TECHNISCHE UNIVERSITÄT WIEN
VIENNA UNIVERSITY OF TECHNOLOGY

Master Thesis

Visualization of n:m Drug Target Interactions

Submitted at the
Institute of Software Technology and Interactive Systems [E188]
of the Vienna University of Technology
Supervisor
Dr. Monika Lanzenberger

by

Gökhan Ibis
0225707
Brigittenauer Lände 160-162/1/64
1200 Wien

Vienna, October 13, 2007

Contents

Kurzfassung

Durch die Fortschritte in der Molekularbiologie und Informatik sind in den letzten Jahren computerunterstützte Verfahren auch bei der Erforschung neuer Arzneimittel immer beliebter geworden. Virtual Screening ist ein solches Verfahren, bei dem 3D Pharmacophorefilter angewendet werden, um eine Vielzahl von chemischen Mitteln virtuell daraufhin zu überprüfen, ob sie eine gewisse biologische Aktivität im menschlichen Körper hervorrufen. Die Gefahr dabei ist allerdings, dass unerwünschte Nebenwirkungen zu wenig in Betracht gezogen werden. Moderne Ansätze lösen das Problem, indem sie chemische Substanzen auf mehrere biologische Aktivitäten überprüfen. Dieses Verfahren, das "Activity Profiling" genannt wird, erschwert allerdings die Analyse der Testergebnisse, da die Ergebnismenge drastisch ansteigt.

Diese Masterarbeit beschäftigt sich mit der Visualisierung von solchen Ergebnissen. Zunächst werden die Quelldaten und Ergebnisdaten des Virtual Screening Prozesses im Rahmen aktueller Screeningverfahren beschrieben. Dabei wird vor allem der Vergleich zu Verfahren, bei denen chemische Mittel reell getestet werden, betrachtet und die Möglichkeit, virtuelles Screening mit dem reellen Ansatz zu kombinieren, erläutert. Visualisierung stellt einen weiteren Schwerpunkt dieser Arbeit dar. Diverse Visualisierungstechniken und deren Anwendbarkeit werden speziell bei dieser Problemstellung genauer unter die Lupe genommen. Außerdem werden Anwendungen, die sich bereits mit dieser Problemstellung auseinandergesetzt haben, analysiert, um Vor- und Nachteile der jeweiligen Anwendung zu ermitteln. Usability wird dabei durch Benutzerfeedback beurteilt.

Aufbauend auf die gewonnenen Erkenntnisse wird im Rahmen dieser Masterarbeit eine Softwareanwendung entwickelt, die einerseits Aktivitätsprofile visuell darstellt und andererseits eine effektive Benutzerinteraktion ermöglicht. Die graphische Darstellung ist mittels OpenGL implementiert, um die Hardwarebeschleuningung durch die Grafikkarte auszunützen. Da der Screening Prozess sehr lange dauern kann, ist es erforderlich, einen Weg zu finden, um diesen Prozess zu beschleunigen. Aus diesem Grund wird eine Schnittstelle zum Screening Prozess zur Verfügung gestellt, die es möglich macht, den Screening-Client beispielsweise gegen eine effizienter Lösung oder eine verteilte Anwendung auszutauschen. Eine weitere Eigenschaft der Applikation im Zusammenhang mit dem Screening Prozess ist, dass der Benutzer/die Benutzerin stets über den Status des Prozesses und über neue Ergebnisse informiert wird. Die Ergebnisse werden unter anderem in einer Heatmap, die zusätzlich mit Höheninformationen ausgestattet ist, dargetstellt. Die Heatmap erlaubt dem Benutzer/der Benutzerin Ergebnisse, die nicht von Interesse sind, zu filtern. Um eine bessere Einsicht in diese komplexen Daten zu ermöglich, bietet die Anwendung mehrere Sichten (Views), die durch moderne Linking and Brushing Techniken miteinander gekoppelt sind.

Abstract

After recent advances in molecular biology and informatics computer-based techniques got more and more popular in pharmaceutical research. Virtual screening using 3D pharmacophore filters has become a modern approach in computational drug discovery that is used to filter chemical libraries according to their simulated affinity to a specific biological target. Typically, several molecules are tested against a single model in order to predict, whether they show a specific biological activity. In the case of screening against a single biological target, the main risk in drug development, off-target effects (or side effects) are often neglected. Therefore, recent approaches screen molecules against several targets ("activity profiling"). However, when screening of several thousand compounds is performed against several targets, the resulting amount of data is huge and difficult to analyze.

The current work focuses on the visualization of the filtering results, i.e. the affinity of several molecules against several targets. The data source used for virtual screening as well as the results are first described and analyzed in the context of existing screening methods. Additionally the relationship between virtual screening and large-scale biological testing, where large libraries of chemicals are physically screened against a biological target is discussed. It is also described how these techniques may be combined in visualizing results. Different established visualization methods are described and compared. Already existing visualization techniques are analyzed and discussed in terms of their applicability to this specific problem, especially focusing on the advantages and disadvantages of each approach. Usability is evaluated based on the feedback of members of the Institute of Pharmacy of the Leopold-Franzens-University of Innsbruck.

Based on these insights, a novel visualization program is developed that on the one hand displays activity profiles, and on the other hand allows for user-effective interactive exploration of the displayed results. The visualization is implemented in OpenGL, taking advantage of hardware-accelerated real-time rendering, and providing an interface to the pharmacophore screening process. This very time consuming process makes it necessary to distribute the screening and permanently display the status of available and new incoming data. The resulting activity profiles are displayed in a heat map including height information allowing the user to set filters for the target affinities in a comfortable way, which are separated into different adaptable categories, like therapeutical or pharmacological class, medical indication, and target type. Furthermore, the program provides other views including a selectable list of target-molecule matches, which gives an overview of the selected hits and a detailed view showing the mapping details of a specific target with respect to a molecule. These views focus on user interaction, which is integrated using linking and brushing techniques and allows the user to explore the connection between targets and molecules.

Chapter 1

Introduction

1.1 Problem Domain

Improvements in informatics have induced many research fields to use computer-based methods for their purposes and so does modern drug research. Computer programs are used to virtually screen molecules against biological targets to find the ones which cause a specific biological activity in the human body to cure diseases.

A common problem of drug development are side effects and toxicity, which are caused by molecules triggering biological activities not only on desired targets, but also on others. To find and remove chemicals with side effects chemists apply pharmacological profiling, where a drug candidate is screened against multiple targets.

Nowadays, the idea of combining the current screening protocol with pharmacological profiling to screen multiple molecules against multiple targets becomes popular. This is called "Activity Profiling". However, analyzing the result set of screening thousands of molecules against several targets is a becoming challenging due to the large amount of data to be processed.

The Vienna University of Technology and Inte:ligand, a company which develops computer-aided software solutions for drug research, have together developed two applications dealing with the problem of visualizing and filtering of activity profiling results. Both tools are using multiple view systems combined with linking and brushing for an interactive visualization of activity profiles. The analysis of those programs has revealed that further work can significantly improve the visualization approaches for activity profiling.

