



Universiteit  
Leiden  
The Netherlands

## Interactive evolutionary algorithms and data mining for drug design

Lameijer, E.M.W.

### Citation

Lameijer, E. M. W. (2010, January 28). *Interactive evolutionary algorithms and data mining for drug design*. Retrieved from <https://hdl.handle.net/1887/14620>

Version: Corrected Publisher's Version

License: [Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden](#)

Downloaded from: <https://hdl.handle.net/1887/14620>

**Note:** To cite this publication please use the final published version (if applicable).

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

*Eric-Wubbo Lameijer,<sup>†</sup> Chris de Graaf,<sup>‡</sup> Daan Acohen,<sup>†</sup> Chris Oostenbrink,<sup>‡</sup> and Ad P. IJzerman<sup>†,\*</sup>*

Leiden/Amsterdam Center for Drug Research, Division of Medicinal Chemistry, Leiden University, PO Box 9502, 2300RA Leiden, The Netherlands, and Leiden/Amsterdam Center for Drug Research, Division of Molecular Toxicology, Department of Chemistry and Pharmacochimistry, Vrije Universiteit, De Boelelaan 1083, 1081 HV Amsterdam, The Netherlands.

## Abstract

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

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

## Introduction

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

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

Evolutionary algorithms have been applied quite often to *de novo* molecule design (Payne, 1995; Nachbar, 1998; Globus, 1999; Douguet, 2000; Schneider, 2000; Pegg, 2001; Vinkers, 2003; Brown, 2004; Dey, 2008; Nicolaou, 2009), and most authors have claimed some success in optimizing molecular structures. However, the

abundance of methods hides the fact that we actually know very little about what kind(s) of evolutionary algorithm should be used for drug design.

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

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

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

choice. We therefore chose to investigate this factor.

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

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

It is not clear from existing literature whether atom-based or fragment-based evolution should be preferred, as no study has compared them yet on the same problem. The main advantage of fragment-based evolution over atom-based evolution is that the produced structures seem easier to synthesize. While this apparent ease of synthesis can be quite deceptive in practice (Vinkers, 2003), it is certainly preferable over the much larger chance that a molecule created by atom-based algorithms needs major modifications before synthesis can take place (see for example the work of Douguet (Douguet, 2000)). On the other hand, atom-based evolution has advantages too. First of all, atom-based evolution does not require one to make the—always somewhat haphazard—choice of building blocks and attachment rules. Secondly, atom-based evolution can in theory construct all possible molecules, in contrast to the fragment-based approach which can typically “only” create  $10^{10}$  to  $10^{20}$  (Pegg et al. estimate a lower bound of  $10^{12}$  for their HIV-RT library (Pegg, 2001)). Being able to create  $10^{12}$  molecules may look impressive, however it represents only a tiny fraction of the total number of possible drug-like molecules, estimated to be  $10^{60}$  or higher (Bohacek,

1996). This implies that fragment-based evolution can only construct about 1 of every  $10^{40}$  possible molecules, which is far less than one molecule in a database containing all compounds currently known to man (which would have about  $10^7$ - $10^8$  entries, as the largest compound databases such as CAS (CAS, 2009), Beilstein (Beilstein, 2009) and PubChem (Pubchem, 2009) contain about 48 million, 10 million and 19 million entries respectively as around June 2009. Thirdly, the big jumps taken by fragment-based evolution through changing 5-10 atoms simultaneously may make optimization much more difficult than when only one or two atoms are changed at a time. This could be compared to carving a statue with a pickaxe instead of with a chisel: it is almost impossible to exert the precise and subtle control needed for creating the exact result one wants. Also, having fewer choices may result in fewer pathways to get to one's destination, and more dead ends.

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

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

The fitness of the molecules during the evolution was defined as their docking score in HIV-RT. While docking scores are not very accurate for predicting binding affinity of a potential drug to a protein, of all evaluation methods it seems best suited to base our investigations on. The highly complex interactions between a flexible ligand and a heterogeneous, irregularly shaped cavity, as modeled by docking, resemble the biological binding process more than for example 2D similarity to known ligands does. We do not expect docking in combination with the evolutionary algorithms described in this paper to result in 'true leads' for RT-inhibitors, as docking algorithms are not yet accurate enough for that purpose. However, for an investigation into making

evolutionary algorithms as efficient as possible in designing drug-like molecules, docking is probably a good approximative method to weed out inefficient evolutionary algorithms, before performing the final fine-tuning with the more realistic but also much more expensive and time-consuming synthesis and biological testing of compounds. One could compare this to first testing a probe for exploring the planet Mars in a cold desert on earth: while the circumstances of the test are not totally accurate, it is a much easier, faster and cheaper method to detect flaws than simply sending the design to Mars for the real test.

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

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

## Methods

### Description of the algorithm

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

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

- A) Create initial population
  - while the initial population contains fewer than 50 molecules:
    - generate a molecule out of NCI fragments
    - add this molecule to the population if it satisfies the physicochemical restraints.
- B) Perform evolution
  - while there are fewer than 10 generations
    - calculate the fitness of each molecule in the current generation by docking it into HIV-RT
    - if the current generation is not the initial generation:
      - select 10 molecules by tournament selection from the previous generation
      - add these molecules and their fitnesses to the current generation
    - make a new generation next to the current generation.
  - while there are fewer than 45 molecules in the new generation
    - 1] choose one molecule from the current generation by tournament selection.
  
    - 2] choose crossover or mutation
      - if crossover:
        - select a second molecule from the current generation by tournament selection
        - cross the two molecules, and pick one of the products at random
        - if crossover cannot be performed since there are no suitable breaking points, go back to step 1.
      - if mutation:

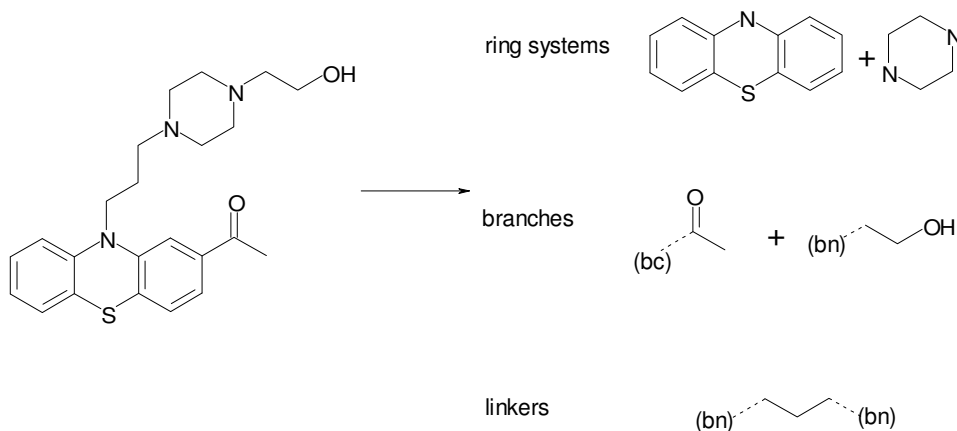


```
-pick a mutation type at random
-perform the mutation on the selected
  molecule
-if the mutation cannot be performed due to
  lack of suitable atoms or bonds, go back
  to step 1.
-if the resulting molecule obeys the
  physicochemical constraints and was not part of a
  previous generation:
  -add the new molecule to the new generation
-endwhile
-for j=1 to 5
  -generate a molecule at random
  -if the molecule obeys the physicochemical
    constraints:
    -add the molecule to the new generation
  -rename the current generation to "previous
    generation", and the new generation to "current
    generation"
-endwhile
```

