tree structure [14]. To solve this problem, we converted the TreeSMILES into an adjacency list. In the adjacency list, any ring bond can be broken easily and afterwards a new TreeSMILES can be built. This is the only mutation where we found it necessary to temporarily convert the TreeSMILES representation of the molecule into an adjacency list format for easier modification. Since other operators such as crossover are much more easily implemented for a TreeSMILES string, we decided not to use the adjacency list for the other mutations. Changing representations is probably a convenient way to accommodate different mutations, though to our knowledge this technique has not been used before in evolutionary algorithms in *de novo* design.

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

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

4. FITNESS

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

1) Similarity to a target molecule [8, 10]

2) QSAR-functions [14]

3) Docking [15, 19]

4) Experiment [11]

For drug design, each of these methods has its advantages and disadvantages. Similarity to a target molecule results in molecules very similar to the target molecule, in many cases even the target molecule itself, which is not useful for designing new molecules. Nevertheless, one of the few successes of evolutionary algorithms in *de novo* design has come from such an approach [17], though finding a molecule that binds a thousand fold worse than its example is maybe not a big success. QSAR-functions should be able to optimize activity, yet have important disadvantages. First they require quite a lot of reprogramming for each new class of molecules that is to be investigated. Secondly, they are generally difficult to use for finding very active compounds since they grow less and less reliable as the molecular structures deviate more from the average structure (and thereby activity) of the training dataset [18]. Unfortunately, this is exactly what would happen during optimization. Docking (calculating how well the molecule fits in the target protein) is still too inaccurate for optimization, and usually results in molecules which have an activity which is 100-1000-fold lower than the calculated value [19]. Experiments, finally, are generally slow and expensive, and so far have only been performed for a class of molecules which are especially easy to synthesize but are not druglike [11]. Thus experimental fitness has yet to prove to be practical in a more realistic drug design scenario.

As a different approach, we decided to use the *user* as a fitness function. This concept has been applied before in interactive evolution [1, 3]. While a user cannot know the binding strength of a given molecule, this defect is not much worse than the inaccuracy of scoring functions. Another advantage in letting the user choose would be that intensive feedback from a medicinal chemist would make the compounds easier to synthesize, and steer the evolution away from areas which have already been explored. Furthermore, the algorithm could still be easily coupled to experimental results or advanced computed fitness functions if so desired.

Since a user cannot evaluate as many structures as a computer program and preliminary experiments have shown that users only want to see “good” structures, we added several descriptor calculations to the Molecule Evoluator. The selected descriptors are the number of hydrogen donors/acceptors, the molecular weight, the logP (lipophilicity), the polar surface area, the number of rotatable bonds, and the number of aromatic systems and substituents. The number of hydrogen donors/acceptors, the molecular weight and the logP are commonly considered to reflect how well a compound can pass the gut wall and enter the body and thus be more ‘drug like’, after Lipinski [13]. The polar surface area is a more modern descriptor that can also be used to predict this passage. Upper and lower bounds for all these descriptors can be set by the user as a filter to create molecules which are more drug like. Additionally, we implemented some filters to eliminate molecules with chemically undesirable structures, such as so-called paracyclophanes.

The Molecule Evoluator can use two kinds of evolution. The normal evolution uses the population viewed by the user, which consists of twelve molecules, and lets the user select the best molecules, which are subsequently either mutated or crossed over. If the user wants to generate a large library of molecules which obey certain strict fitness criteria, an internal evolution is used with a population of 50 molecules, of which the best 25% (the fitness is the sum of squares of the deviations from the constraints, scaled for each physicochemical property by its range over the molecules of the population) are selected by tournament selection (tournament size 2), and are mutated or crossed for the next generation. This evolution lasts until a molecule has been found which obeys the filters. Since computing times so far are acceptable to the users, the evolution parameters have not yet

been fine-tuned for optimal performance. However, this might be a interesting subject to be investigated further.

5. THE INTERFACE OF THE MOLECULE EVOLUATOR

When the user starts the Molecule Evoluator, he or she can create the initial population of the program by drawing or loading molecules. Alternatively, the Molecule Evoluator itself can initialize the population with random molecules .

After the user presses the "Go" button, a window appears that contains the selected old molecules (elitism is on by default) together with the newly generated molecules (Figure 8). The user can again select the most attractive molecules, press “Go”, and this process is repeated until the user has gathered enough ideas.