1.2 Suggested Approach

The development of a novel application which takes advantage of the strengths of previous approaches is also part of this work. Especially the visualization of activity profiles of Hitvis [8] by using an interactive map with heat- and height information is taken as a starting point. However, testing and evaluating Hitvis has shown that there are significant shortcomings of the map which are addressed in this work.

The fact that a huge amount of data has to be displayed and that the displayed items often have to be updated through filtering operations, requires hardware acceleration for rendering. Therefore the visualization is implemented in OpenGL which on the one hand takes advantage of hardware-acceleration, and on the other hand allows for a 3D implementation of the map. Furthermore, the new application is written in Java which provides platform independence.

1.3 Thesis Outline

After a short introduction into the problem domain and the suggested solution in chapter 1, the following chapter, chapter 2, describes virtual screening and introduces into modern drug discovery techniques. The problem of side effects and toxicity and why Activity Profiling is an approach to address this problem is the topic of chapter 3. Chapter 4 delineates the principles of visualization and shows state of the art visualization taxonomies. Furthermore, this chapter describes the differences between Scientific Visualization and Information Visualization. The latter is covered in more detail in chapter 5, where two visualization methods, which are used in the developed application, are depicted. Chapter 6 describes Multiple View Systems and how they can be used in an effective manner. Linking and Brushing, which is often used in multiple view systems, is mentioned in chapter 6 and covered in more detail in chapter 7, where different brushing techniques are discussed. chapter 8 provides an overview of state of the art tools for the visualization of n:m drug target interactions. Together with the members of Institute of Pharmacy of the Leopold-Franzens-University of Innsbruck these tools have been discussed and evaluated. The results of this evaluation are also described in chapter 8. Chapter 9 is about the application developed as part of this thesis and describes important aspects of the program. Chapter 10 provides a summary of this work and indicates remaining issues.

Chapter 2

Virtual Screening

Since molecular biology and informatics have made big advances over the last years, pharmaceutical researchers make more and more use of computational techniques. *Virtual screening* [57, 21] is such a technique and was introduced in the 1970s. The main goal of virtual screening is to virtually screen large chemical libraries for compounds that fit to targets of known structure. Virtually means to screen by using computer programs, and the chemists experimentally test only those that are predicted to bind well. This new screening method was a hot prospect, but insufficient software support and the necessity to determine receptor structures to atomic resolution and catalog them, avoided virtual screening getting popular at that time. Chemists preferred traditional screening techniques, where large libraries of chemicals are physically screened against a biological target. However, virtual screening is popular nowadays and is predicted to be a very important part of drug research in the future.

2.1 Virtual vs. Traditional Screening

In the past chemists could only synthesize a few hundred or probably fewer compounds in a year. Traditional screening methods have increased this number to 10^3-10^4 compounds a year labor work [57]. To synthesize millions or billions of compounds computational chemists take advantage of computer programs to automatically evaluate very large libraries of compounds (virtual screening). However, virtual screening is not replacing traditional screening. Virtual screening is used to reduce the number of possible molecules that fit to a certain target, but the remaining molecules have to be screened physically. A typical pharmaceutical project timeline, is shown in Figure 2.1.

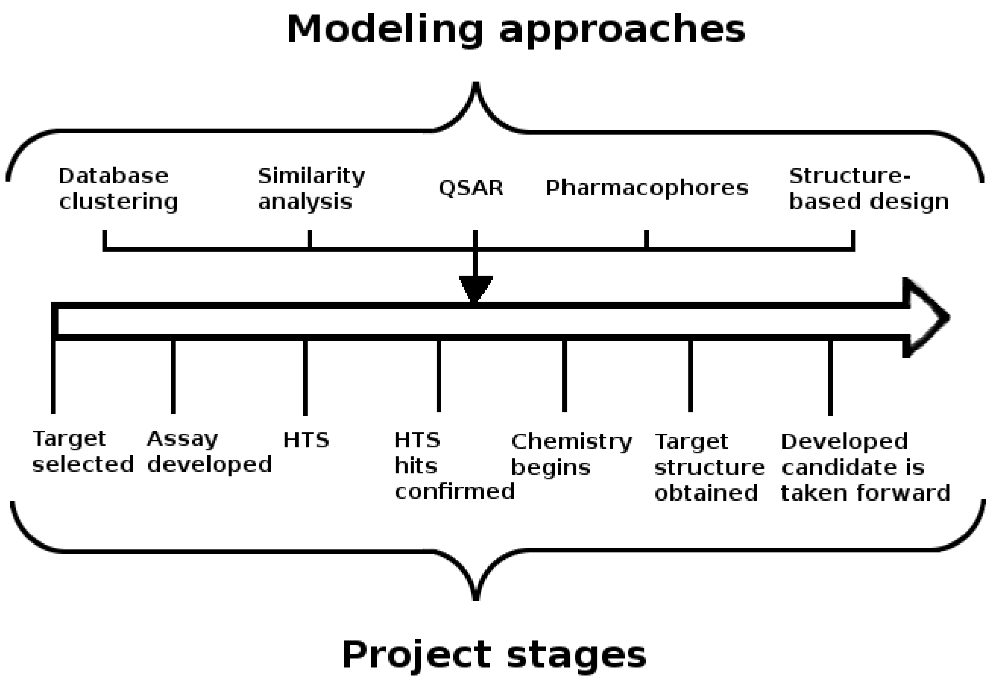


Figure 2.1: A typical pharmaceutical project timeline. The stages of the project are shown below the arrow. Contributions which may be made by the computational chemistry group are shown above [57].

The stages of the project are shown below the arrow. First a target is selected and an assay is developed. The next stage is high-throughput screening (HTS), where a huge amount of molecules are experimentally screened against a target. Afterwards, hits found by this process are confirmed. In the last two stages the target structure is obtained and the developed candidate is taken forward for further tests.

Contributions wich may be made by the computational chemistry group are shown above the arrow. Common contributions are:

- **database clustering**, where molecules are grouped based on similarity.
- **similarity analysis**, where the structure of molecules is used to calculate a wide variety of descriptors.
- **quantitative structure-activity relationship (QSAR)**, which is a process that quantitively correlates the structure of molecules with biological activity.

- **pharmacophores**, which are congeries of chemical features a molecule must have to trigger biological activity on a target.
- **structure-based design**, where pharmacophores are created based on structural information of a drug target and molecules interacting with the target.