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

Splitting the molecules of the NCI database across ring attachment bonds also resulted in data about their fragment composition. For example, 7.5% of all molecules consisted of exactly two ring systems, two branches, and one linker connecting two fragments, like the acetophenazine molecule in Figure 6.1. This information was also used in constructing the random molecules of the initial population. Each molecule was created by first picking a fragment composition (for example, a 7.5% chance to have the "2-2-1" composition), and then picking the fragments themselves out of the 300 most occurring ring systems and the 300 most occurring non-ring systems (branches

and linkers). The probability of a particular fragment being selected was proportional to its occurrence in the NCI. Since our database however also incorporated data to which atom type each non-ring fragment was attached, initially selected fragments for which there was no suitable atom type in the rings to connect to were replaced by other fragments. For example, the keto group created by dissecting the molecule in Figure 1 would be stored as (bc)-C(=O)C, to indicate that it can only be re-attached to a carbon



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

atom, and not to for example a nitrogen atom in piperidine. In some cases, the atom type demanded by a particular fragment was not available in the fragments that should be linked to it: for example, a methyl fragment that in the NCI was attached to a nitrogen atom ((bn)-CH<sub>3</sub>) could not be attached to a phenyl ring (other fragments, like (bc)-CH<sub>3</sub>, however, could).

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

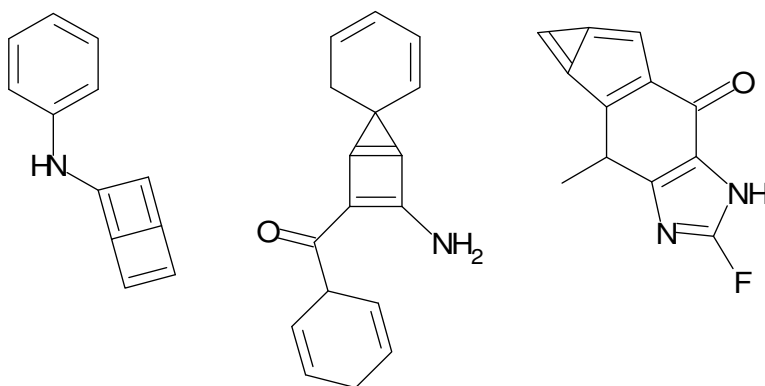
After the initial population was generated, it was evolved (via atom- and fragment-based evolution) as follows. First the fitness scores of all molecules were calculated by docking them with GOLD in the HIV-RT crystal structure and selecting the best ranked pose for every molecule (the preparation of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

Since we did not want to evaluate molecules more than once, we introduced the restraint that molecules created by mutation and crossover would only be added to the new population if they had not occurred yet in previous generations. This could, however, hamper optimization, since the best molecules would be forgotten after one generation. In contrast, most evolutionary algorithms allow a good individual to survive indefinitely and to even clone itself, so it can eventually fill the entire population. We tried to avoid such loss of diversity yet conserve the memory of the best molecules by adding some old molecules to each generation. After the 50 molecules of a population were generated in the normal way, we added 10 of the best molecules of the previous generation (selected by tournament selection), making the effective population size 60. The next generation was created by mutating and crossing those 60 molecules. So, in principle, a superior molecule from generation 1 could contribute its genetic information directly to several generations by first being a parent for the molecules in the next generation (generation 2), and subsequently be added to the generation of its children by being chosen as one of the ten grandparents (so forming the extended generation 2 together with its direct offspring). In this way, it can

produce offspring into the generation of its grandchildren (generation 3). As a member of the extended generation 2 it could be also added to the generation 3 as one of the best of the generation 2, into the extended generation 3, becoming one of the sixty parents for the molecules in generation 4. While the randomness of tournament selection makes it unlikely that any molecule, no matter how good, will remain producing offspring for more than three generations, it will buy the best molecules two or three generations extra time to produce high quality offspring.

### Manual filtering of molecules

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



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

## Molecule modifications

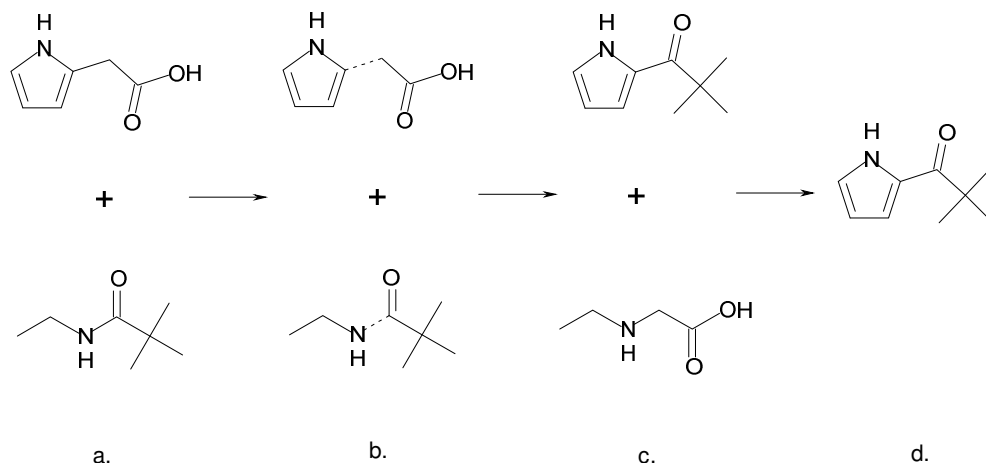
All evolution experiments were performed with the Molecule Evoluator (chapter 4), which was adapted to include fragment-based mutations and crossover. Both the atom-based mutations (which were already present in the Molecule Evoluator) and the fragment-based mutations (which were added during this investigation)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