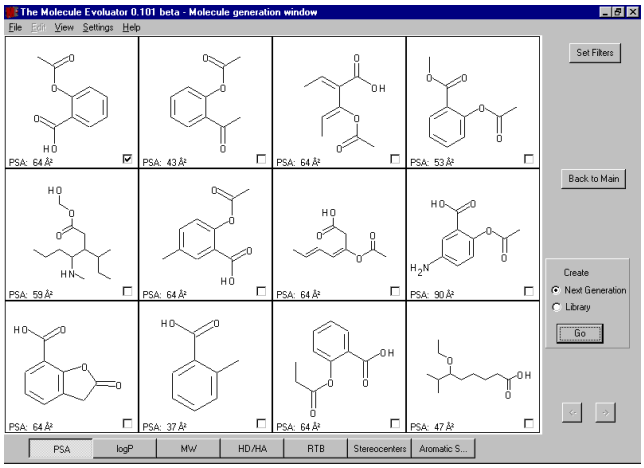


Figure 4. Pressing “Go” generates mutants. Elitism (optional) conserves the original molecule, and some molecules are generated from scratch (like that in the bottom right corner). Most molecules however are mutants of the original molecule; the one in the bottom left corner is a ring closing mutant.

Comments from medicinal chemists have led us to add three extra features that give the user more control over the evolution: editing the molecules directly, fixing parts of the molecule, and using filters to prevent that unsuitable molecules are shown to the user. We next discuss these features in more detail.

Editing molecules is useful when the user wants to start the evolution with a molecule that has not been stored yet in the computer and must be drawn. Additionally, if during the evolution the user sees a molecule which inspires him/her to a better or more interesting structure, he or she can edit the molecule into the desired structure. This will allow the user to evolve molecules immediately from the desired structure, instead of having to wait until it is finally generated by the program. Editing the molecules is performed in the "Molecule Edit" window, which pops up when the user clicks on a molecule in the main window. Editing is similar to that in normal chemical drawing programs such as ISIS/Draw [21], be it that only the basic facilities (adding, removing and changing atoms and bonds) are supported. After the popup window has been closed, the drawn structure is converted into TreeSMILES-format.

Fixing part of the molecule can be useful in cases where knowledge of structure-activity relationships might make the user want to ensure that a particular, necessary part of the molecule is present in all its descendants. The Molecule Evoluator allows this conservation with the “fix atoms/bonds” option, which enables the user to generate new molecules with the conserved part constant, and only variation on the “free” atoms.

The third extra feature for user influence is the “Filter Window”. In the “Physical Filters” the ranges are set in which the physicochemical properties of a molecule must lie for the molecule to be incorporated into the population (for example: molecular weight between 100 and 400). Molecules which for example have too many rotatable bonds (and will probably bind weakly and aselectively) can be automatically eliminated by the Molecule Evoluator and will therefore not be shown to the user. Additionally, some chemical structures which are usually undesirable, such as hemiketals, can be forbidden in the “Chemical Filters”. The Molecule Evoluator creates offspring molecules using mutation and crossover until feasible molecules – fulfilling all filter conditions – have been found.

In addition to these three main control features, there is an “Evolution Parameters” window in which the user can influence the evolutionary process itself instead of the molecules. Via this window the user can steer evolution by, amongst others, enabling/disabling certain kinds of mutations. For example, the “decrease bond order” mutation tends to partially reduce phenyl rings, which is chemically undesirable. Disabling this mutation (as an alternative to fixing the phenyl bonds explicitly) will protect the bonds from being reduced. This will however also prevent useful mutations, such as those which reduce a ketone (C=O) to an alcohol (CHOH). An alternative would be a special version of the "decrease bond order" mutation that does not reduce aromatic rings, this might however just make the program harder to understand and use.

The “Evolution Parameters” window (Figure 5) has several options to influence the evolution.

The main group of parameters decides the relative amounts of crossover and mutation, which can be set between 0 and 100%. Since in most cases mutation is preferred over crossover, the default settings are mutation 80% and crossover 20% (and so are applied with probabilities of 0.8 and 0.2 respectively).

The second option is whether it is allowed for the Molecule Evoluator to occasionally add random molecules to the population. The relative amount of random molecules is approximately 16% (so 1-2 new random molecules in a new population). This option is on by default.

Thirdly, the user can toggle elitism on and off. Elitism conserves the selected molecules in the next generation and makes them also the first molecules on the screen, so the user can quickly see which were the source molecules.