2.2 Limiting the search space

The total number of molecules in the virtual chemistry space is about 10^{100} [57]. There are some assumptions to limit the search space. A basic assumption is to focus on libraries that are more practical. Molecules, which are too large or too lipophilic are not associated with drugs [57]. Chemical libraries containing such molecules can be excluded from the search space. Cheap libraries with molecules that are often used in drugs are preferred instead. There are some other reasons, which make many of the molecules in a virtual library impractical. For example certain combinations of functional groups are not synthetically compatible or some combinations of functional groups are quite rare to find in drugs. Such molecules can also be excluded from the search space. Figure 2.2 shows, in schematic form, some of the different reasons for eliminating compounds. Another basic assumption is to avoid to fully enumerate the virtual library. There are some computational techniques like similarity clustering, which is described in section 2.3, and figure 2.3, to avoid full enumeration.

2.3 Current Practices in Virtual Screening

In the following some state of the art techniques in virtual screening as well as the current situation of virtual screening is described using statistical information provided by Walters [57].

Commercial software:
Commercial software in pharmaceutical drug research is driven by a few software vendors [55, 25, 23, 44, 29] and is widely used by computational chemists nowadays. Altough the software is valuable in a wide range of drug design applications, it should be mentioned that there are still certain limitations when working with large libraries of molecules. Many of the software tools get very slow and unstable

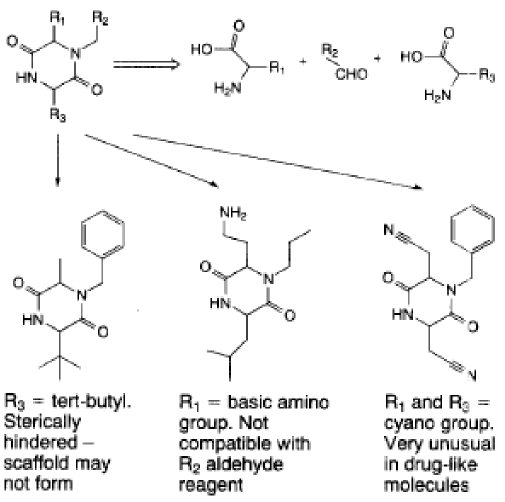

R₁ R₂
HO CHO H₂N R₃
R₃ = tert-butyl. Sterically hindered – scaffold may not form
R₁ = basic amino group. Not compatible with R₂ aldehyde reagent
R₁ and R₃ = cyano group. Very unusual in drug-like molecules

Figure 2.2: A molecule library and three specific molecules within the library that may be rejected for various reasons [57].

when loading large virtual libraries. Additionally to the limitations in software engineering, algorithms used by the commercial software are relatively primitive and improvable [57].

How fast is fast?
When comparing different virtual screening methods an important aspect is speed. If one molecule per minute is processed, screening parallel with 32 processors would mean that $\tilde{1}$.4 million compounds can be screened in a month [57]. This is nowhere near the size of typical libraries. Therefore virtual screening has generally been used to study a single molecule or lead class[1].

2D similarity
The use of 2D-similarity methods can greatly speed up processing of a virtual library compared to traditional screening methods. The 2D representation of a molecule allows to calculate a wide variety of descriptors [53]. Most of them can be calculated very fast, which makes it possible to process hundreds of thousands of structures in an hour. 2D similarity methods are so rapid that they are often used to select compounds from a virtual library that are similar to an existing lead. This is shown in Figure 2.3.

[1]A set of molecules which have a common chemical property.

The lead molecule is in the middle. The compounds araound the lead illutrate the

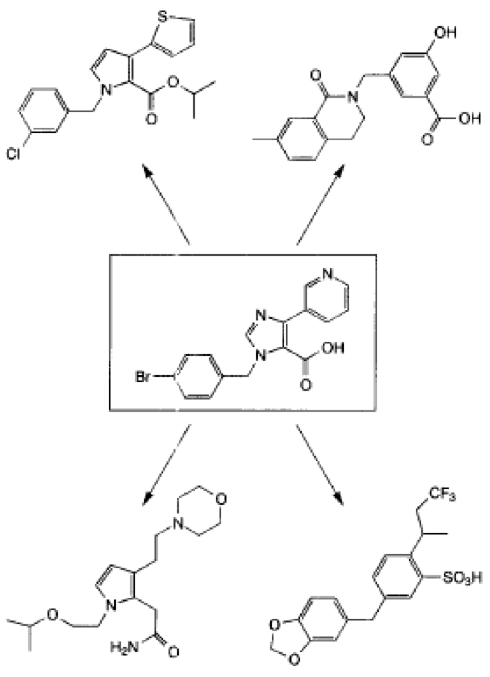


Figure 2.3: Identifying similar compounds using simple 2D methods [57].

molecule library, where molecules that are similar to the lead are searched for.
Clustering or 'pooling' reagents
Clustering or 'pooling' reagents[2] based on similarity is another approach to enhance the speed of processing of a virtual library. It is important to avoid to fully enumerate the virtual library. This can be achieved by clustering building blocks, based on their similarity. After clustering only one representative member of each family in the library needs to be evaluated. This can dramatically reduce the number of compounds. Figure 2.4 shows how reagent pooling might work. The reagent in the middle is used for screening and is representing all reagents around it.
Evolutionary methods
Evolutionary algorithms are metaheuristic optimization algorithms for searches based on the principles of biological evolution. In genetic algorithms, which are related to evolutionary algorithms, these principles are selection, crossover, and mutation. At the beginning of the process an initial population is generated. Typically this is done randomly. After evaluating the fitness of each individual,

[2]A reagent or reactent is a substance consumed during a chemical reaction.

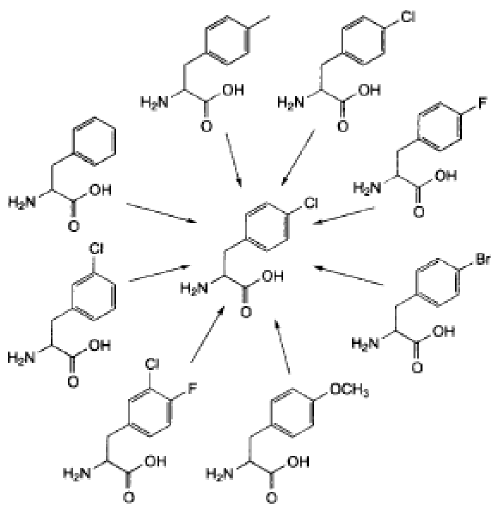


Figure 2.4: Reagent pooling [57].

some individuals are selected, based on their fitness (selection phase). To generate a second generation population, in the next step (crossover phase) some of the selected parents are pared and genetic material is exchanged. In the mutation phase a small fraction of the population undergoes point mutation, which increases the gene pool. This is important to avoid being trapped in local minima. In Figure 2.5 the flowchart of a genetic algorithm is shown. Some evolutionary methods like