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

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

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

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



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

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

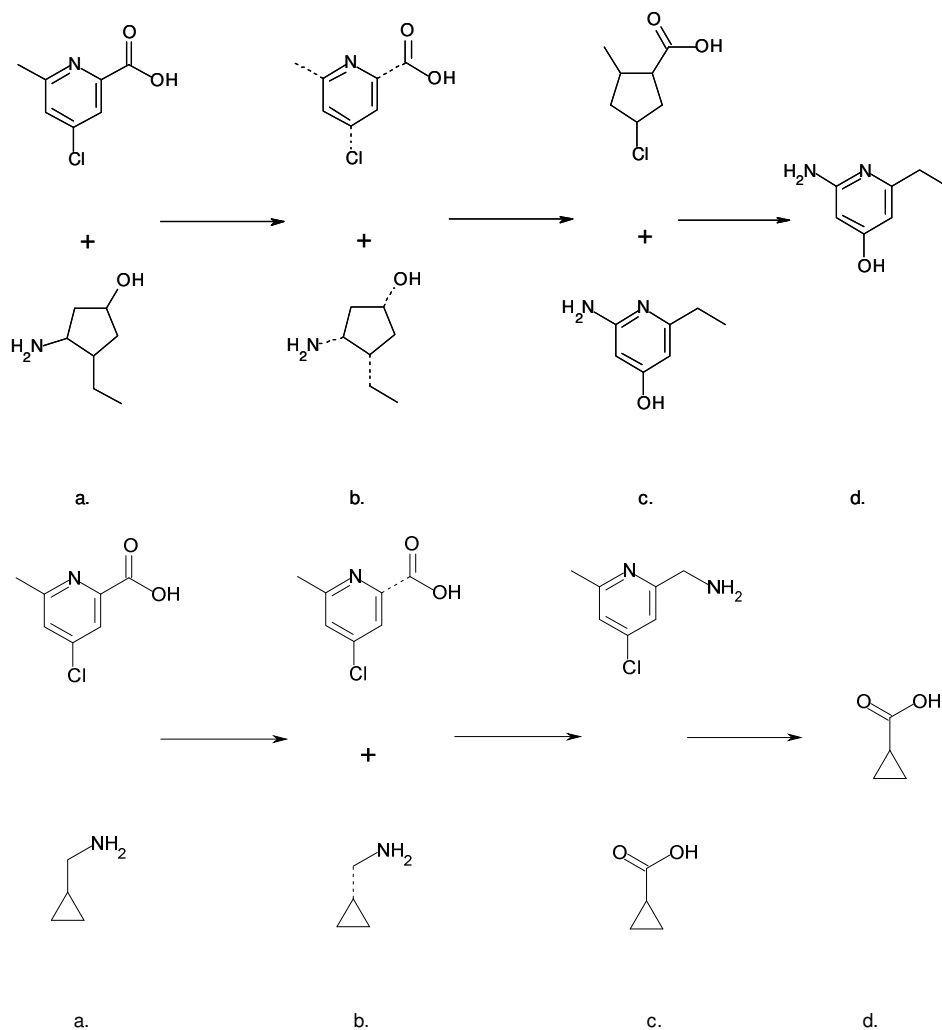
In both atom-based and fragment-based evolution, the mutation:crossover ratio was set to 85:15, based on experience with the interactive version of the Molecule Evaluator. All mutation subtypes were applied with equal probability; so the probability of applying one of the nine atom-based mutations, such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

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

Mutation type	Initial structure	Final structure
Rotation		
Exchange		

### Protein preparation and automated docking simulations

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



**Figure 6.4:** Fragment-based molecule crossover, which can both be performed on ring systems (top) and on branches (bottom). a) Two parent molecules are selected, b) the molecules are split: in both molecules a random fragment of the same type (branch or ring) is selected and the bonds between the chosen fragment and the rest of the molecule are broken c) the molecule parts are recombined, d) one molecule is selected at random as the crossover product. Note that the  $\text{CH}_2\text{NH}_2$  group could also have been exchanged with the methyl or chlorine, the pyridine-cyclopropane interchange is however not possible since the ring systems have one and three groups attached, respectively.