The other user-controllable parameters are fraction of fragments, which determines how many of the “add group” mutations add a functional group instead of an atom (default 0.1 = 10%), the “inducing evolution”-limit: how many times creating a random molecule/mutant is tried before unsupervised evolution is started to find a molecule that obeys all filters, and finally the “number of mutation steps allowed”, which allows the user to specify how many mutations the molecule is allowed to undergo before it is

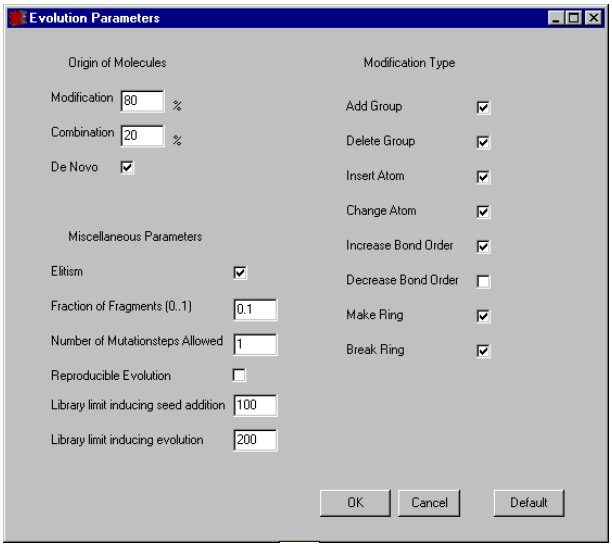


Figure 5. The "Evolution Parameters" window.

placed in the next generation. This option increases the step size of the evolutionary algorithm.

The mutations "add group", "delete group", "insert atom", "change atom", "increase bond order", "decrease bond order", "make ring", and "break ring" can be toggled on and off. In the window the mutations "delete atom" and "uninsert atom" are fused under the heading "delete group", since they are strongly related chemically. By default, each of these eight categories of mutations is applied with the same frequency (0.125). When mutation types are disabled, the remaining active mutation types are still applied with identical frequency, so if only three mutations are checked, the probability of each of them is 0.33.

6. EXPERIMENTS

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

First we generated a library of 10000 molecules with druglike features: either one or two aromatic rings, 5 or fewer rotatable bonds, 2 or fewer hydrogen donors, 4 or fewer hydrogen acceptors, a polar surface area of at most 70Å^2 and a molecular weight between 150 and 500.

Out of this library, three sublibraries of 100 compounds were chosen randomly. Each of these sublibraries was presented to a different chemist, who could choose and modify the molecules created by the program. Out of the 300 compounds, 35 were chosen for further investigation.

Checking the molecules in the Beilstein database (over nine million compounds, contains almost all molecules which have been synthesized so far), we found that six structures represented chemical classes yet unknown in literature. Based on these six core structures ten derived structures were designed.

From these structures eight compounds were synthesized successfully. This procedure appeared highly interesting to one of our industrial partners. They offered to have the compounds tested on more than 80 drug targets. This evaluation is currently taking place.

In a small experiment, we used a dataset of biological activities of neuramidase inhibitors [12]. Using the measured activities as input for the evolutionary algorithm we found the experimental minimum (1 nM, a 6300-fold improvement over the original structure) within four generations.

Finally, we were able (without using the edit function) to evolve drug molecules such as acetylsalicylic acid (in 12 generations), diazepam (in 65 generations) and quinidine (in 22 generations). This provides evidence that our set of mutations is sufficient to transform a random starting molecule into a drug molecule and would suggest that our algorithm can access the major part if not the whole of the chemical space of drug molecules.

7. DISCUSSION

In this paper we have presented an evolutionary algorithm to help design new molecules. The literature in this area, using evolutionary algorithms to design new (drug) molecules, is quite extensive, with a large variety in methods used. The two most important components of all these methods are the molecule representation and the fitness function.