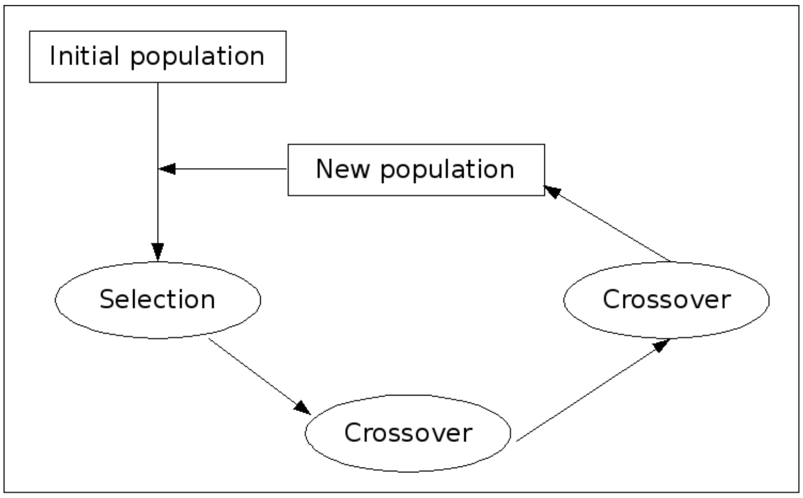


Figure 2.5: Flowchart of a genetic algorithm [57].

genetic algorithms are used in some drug design applications [57] to focus on a smaller number of compounds. The main advantages of these methods are that they are rapid and easy to implement, but there are also some drawbacks. An important one is that the evolutionary algorithms are non-deterministic, which means that each run could produce a different solution. Additionally, the solution found by the algorithm may not be the best solution anyway.

Multi-conformer databases

To approach molecular flexibility databases can be pre-calculated where the conformers need only be generated once, stored and accessed when needed. Considering the CPU time needed to evaluate a 3D virtual library, this method can save a lot of time.

2.4 Pharmacophore-based Virtual Screening

The official definition of a pharmacophore elaborated by a working party of the International Union of Pure and Applied Chemistry (IUPAC) and published 1998 is:

A pharmacophore is the ensemble of steric and electronic features that is necessary to ensure the optimal supramolecular interactions with a specific biological target structure and to trigger (or to block) its biological response.

This means a pharmacophore does not represent something real like a molecule or a functional group, but an abstract concept that describes molecular features necessary to trigger a biological response of a specific target. Based on these abstract models virtual screening can be applied by screening virtual libraries against the created pharmacophores. Molecules matching a specific pharmacophore are expected to trigger biological activity at the receptor site. A typical pharmacophore-based virtual screening workflow is depicted in Figure 2.6. First, the 3D structure of a target is used together with the 3D structure of a set of ligands to create pharmacophore models. Afterwards, the 3D database is searched until known ligands are found. If compounds are found that are not known as ligands, molecules having a structure that fits the model are designed. After a final optimization process new leads are found and the virtual screening process is finished.

In the following two different approaches for pharmacophore screening are described.

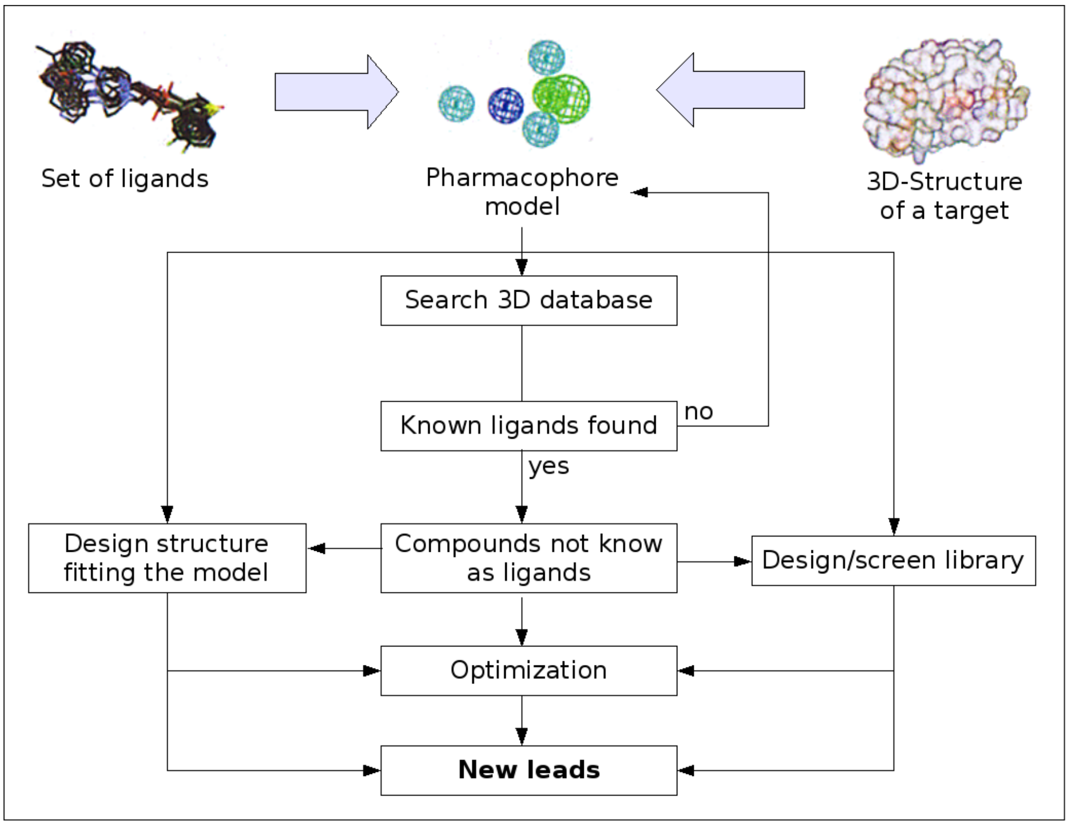


Figure 2.6: A typical pharmacophore-based virtual screening workflow [35].

Ligand-based vs. structure-based pharmacophore screening
There are two different types of pharmacophore based virtual screening (pharmacophore screening). One is to start from a set of ligands[3] that are known to bind to the same target in a comparable way, which is called ligand-based pharmacophore screening. The other is to investigate the geometry of the target and the bound ligand if its structure is available. This is called structure-based pharmacophore screening. The relationship between these two methods is shown in Figure 2.7. A pharmacophore, which is shown in the middle, is needed for virtual screening. On the left of the pharmacophore the ligand-based approach is depicted. Multiple ligands which are known to trigger activity are put together to create the

[3]A ligand is a molecule that is able to bind to and form a complex with a protein to trigger a biological activity.

pharmacophore. On the right of the pharmacophore the structure-based approach is shown. The blue, meandering object represents a protein. The ligand which is known to fit to this target is inside the yellow box in the center of the protein. The pharmacophore is created by the binding information of the ligand with the protein. Although there are some approaches for ligand-based screening, this work will