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

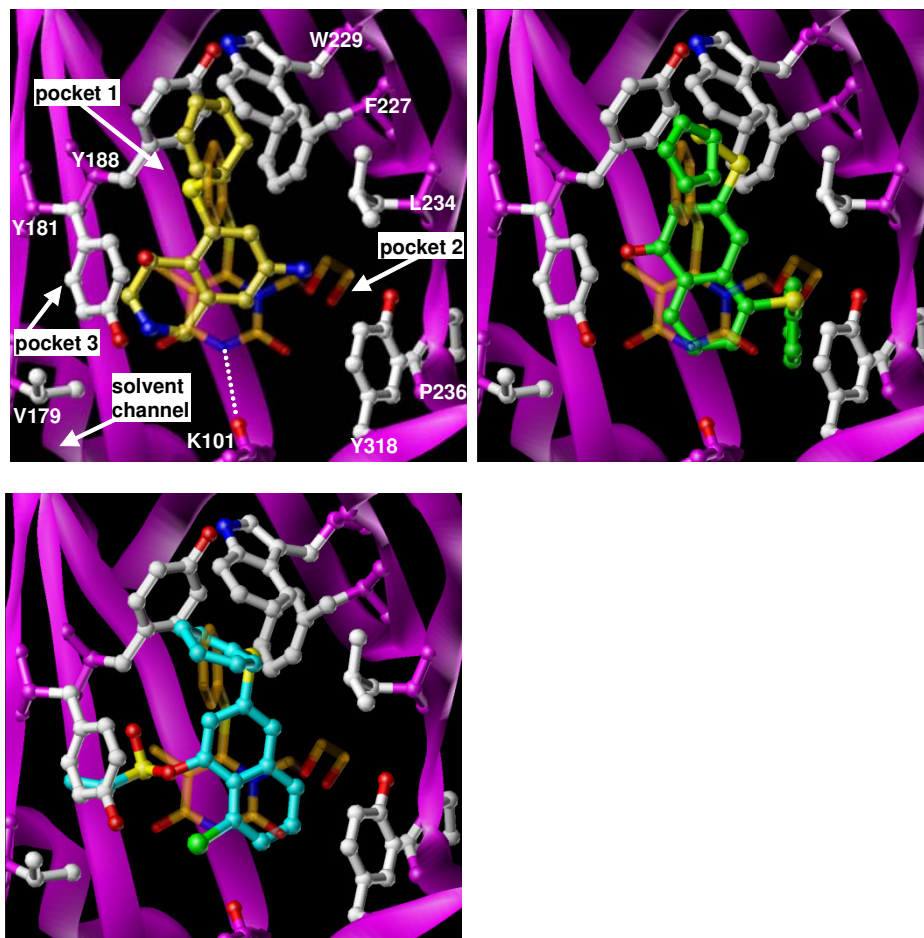
### **Estimating ease of synthesis**

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

## **Results and discussion**

### **Docking the molecules into HIV-RT**

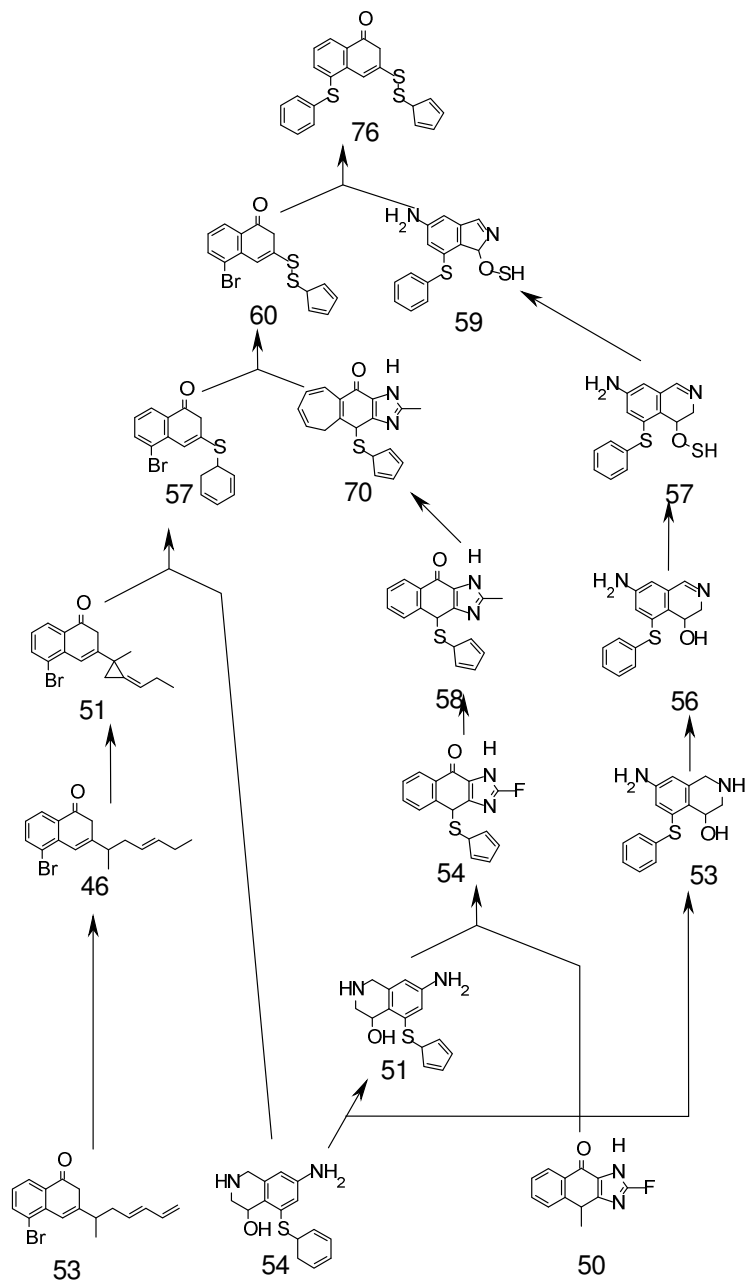
Docking the EA-generated compounds in the HIV-RT crystal structure resulted in the dockings shown in Figure 5. The binding pocket can be divided into a large hydrophobic subpocket 1 (enclosed at the top by residues Y188, W229, F227), and two small subpockets 2 (formed by L234, P236, and Y318) and 3 (formed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



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

respectively, in addition to subpocket 1. None of the compounds formed H-bonds with the backbone of K101 and/or K103. However, hydrophobicity is another key feature driving the potency of inhibitor binding to the NNRTI site (while GoldScore does not have a separate term for hydrophobic interactions, it calculates van der Waals

interactions and therefore assigns a higher score to a ligand that better fits the three-dimensional structure of the binding pocket). Optimization of hydrophobic interactions in pocket 1 is already observed in the first generation cycle (see the left panel of Figure 6.5), in which an S-phenyl group attached to a two-ring ring system dips into the aromatic cluster of Y188, W229, F227. A two-ring core, consisting of two fused 6-rings, is maintained throughout both atom-based and fragment-based evolution (see Figures 6.6 and 6.7), even though the original 1,2,3,4-tetrahydroisoquinoline ring system is changed during atom-based and fragment-based evolution into a 1,2-dihydronaphthalene ring system and a naphthalene ring system, respectively. Furthermore, the initial aromatic ring dipping into pocket 1 is only slightly changed into other functional groups of the same hydrophobic character along the different evolution pathways. The evolution has led to probing and optimization of hydrophobic interactions with different smaller subpockets of the HIV-RT binding site. Repositioning of the pocket 1-occupying substituent from the 5' to the 6' position accommodates interactions of hydrophobic substituents on other positions at the two-ring core with subpockets 2 and 3. In the atom-based evolution, the initial hydroxyl group sticking out in the direction of subpocket 2 (Figure 6.5a) is replaced by an S-phenyl group which can interact with subpocket 2 formed by L234, P236, and Y318 (Figure 6.5b). During fragment-based evolution, an ethenesulfinate group was attached *meta* to the hydrophobic group interacting with subpocket 1. This ethylenesulfinate group has just the appropriate size to dip into the small subpocket 3 (Hopkins, 2004).



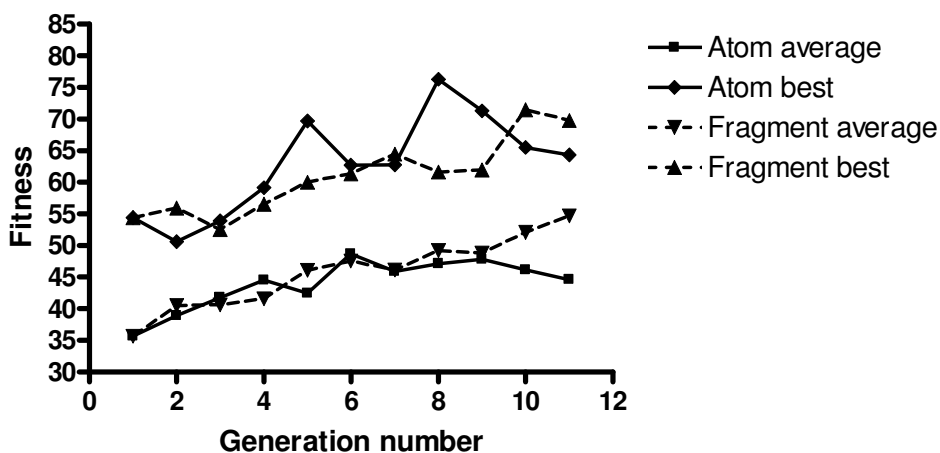
**Figure 6.6.** The optimization trajectory leading to the best molecule of the atom-based evolution. The numbers under the structures indicate their fitness score.



**Figure 6.7:** The optimization trajectory leading to the best molecule produced by the fragment-based evolution. The numbers under the structures indicate their fitness score. Note that the  $\text{SO}_3\text{C}_2\text{H}_3$ -group of the second parent molecule was modified to a  $\text{SO}_2\text{C}_2\text{H}_3$ -group by an error in our program. This bug was later fixed. In later fitness comparisons, not much influence on fitness was found by the presence or absence of the extra oxygen atom, so this glitch will probably not have influenced evolution much.

### The change in fitness over the generations

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

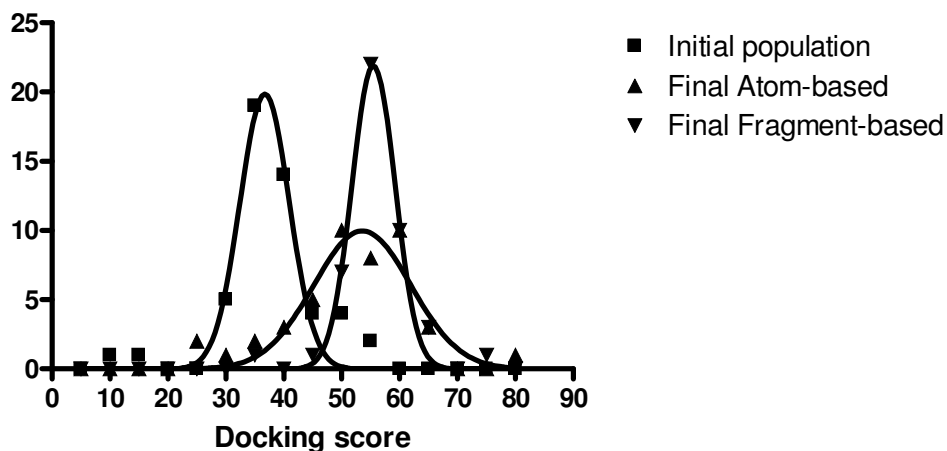


**Figure 6.8:** The average and the best fitness of molecules in each generation for the atom-based and fragment-based evolution.