7.1 The molecule representation

One of the main choices made by investigators is whether to make their algorithm atom-based or fragment-based. Atom-based algorithms work by mutating atoms, and can therefore fine-tune each structure optimally. This approach has been chosen in many articles [4, 8, 10, 14]. On the other hand, several investigators construct molecules using larger fragments [17, 19]. This has the advantage that the representation can be simpler, since there is generally no need for the genome to contain cycles (for these are incorporated into the fragments) with the additional advantage that the compounds would be easier to synthesize than the molecules generated by the atom-based methods. While the current version of the Molecule Evoluator uses both atoms and fragments to construct molecules, its mutations are atom-based. We believe that atom-based evolution is superior to fragment-based evolution for adapting the molecular structure. The main disadvantage of using fragments instead of atoms is that most mutations in fragment-based evolution are macromutations which change the molecule into something completely different, with a vastly different fitness value. In most cases, it is not clear whether fragment-based evolution improves over random search, unless the fitness function is fragment-based. However, this is certainly not the case in drug design where biological activity is subtly dependent on the molecular structure. We expect that making an atom-based algorithm interactive, as in the Molecule Evoluator, will partially compensate for the disadvantage that molecules generated on the basis of atoms are generally more difficult to synthesize, for the chemist could discard or modify structures at will.

7.2 The user as fitness function

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

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

One disadvantage of user interaction is that the population must be small. It is unlikely that any chemist would want to see 50 to 100 molecules before pressing "next". These small population sizes (12) may lead to premature convergence.

The second disadvantage is that the more the user can interact with the program, the more is required from the user interface. In this project, more time was spent on constructing the user interface than on creating and fine-tuning the evolutionary algorithm. Modifications of the evolutionary algorithm should in many cases be reflected by changes in the user interface, and this makes programming and testing new ideas more time-consuming than in a non-interactive system.

A third disadvantage is that testing is more difficult – one cannot well run hundreds of tests automatically to objectively verify whether the algorithm outperforms other algorithms. A user is not an objective function that can be easily shared with others. While it might be possible to compare the idea generation rate of chemists using the Molecule Evoluator to chemists not using the Molecule Evoluator, scores are likely to vary greatly per individual and per molecule to be optimized.

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

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

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

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

7.3 Adding extra user control to the evolution

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

Medicinal chemists are still testing the Molecule Evoluator and its features. One of the most interesting comments so far was that chemists liked that while a molecule is edited, the changes in its physicochemical properties are shown on screen. While this praise is not related to the evolutionary algorithm, it does suggest that there are ample opportunities for improvement in the current chemical software.

The Molecule Evoluator is currently being beta tested by several pharmaceutical companies.

8. CONCLUSIONS AND FUTURE PERSPECTIVES

In this paper we have described the "Molecule Evoluator", a program based on evolutionary algorithms that has been created to aid chemists in designing new drug molecules. With this program all relevant chemical mutations are possible. The most

distinguishing feature of the Molecule Evoluator relative to other de novo design programs is having the user as fitness function, which can combine the domain knowledge of the chemist with the memory and processing speed of the computer. We therefore added a graphical user interface for the evolution and extended the program with options for directly editing the molecule, marking part of a molecule as conserved, and calculating relevant physicochemical parameters.

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

9. REFERENCES

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

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

[3] Bentley, P.J. *Evolutionary Design by Computers*, Morgan Kaufmann Publishers, San Francisso, CA, 1999.

[4] 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.

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

[6] Dalby, A., Nourse, J.G., Hounshell, W.D., Gushurst, A.K.I., Grier, D.L., Leland, B.A., and Laufer, J. Description of Several Chemical Structure File Formats Used by Computer Programs Developed at Molecular Design Limited. *J. Chem. Inf. Comput. Sci. 32* (1992), 244-255.

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

[8] 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.

[9] 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.

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

[11] 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.

[12] Kim, C.U., Lew, W., Williams, M.A., Liu, H., Zhang, L., Swaminathan, S., Bischofberger, N., Chen, M.S., Mendel, D.B., Tai, C.Y., Laver, W.G., and Stevens, R.C. Influenza Neuramidase Inhibitors Possessing a Novel Hydrophobic Interaction in the Enzyme Active Site: Design, Synthesis, and Structural Analysis of Carbocyclic Sialic Acid Analogues with potent Anti-Influenza Activity. *J. Am. Chem. Soc 119* (1997), 681-690.

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

[14] 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.

[15] 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.

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

[17] Schneider, G., Clément-Chomienne, O., Hilfiger L. Scheider, 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.

[18] Sheridan, R.P., Feuston, B.P., Maiorov, V.N., and Kearsley, S.K. Similarity to Molecules in the Training Set Is a Good Discriminator for Prediction Accuracy in QSAR. *Journal of Chemical Information and Computer Sciences, 44* (2004), 1912-1928.

[19] 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.

[20] Weininger, D. SMILES: a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. *J. Chem. Inf. Comput. Sci. 28* (1988), 31-36.

[21] http://www.mdli.com/downloads/public/ctfile/ctfile.jsp