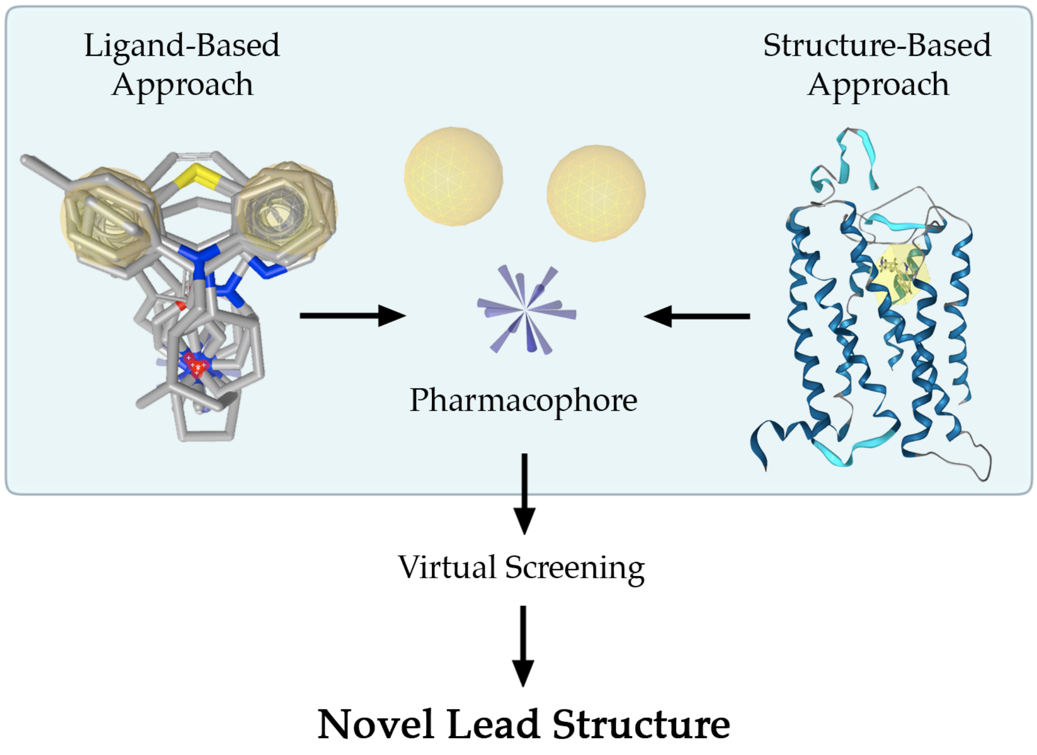


Figure 2.7: The relationship between ligand-based and structure-based pharmacophore screening [13].

focus on structure-based screening, which is more efficient [62].

3-D structure generation

In the 1940s SAR (structure-activity relationship) considerations enabled the construction of 2-D model structures. X-ray analysis and conformational chemistry made 3-D models possible in the 1960s. For structure-based pharmacophore screening it is necessary that the structural data to be screened is available in 3-D form.

The 3-D structure can be created by computer programs, which use 2-D connectivity files as input. Such files contain atom and bonding descriptions for each molecule and can be in the form of SMILES[4] [17, 3, 59] strings, SLN [5] [5] strings, and 2-D SD [6] [15] connection tables. These programs do the conversion from 2-D to 3-D by using a mixture of rules (linking atom hybridization bond lengths, bond angles and non-bonded forces), pre-calculated 3-D fragment databases, and distance geometry techniques.

Pharmacophore atom-typing

To screen molecules against pharmacophores it is necessary to specify the functional properties of the molecules. The most common functional properties are hydrophobe, negative ionizable, positive ionizable, hydrogen bond donor and hydrogen bond acceptor [64]. Figure 2.8 shows pharmacophoric elements created with LigandScout [64, 60, 61] in 3D and Figure 2.9 in 2D.

The chemical feature definitions of LigandScout are described in the following:

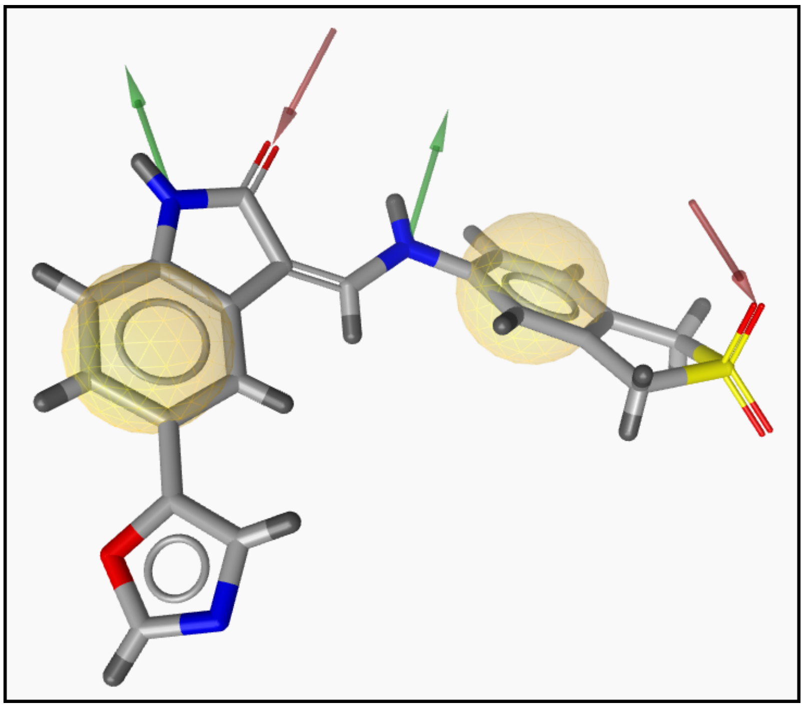


Figure 2.8: Pharmacophoric elements of a molecule in the 3D-view of LigandScout. Hydrophobic Interactions (yellow sphere), Hydrogen Bond Donor (green arrows) and Hydrogen Bond Acceptor (red arrows).

[4]Simplified Molecular Input Line Entry Specification

[5]SYBYL Line Notation. SYBYL is a molecular modeling program developed by Tripos [55].