Figure 6.8 shows that both the average fitness and the maximum fitness of the molecules in the population grow as the evolution proceeds. This means that new and better molecules are found, which implies that evolution improves upon pure selection (since pure selection would cause the average and maximum fitnesses of the later generations to approach the maximum of the first generation). However, virtual screening of a large library will also increase maximum fitness, as there is always a

probability that a new molecule will improve upon the known compounds. We should therefore analyze our data further to be able to say whether evolution is truly more effective or efficient than random search.

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



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

The best overall result of evolution (score=76) lies about 9 standard deviations from this average. Therefore the probability that a random search would produce a molecule with this or a higher fitness would be smaller than 10<sup>-9</sup>%. Scanning 500 molecules and getting the improvement shown in Figure 6.9 suggests that this improvement cannot be the result of a random search only. Therefore evolution seems truly more efficient than random search. We should note hereby that the distribution of the initial population, as shown in Figure 9, is Gaussian-like, but not exactly Gaussian. For example, there are about 5 molecules with a fitness score of about 50, which is more than would be expected from a true Gaussian distribution. Therefore, putting the odds that random

search produces a similar improvement as the evolutionary algorithm at  $1:10^{11}$  may be mathematically questionable. However, the distance between the best fitness found by the EAs and the average and best fitnesses of randomly produced molecules seems large enough to indicate that both of our evolutionary algorithms noticeably improve on random search.

### **Fitness peaks/plateaus**

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

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

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



fragment-based evolution does not show a similar deterioration. It is possible that the fragment-evolution has not reached a similar plateau yet, since it reached its peak later than the atom-based evolution.

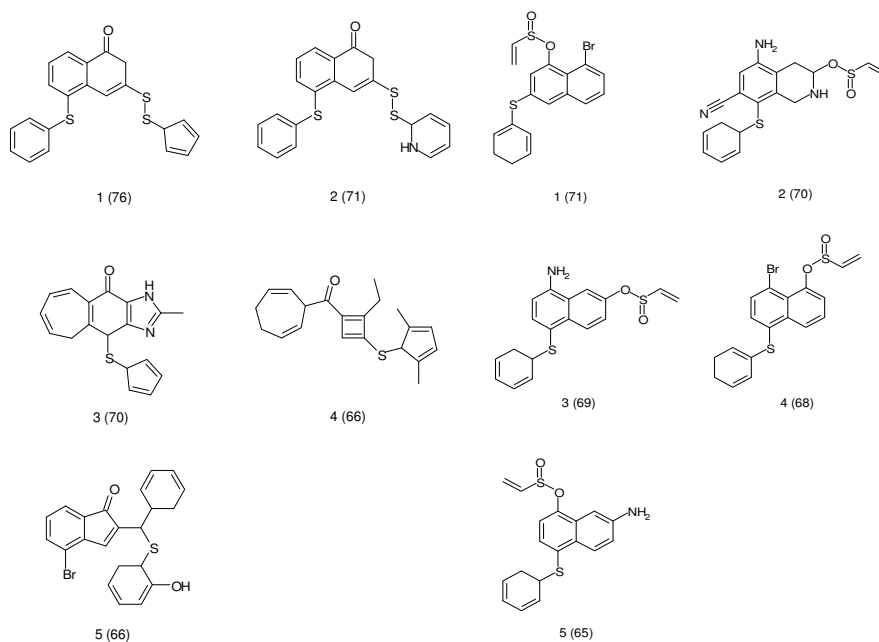
### **Comparing the molecules created by atom-based and fragment-based evolution**

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

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

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

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



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

Another reason for not making the selection pressure too high is the presence of local optima. It is quite common for a high-scoring molecule (like the bottom-left molecule in the atom-based evolution, score 53) to produce a lower-scoring molecule (score 46 in this case) which however gives birth to new molecules that have a fitness comparable to or even higher than that of its (great)grandparent (52 and 57, respectively). One may argue that this could be caused by imprecision in the docking procedure, which may assign somewhat different scores to one molecule if the docking experiment were to be repeated. In such a case the intermediate molecule may indeed have an intermediate score if more calculations would be performed. Alternatively, there could really be local maxima in an optimization trajectory, which would trap a greedy (“only better offspring”) algorithm too easily. Methods like evolutionary algorithms (used here) or simulated annealing, which generally explore around local optima, may therefore be inherently more suitable for molecular space than a greedy search which discards all variants with a lower score than the highest scorer. Another reason not to select too greedily is that both computational fitness functions and

experimental affinity measures are bound to have some noise, with some superior molecules perhaps scoring slightly worse than their competition due to the random nature of the docking process and the random variations in experimental measurements. An evolutionary algorithm, to be effective, should therefore tolerate this noise and not select too strictly and too greedily. We also stress that the accuracy of the docking score is not too relevant for the comparisons made here. Our aim was not to find novel HIV-RT inhibitors, but to evaluate the two EAs in a realistic setting. Docking procedures possibly assigning different scores over different runs to the same molecule can even be considered to incorporate an extra element of realism since biological experiments also tend to yield ranges of values, and a good optimization algorithm should work despite these irregularities.

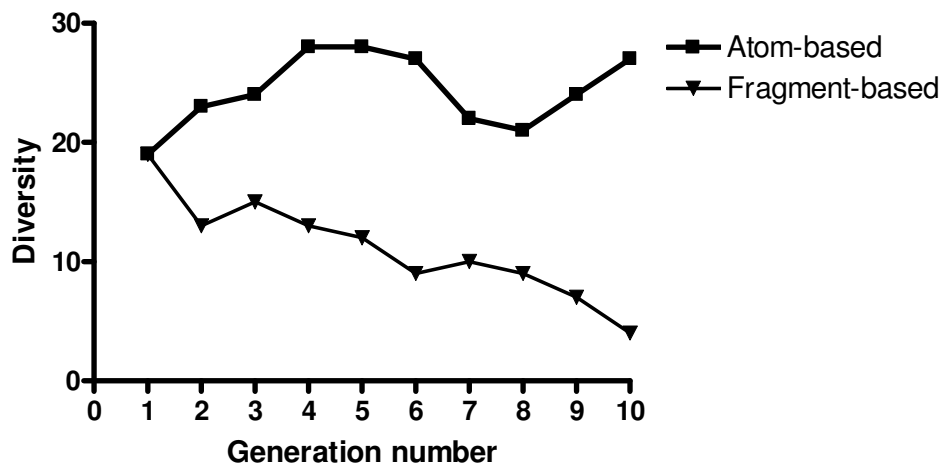
The third similarity between the different evolution methods is the relatively high occurrence of crossover. While our settings should have led to about 15% crossover versus 85% mutation for modifying molecules, the optimization tree of highest scoring compound in the atom-based evolution had 4 crossovers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