[6]Structure Data

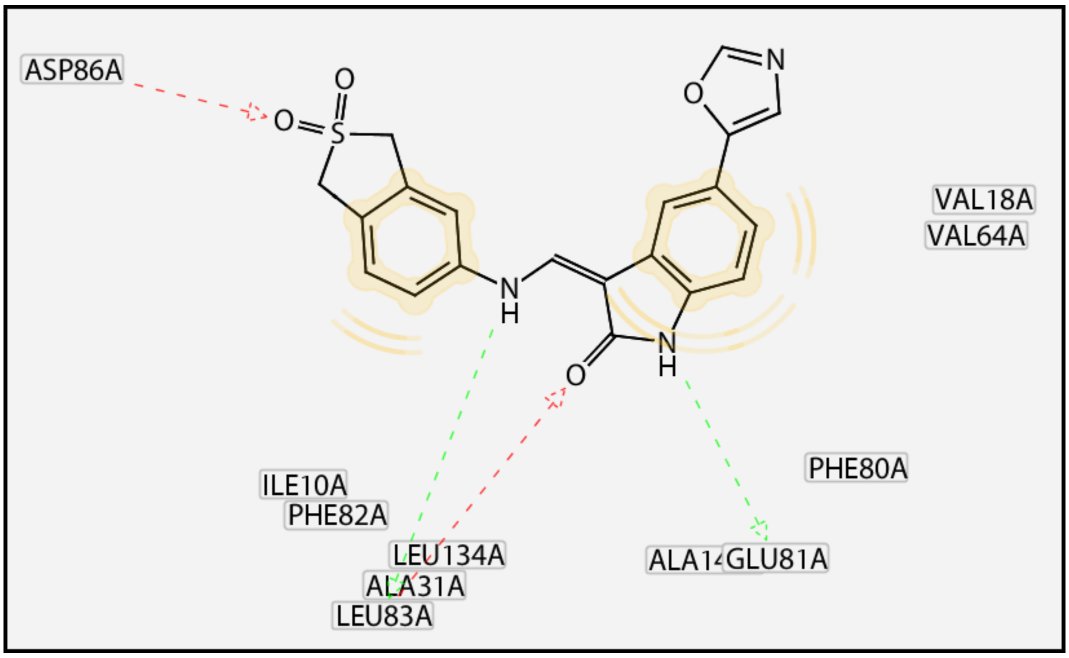


Figure 2.9: Pharmacophoric elements of a molecule in the 2D-view of LigandScout. Hydrophobic Interactions (yellow circles), Hydrogen Bond Donor (green arrows) and Hydrogen Bond Acceptor (red arrows).

- **Hydrogen Bonding**
  Hydrogen bonding occurs when a covalently bound[7] hydrogen with a positive partial charge interacts with another atom with a negative partial charge [40]. This typically happens when the partially positively charged hydrogen atom is positioned between partially negatively charged oxygen and nitrogen atoms [62]. The element of the interaction, containing the hydrogen, is called *hydrogen-bond donor* (e.g. *NH* or *OH*) and the opposite partner is a *hydrogen-bond acceptor*, because it possesses a partially negatively charged atom (*N*, *O*) [13].

- **Ionic Interaction**
  Another commonly found interaction type appears when areas of the ligand, such as charged groups or atoms, bind to areas of the protein of opposite charge [13]. *Positive ionizable* areas appear if chemical features are protonated at a physiological pH, whereas *negative ionizable* areas are atoms or groups of atoms that are likely to be deprotonated at physiological pH

[7]A covalent binding is an atom binding.

[13, 62].

- **Hydrophobic Interaction**
  This kind of interaction only takes place when hydrophobic residues of amino acids are close to lipophilic groups of the ligand [13]. A hydrophobic interaction has no directional constraint like hydrogen-bond interactions. An exclusion is given if an aromatic ring[8] of the ligand interacts with an aromatic ring of the protein [12].

To create pharmacophores with such features LigandScout uses a structure-based approach, where the target and ligand informations are loaded from a PDB[9] [9, 10, 11, 52] file and chemical features are created through applying chemical substructure patterns to the ligands [64].

[8]Aromatic rings are ring-shaped organic compounds.
[9]Protein Data Bank

Chapter 3

Activity Profiling

When virtually screening libraries against a model of target the goal is to find molecules fitting to that target. This fitting of the molecule triggers a biologic response, which should be useful in the efficient and safe treatment of a certain disease. Unfortunately molecules often fit to more than a single target, which can cause side-effects and toxicity.

Side-effects and Toxicity
94% of all drugs widthdrawn from the market between 1992-2002 are caused by side-effects and toxicity as shown in Figure 3.1 [45]. Therefore, the modeling of all relevant targets responsible for drug action and side-effects is desired but not provided by current screening protocols.

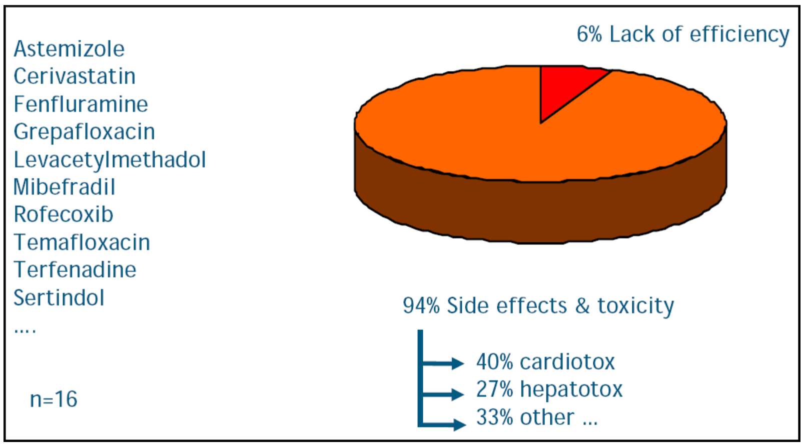


Figure 3.1: Drugs widthdrawn from the market between 1992-2002 [34].

Current Screening Protocol

In a typical current screening protocol, shown in Figure 3.2, a molecule library with thousands of molecules is screened against a single biological target. This kind of screening does not show side-effects and toxicity. Therefore, pharmacological profiling is applied to drug candidates.

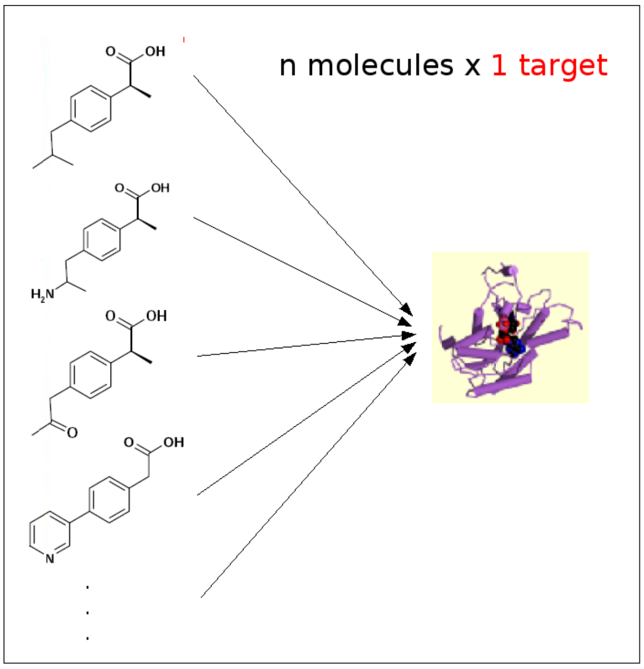


Figure 3.2: Current Screening Protocol.

Pharmacological Profiling

In Figure 3.3 pharmacological profiling is depicted, where a molecule is screened against one or more target-sets. Pharmacological profiling shows side-effects and toxicity of a drug candidate and is applied late in the research process of drugs. A technique which combines the current screening protocol with pharmacological profiling is desired. This is achieved by activity profiling [50].

Activity Profiling

Activity profiling is the combination of the two methods described above. Figure 3.4 illustrates the idea of activity profiling.

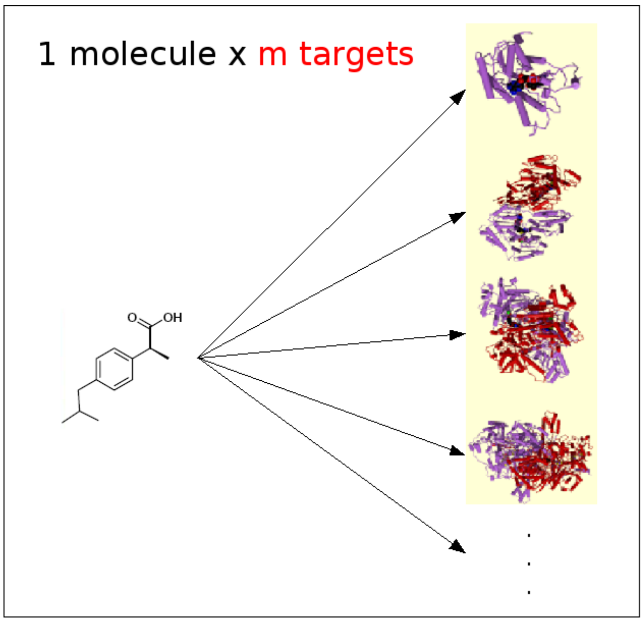


Figure 3.3: Pharmacological Profiling.

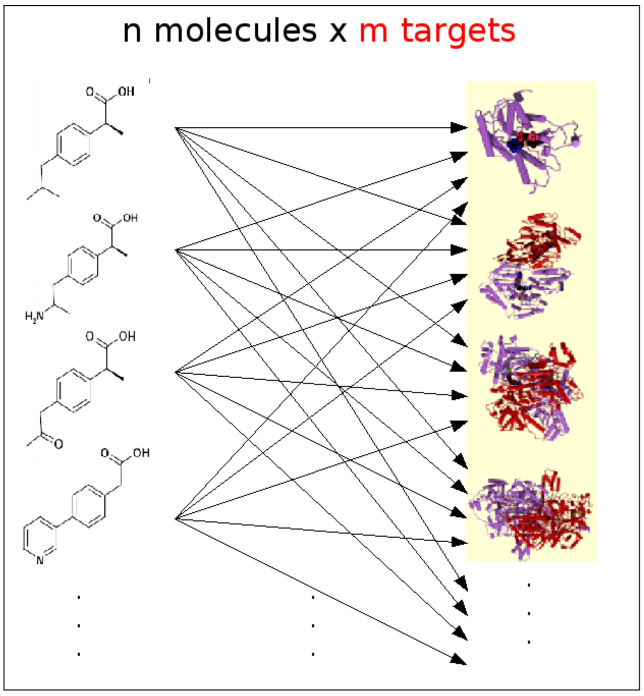


Figure 3.4: Activity Profiling.

Steindl et al. [51] describes five aspects of activity profiling which are:

- addressing the *selectivity*[1] *issue* in drug design
- identifying potential *side-effects, toxicity or metabolic pathways*[2] early on in the research process
- screening against databases of known multiple therapeutic targets to discover *new applications for a known compound* or marketed drug
- searching for a potential mode of action of acquired compounds (*parallel screening*)
- designing privileged structure fitting to desired targets, while minimizing the probability of binding to any of the other targets (*library profiling*)

In the following parallel screening, which is an important part of activity profiling, is described.

Parallel Screening Parallel screening is the simultaneous screening of one or more molecules against a set of pharmacophores, which can represent different targets. The aim is the fast in-silico[3] determination of the biological activity profile of a molecule in order to speed up the time and cost-intensive drug discovery development process and increase its efficiency [50].

The successful application of the parallel screening concept is shown in an experiment by Steindl et al. [50], where 89 known HIV protease inhibitors and 85 druglike inactive compounds were screened against a total of 81 HIV protease inhibitor pharmacophore models. The results, which were displayed in the heatmap-mode of Pipeline Pilot[4] are shown in Figure 3.5. Molecules are listed on the y-axis, pharmacophores on the x-axis. The color coding indicates the score of the compound, with an encoding from red boxes for high score compounds to light blue boxes for low score compounds and black boxes where no compounds are matching to the corresponding pharmacophore.

[1]In pharmacology selectivity is the preference of a drug for one mechanism of action over others that cause side effects.

[2]A metabolic pathway is a series of chemical reactions occuring within a cell.

[3]In-silico is an expression used to mean “performed on computer or via computer simulation“.

[4] Pipeline Pilot is a software application developed by SciTegic Inc and Accelrys Inc [26, 25].

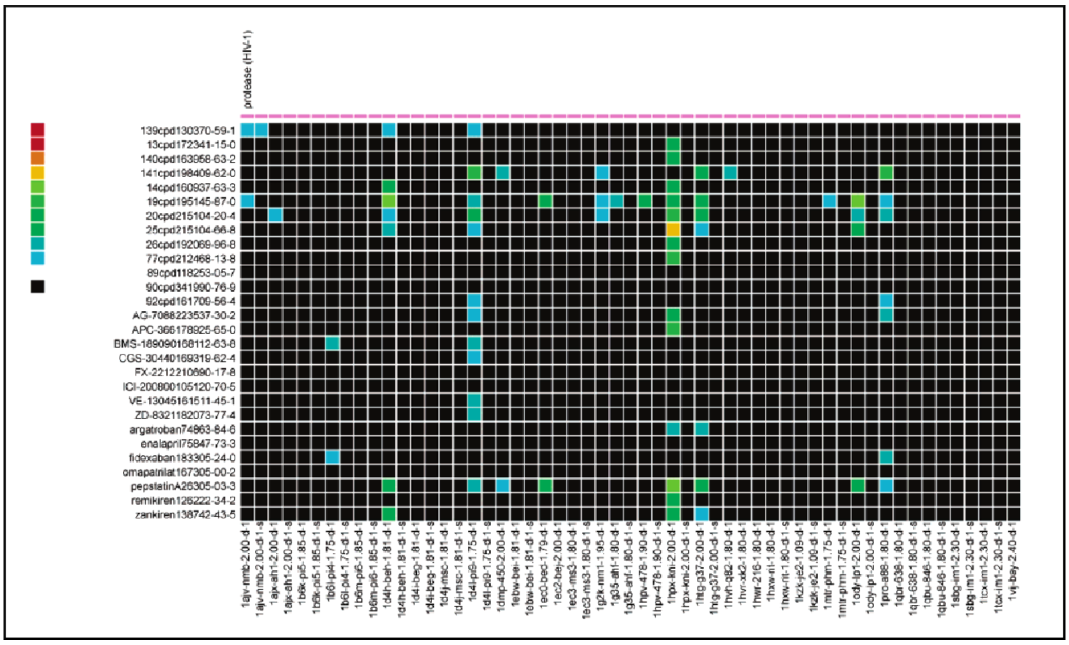


Figure 3.5: Results of a parallel screening example shown in a heatmap with molecules on the y-axis, pharmacophores on the x-axis and color coding from red boxes for high score compounds to light blue boxes for low score compounds, black boxes where no compounds are matching to the corresponding pharmacophore [50].