Surveying our results, we learned that atom-based and fragment-based evolution have lots in common and that both seem approximately equally efficient in this experiment. However, there are also differences. First of all, the evolution trees show more clearly than the fitness plot that atom-based evolution does run more smoothly. It reaches a better molecule than the fragment-based evolution, and it does so in fewer generations. It truly seems to be more efficient, though not by a large amount. In the fragment-based evolution shown in Figure 6.7, big substituents make large jumps around the molecular core. An example would be the  $\text{SO}_2\text{C}_2\text{H}_3$  group in the middle column of Figure 6.7, which moves to very different positions of the ring system. If a large substituent is attached at very different positions in parent and offspring molecules, this would imply

that either the substituent of the offspring molecule is docked in another part of the binding pocket than it was in the parent molecule or that the offspring's core is docked in another place. Since the dockings of parents and offspring in fragment-based evolution are therefore more likely to be dissimilar, the evolution is probably less efficient. If, for example, in fragment-based evolution only half of the offspring of a good molecule is docked in the way of their parent, it is likely that only this half will have a good or even better score than the parent molecule, and the time spent to evaluate the other half of the offspring is wasted. The smaller changes in atom-based evolution should result in offspring of which a large part will be docked similarly to the parent, say 90%, leading to fewer offspring with low fitness scores due to radically different docking. Quantitatively, one would have to analyze the correlation between the fitness scores of parent and offspring. It would definitely be interesting in future investigations to automate such a "docking comparison" process and get the actual numbers.

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

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



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

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

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

### **Synthetic feasibility**

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

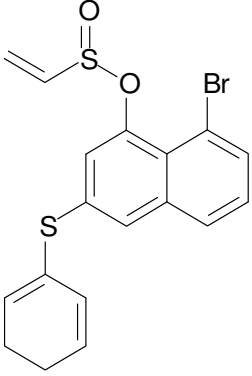
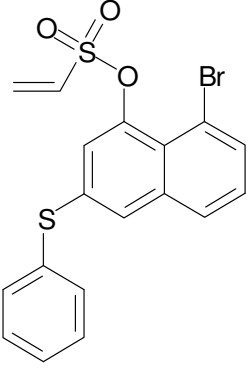
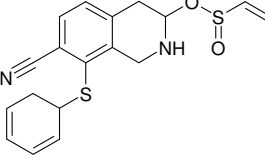
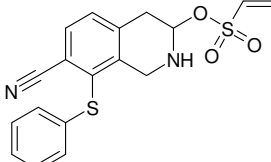
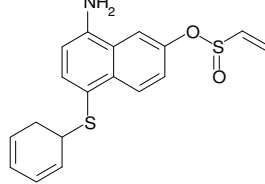
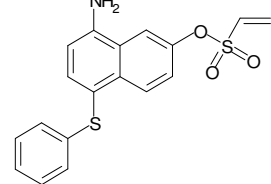
To assess the effects of ease-of-synthesis requirements on the molecules reached by atom-based and fragment-based evolution, we submitted a selection of the top-10 atom-based molecules and top-10 fragment-based molecules to a chemist, who suggested improvements to enhance ease of synthesis. The resulting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

From comparing the structures and scores of the molecules in Table 6.2 we conclude that fragment-based evolution indeed produces superior results respecting ease of synthesis. This is mainly because fragment-based molecules need fewer modifications, which leads to a smaller change in structure, which leads to a smaller change in activity, and since changing an optimized structure is likely to decrease activity, the smaller adaptation needed for the fragment-based molecules is beneficial. From our results, it seems that the position of the modifications also plays a role; if modifications are in or near the core of the molecule, the change in activity is much more dramatic than when a substituent at the “edge” is changed. For example, only

modifying the central ring of the 4<sup>th</sup> best atom-based compound (Figure 6.10) from cyclobutadiene to phenyl made the docking score fall from 66 to 9. In contrast, the decrease in fitness in the fragment-based molecules is generally much smaller, about 10 units, probably because only the substituents are changed.

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

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

Fragment-based evolution			71	63
			70	50
			69	68

### Comparison with literature

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

**Atom-based evolution.** To our knowledge, the earliest publication that described computational evolution of molecules was by Glen and Payne (Payne, 1995). Their mutations and crossover were more or less the same as our atom-based operators, though they included a 2-point crossover (which exchanged the central parts of two molecules) and limited insertions to methylene (only carbon-atoms could be inserted). Douguet et al. (Douguet, 2000) also included the 2-point crossover, but did not use a ring breaking operator, and seemed to focus much on changing larger parts of the



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

**Fragment-based evolution.** The fragment-based evolutionary approaches we found in literature are those of Schneider et al. (Schneider, 2000), Pegg et al. (Pegg, 2001), Vinkers et al. (Vinkers, 2003) and Dey et al. (Dey, 2008). The evolutionary algorithm of Nicolaou et al. (Nicolaou, 2009) used both atom-based and fragment-based evolution, their atom-based operators were discussed in the preceding paragraph, while their fragment-based operators will be discussed here. Schneider obtained his fragments by using retrosynthetic rules on the WDI database (Daylight, 2005), resulting in over 24,000 fragments. From these fragments, molecules were constructed by applying the same retrosynthetic rules in the reverse direction. The algorithm did

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

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

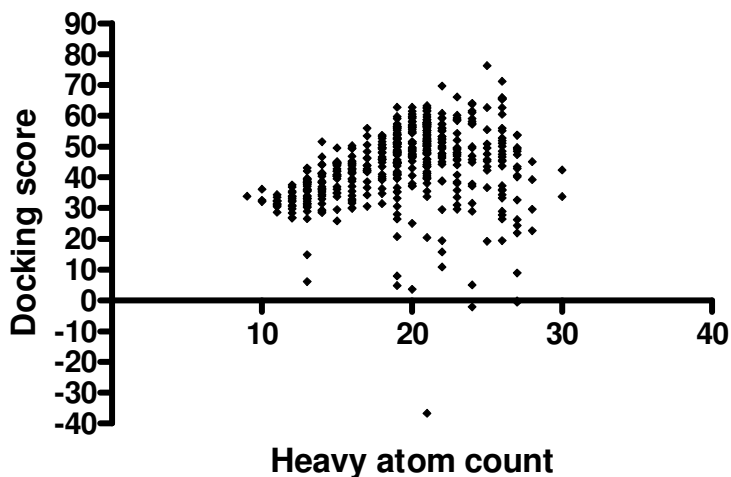
Next to comparing the mutation and crossover operators, it may be useful to compare the parameters of evolution of our investigation with those in literature. In the above-mentioned investigations, population sizes ranged from 30 to 1000, and the number of generations allowed ranged from 50 to 1000.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

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

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

One important point to consider is whether our test system, docking and its score, are suitable as fitness function. For example, Verdonk et al. (Verdonk, 2004) have

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



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

## Future perspectives

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

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

In addition, there are some factors which may hamper the optimization. One of those is the conversion of the molecules from 2D to 3D. The encoding of molecules in the current version of the Molecule Evaluator does not specify stereochemistry. This means that the conversion of the molecule "genome" to 2D will give rather randomly a *cis* or *trans* stereoisomer for each double bond outside rings, and the subsequent 2D to 3D conversion with MOE will add undefined R/S stereocenters. The Molecule Evaluator and (as far as we could observe) MOE are deterministic in their conversion procedures, which would ensure that a molecule genome would always be converted into one specific stereoisomer, so this will not affect reproducibility. However, offspring, even if modified outside the stereocenters, might have different stereochemistry assigned to them. Such "non-genetic" jumps in fitness would add noise and hamper the optimization. It would therefore be useful to encode stereochemistry in future versions of the Molecule Evaluator. Another concern is that some structures might be in a different tautomeric state in a biological environment (an enol might be a

keto group in solution). Since bonds are coded into the genes, the tautomer remains fixed over generations and does not give rise to offspring in a different configuration, in contrast to enantiomerism. Therefore tautomerism will disturb evolution less than enantiomerism. However, a chemist should inspect the molecules with the highest scores for possible tautomerism, and dock the tautomers to explore the effects of the equilibria on the fitness score.

## Summary and conclusion

In this investigation we compared the relative usefulness of atom based (small step) evolution and fragment based (large step) evolution for optimizing potential drug molecules. As was expected, atom-based evolution found a better molecule after fewer steps (although the difference was not large), while the fragment-based evolution produced molecules that seemed easier to synthesize. By comparing the two approaches, also a number of other lessons were learned concerning molecule evolution and how to improve the Molecule Evuator.

Firstly, there is no need to add entirely novel molecules after the first generation. The fitness of the population increases so steeply, that the new molecules do not contribute any useful “genes”.

Secondly, and somewhat surprisingly, crossover can be a valuable operator in molecule optimization; in contrast to “normal” optimization strategies in pharmaceutical research, in which variations are made on one molecule, it may be worthwhile to combine one molecular core and some of the side groups of other high-scoring molecules.

Future modifications to improve the Molecule Evuator would be to stop adding *de novo* molecules each generation, experiment whether allowing a molecule to appear more than once in a generation would find an even better optimum, and implement automatic gathering of statistical data of all evolved molecules to analyze the contributions of the various possible mutations or crossovers to molecule fitness. Finally, new mutation operators such as “add/delete fragment” will have to be added to the fragment-based evolution to prevent excessive loss of diversity and premature convergence.

Optimizing molecule optimization techniques is still complicated and requires lots of experiments and sufficiently sophisticated fitness functions. However, if drug designers want to explore a greater part of the huge number of potential drug-like

molecules (over  $10^{60}$ ) or want to use accurate and therefore slow fitness functions for a more reliable *in silico* screening, efficiency in optimization is needed. We hope that studies like this one will not only lead to faster optimization methods, but will also give insight into the general craft of molecule optimization.

### Acknowledgement

We thank Johannes Brussee for his help in assessing the synthetic feasibility of the best-scoring compounds, and for suggesting modifications for greater ease of synthesis.

### References

- (Barreiro, 2007) Barreiro, G.; Guimarães, C. R. W.; Tubert-Brohman, I.; Lyons, T. M.; Tirado-Rives, J.; Jorgensen, W. L. Search for Non-Nucleoside Inhibitors of HIV-1 Reverse Transcriptase Using Chemical Similarity, Molecular Docking, and MM-GB/SA Scoring. *J. Chem. Inf. Model.* **2007**, *47*, 2416-2428.
- (Beilstein, 2009) <http://www.info.crossfirebeilstein.com/features.shtml>.
- (Bohacek, 1996) Bohacek R. S.; McMartin C.; Guida, W.C. The Art and Practice of Structure-Based Drug Design: A Molecular Modeling Perspective. *Med. Res. Rev.* **1996**, *16*, 3-50.
- (Brady, 2000) Brady, G. P.; Stouten, P. F. W. Fast prediction and visualization of protein binding pockets with PASS. *J. Comput.-Aided Mol. Des.* **2000**, *14*, 383-401.
- (Brown, 2004) Brown, N.; McKay, B.; Gilardoni, F.; Gasteiger, J. A Graph-Based Genetic Algorithm and Its Application to the Multiobjective Evolution of Median Molecules. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 1079-1087.
- (Carter, 2005) Carter, E.; Ebdon, S.; Neal-Sturgess, C. Optimization of Passenger Car Design for the Mitigation of Pedestrian Head Injury Using a Genetic Algorithm. In *Proceedings of GECCO 2005*; Beyer, H.G. et al., Ed.; ACM: New York, 2005; Vol 2, pp 2113-2120.
- (CAS, 2009) <http://www.cas.org/expertise/cascontent/ataglance/index.html>.
- (Dalby, 1992) Dalby, A.; Nourse, J.G.; Hounshell, W.D.; Gushurst, A.K.I.; Grier, D.L.; Leland, B.A.; Laufer, J. Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 244-255.

- (Daylight, 2006) World Drug Index, published by Daylight Chemical Information Systems Inc. <http://www.daylight.com/products/wdi.html>.
- (Dey, 2008) Day, F.; Cafilisch, A. Fragment-Based de Novo Ligand Design by Multiobjective Evolutionary Optimization. *J. Chem. Inf. Model.* **2008**, *48*, 679-690.
- (Douguet, 2000) Douguet, D.; Thoreau, E.; Grassy, G. A genetic algorithm for the automated generation of small organic molecules: Drug design using an evolutionary algorithm. *J. Comput.-Aided Mol. Des.* **2000**, *14*, 449-466.
- (Ertl, 2000) Ertl, P.; Rohde, B.; Selzer, P. Fast Calculation of Molecular Polar Surface Area as a Sum of Fragment-Based Contributions and Its Application to the Prediction of Drug Transport Properties. *J. Med. Chem.* **2000**, *43*, 3714-3717.
- (Esnouf, 1997) Esnouf, R. M.; Ren, J.; Hopkins, A. L.; Ross, C. K.; Jones, E. Y.; Stammers, D. K.; Stuart, D. I. Unique features in the structure of the complex between HIV-1 reverse transcriptase and the bis(heteroaryl)piperazine (BHAP) U-90152 explain resistance mutations for this nonnucleoside inhibitor. *Proc. Natl. Acad. Sci. USA* **1997**, *94*, 3984-3989.
- (Gillet, 1999) Gillet, V. J.; Willett, P.; Bradshaw, J.; and Green, D.V.S. Selecting Combinatorial Libraries to Optimize Diversity and Physical Properties. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 169-177.
- (Gillet, 2000) Gillet, V. J. *De Novo* Molecular Design. In *Evolutionary Algorithms in Molecular Design*, Clark D.E., Ed.; John Wiley & Sons, Inc.: New York, 2000; Chapter 4, pp 49-69.
- (Globus, 1999) Globus, A.; Lawton, J.; Wipke, T. Automatic molecular design using evolutionary techniques. *Nanotech.* **1999**, *10*, 290-299.
- (Hann, 2000) Leach, A. R.; Hann, M. M. The *in silico* world of virtual libraries. *Drug Discovery Today* **2000**, *5*, 326-336.
- (Hopfinger, 1996) Hopfinger, A. J.; Holzgrabe, U. Conformational Analysis, Molecular Shape Comparison, and Pharmacophore Identification of Different Allosteric Modulators of Muscarinic Receptors. *J. Chem. Inf. Comput. Sci.* **1996**, *36*, 1018-1024.
- (Hopkins, 2004) Hopkins, A. L.; Ren, J.; Milton, J.; Hazen, R. J.; Chan, J. H.; Stuart D. I.; Stammers, D. K. Design of Non-Nucleoside Inhibitors of HIV-1 Reverse Transcriptase with Improved Drug Resistance Properties. 1. *J. Med. Chem.* **2004**, *47*, 5912-5922.



- (Jones, 1997) Jones, G.; Willett, P.; Glen, R. C.; Leach, A. R.; Taylor, R. Development and Validation of a Genetic Algorithm for Flexible Docking. *J. Mol. Biol.* **1997**, *267*, 727-748.
- (Jorgensen, 2006) Jorgensen, W. L.; Ruiz-Caro, J.; Tirado-Rives, J.; Basavapathruni, A.; Anderson, K. S.; Hamilton, A. D. Computer-aided design of non-nucleoside inhibitors of HIV-1 reverse transcriptase. *Bioorganic & Medicinal Chemistry Letters* **2006**, *16*, 663-667.
- (Kimura, 1998) Kimura, T.; Hasegawa, K.; Funatsu, K. GA Strategy for Variable Selection in QSAR Studies: GA-Based Region Selection for CoMFA Modeling. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 276-282.
- (Klebe, 2006) Klebe, G. Virtual ligand screening: strategies, perspectives and limitations. *Drug Discovery Today* **2006**, *11*, 580-594.
- (Koza, 2005) Koza, J. R.; Al-Sakran, S. H.; Jones, L. W. Automated Re-Invention of Six Patented Optical Lens Systems using Genetic Programming. In *Proceedings of GECCO 2005*; Beyer, H.G. et al., Ed.; ACM: New York, 2005; Vol 2, pp 1953-1960.
- (Lazar, 2004) Lazar, C.; Kluczyk, A.; Kiyota, T.; Konishi, Y. Drug Evolution Concept in Drug Design: 1. Hybridization Method. *J. Med. Chem.* **2004**, *47*, 6973-6982.
- (Lewell, 1998) Lewell, X. Q.; Judd, D. B.; Watson, S. P. and Hann, M. M. RECAP- Retrosynthetic Combinatorial Analysis Procedure: A Powerful New Technique for Identifying Privileged Molecular Fragments with Useful Applications in Combinatorial Chemistry. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 511-522.
- (Morris, 1998) Morris, G. M.; Goodsell, D. S.; Halliday, R. S.; Huey, R.; Hart, W. E.; Belew, R. K.; Olson, A. J. Automated Docking Using a Lamarckian Genetic Algorithm and an Empirical Binding Free Energy Function. *J. Comput. Chem.* **1998**, *19*, 1639-1662.
- (Nachbar, 1998) Nachbar, R. B. Molecular Evolution: A Hierarchical Representation for Chemical Topology and its Automated Manipulation. In *Proceedings of the Third Annual Genetic Programming Conference*; Madison, Wisconsin, 1998, pp 246-253.
- (Nachbar, 2000) Nachbar R. B. Molecular Evolution: Automated manipulation of hierarchical chemical topology and its application to average molecular structures. *Genetic Programming and Evolvable Machines* **2000**, *1*, 57-94.

- (NCI, 2000) NCI open database, as found on <http://cactus.nci.nih.gov/ncidb2/download.html>, the August 2000 2D file.
- (Nicolaou, 2009) Nicolaou, C. A.; Apostolakis, J.; Pattichis, C. S. De Novo Drug Design Using Multiobjective Evolutionary Graphs. *J. Chem. Inf. Model.* **2009**, *49*, 295-307.
- (Payne, 1995) Glen, R. C.; Payne, A. W. R. A genetic algorithm for the automated generation of molecules within constraints. *J. Comput.-Aided Mol. Des.* **1995**, *9*, 181-202.
- (Pegg, 2001) Pegg, Scott C.-H.; Haresco, J. J.; Kuntz, I. D. A genetic algorithm for structure-based de novo design. *J. Comput.-Aided Mol. Des.* **2001**, *15*, 911-933.
- (Pubchem, 2009) <http://en.wikipedia.org/wiki/PubChem>
- (Rees, 2003) Rees, P. Big pharma learns how to love IT. *Scientific Computing World* **2003**, 16-18.
- (Ren, 1995) Ren, J.; Esnouf, R.; Garman, E.; Somers, D.; Ross, C.; Kirby, I.; Keeling, J.; Darby, G.; Jones, Y.; Stuart, D.; Stammers, D. High resolution structures of HIV-1 RT from four RT-inhibitor complexes. *Struct. Biol.* **1995**, *2*, 293-302.
- (Schneider, 2000) Schneider, G.; Lee, M.-L.; Stahl, M.; Schneider, P. De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks. *J. Comput.-Aided Mol. Des.* **2000**, *14*, 487-494.
- (Shen, 2003) Shen, L.; Shen, J.; Luo, X.; Cheng, F.; Xu, Y.; Chen, K.; Arnold, E.; Ding, J.; Jiang, H. Steered Molecular Dynamics Simulation on the Binding of NNRTI to HIV-1 RT. *Biophys. J.* **2003**, *84*, 3547-3563.
- (Sheridan, 2000) Sheridan, R. P.; SanFeliciano, S. G.; Kearsley, S. K. Designing targeted libraries with genetic algorithms. *J. Mol. Graph. Model.* **2000**, *18*, 320-334.
- (Tan, 2005) Tan, T. Z.; Quek, C.; Ng, G. S. Brain-inspired Genetic Complementary Learning for Stock Market Prediction. In *Proceedings of IEEE CEC 2005*; IEEE Press: Piscataway, 2005; Vol 3, pp 2653-2660.
- (Titmuss, 1999) Titmuss, S. J.; Keller, P. A.; Griffith, R. Docking Experiments in the Flexible Non-nucleoside Inhibitor Binding Pocket of HIV-1 Reverse Transcriptase. *Bioorganic & Medicinal Chemistry* **1999**, *7*, 1163-1170.
- (Verdonk, 2004) Verdonk, M. L.; Berdini, V.; Hartshorn, M. J.; Mooij, W. T. M.; Murray, C. W.; Taylor, R. D.; Watson, P. Virtual Screening Using Protein-Ligand Docking: Avoiding Artificial Enrichment. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 793-806.

- (Vinkers, 2003) Vinkers, H. M.; De Jonge, M. R.; Daeyaert, F. F. D.; Heeres, J.; Koymans, L. M. H.; Van Lenthe, J. H.; Lewi, P. J.; Timmerman, H.; Van Aken, K.; Janssen, P. A. J. SYNOPSIS: SYNthesize and OPTimize System in Silico. *J. Med. Chem.* **2003**, *46*, 2765-2773.
- (Vinkers, 2004) Vinkers, H. M. *Computational drug design*. PhD thesis Free University, Amsterdam, the Netherlands, 2004.
- (Westhead, 1996) Clark, D. E.; Westhead, D. R. Evolutionary algorithms in computer-aided molecular design. *J. Comput.-Aided Mol. Des.* **1996**, *10*, 337-358.