Chapter 4

Visualization

Visualization is *the use of computer-supported, interactive, visual representations of data to amplify cognition* [14].

This definition results from the fact, that humans have remarkable perceptual abilities like scanning, discovering, remembering images rapidly and automatically detecting patterns and changes in size, colour, shape, movement or texture. So, visualization helps us to gain insight into data and to make (better) decisions resulting from this insight.

According to Ware [58] some of the advantages of visualization are:

- Visualization provides an ability to comprehend huge amounts of data.
- Visualization allows the perception of emergent properties that were not anticipated.
- Visualization often enables problems with the data itself to become immediately apparent.
- Visualization facilitates understanding of both, large-scale and small-scale features of the data.
- Visualization facilitates hypothesis formation.

Ware [58] also defines four basic stages in the process of data visualization:

1. The collection and storage of data itself.
2. The preprocessing designed to transform the data into something we can understand.
3. The display hardware and the graphics algorithms that produce an image on the screen.
4. The human perceptual and cognitive system.

A schematic diagram of the visualization process a number of feedback loops is shown in Figure 4.1. There are three feedback loops shown in Figure 4.1. The

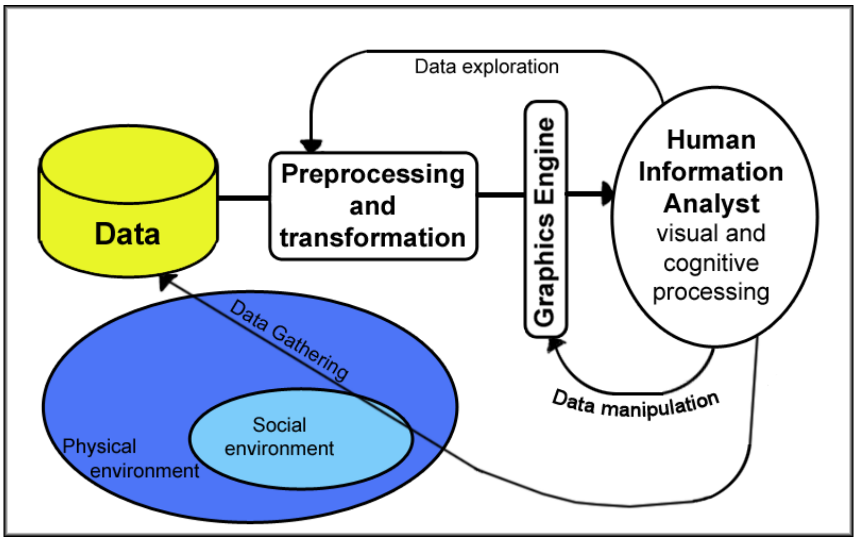


Figure 4.1: A schematic diagram of the visualization process [58].

longest one is data gathering, where a data seeker may gather more data to follow up on an interesting lead. The data gathering loop is closely coupled with both, the physical environment and the social environment. The physical environment is a source of data and the social environment designates what is collected and how it is interpreted. The second feedback loop is data exploration, where the human analyst may change the transformation, if he or she feels that the data subjected by a certain transformation could give up its meaning. The last one is data manipulation where the analyst may display different aspects of the data through an interactive visualization.

4.1 Scientific- vs. Information Visualization

In the past the lack of graphics power and CPU speed often limited the usefulness of computer-aided visualization of data. Since computers got more and more powerful, the utilization of computer graphics in science and other fields has increased a lot. Based on the data domain we can classify visualization into:

- **Scientific Visualization**

  Scientific visualization is *the use of visual representations of* ***scientific*** *data, typically* ***physically based****, to amplify cognition* [14, 42].

  Scientific data could be the human body, the earth, molecules, or other. The main goal of scientific visualization is to analyze such data (e.g. medical monitoring data, medical image data, or GIS-data).

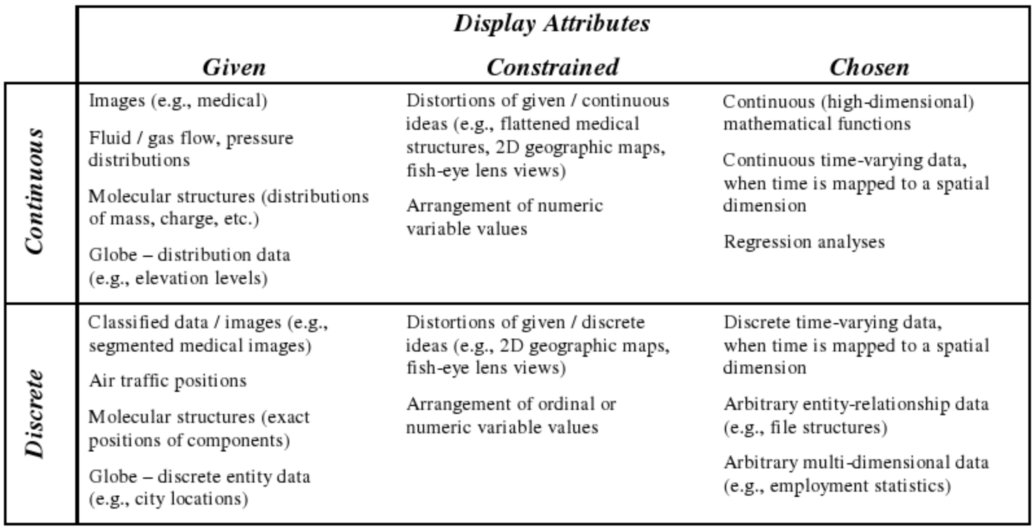
- **Information Visualization**

  Information visualization is *the use of visual representations of* ***abstract, non-physically based*** *data to amplify cognition* [14, 42]. Some information visualization techniques are described in chapter 5.

4.2 Model-based Visualization Taxonomy

The classification above has its seeds in the early days of the evolution of visualization and is dividing the entire research area into distinct parts. It appears that these groups are not completely disjoint, but are overlapping in some aspects.

Tory and Möller [54] have introduced a classification scheme that organizes visualization techniques in a new way. The taxonomy is based on characteristics of models of the data rather than on characteristics of the data itself. Therefore, it is called a *model-based visualization taxonomy*. Design models are classified according to two criteria. First, whether the design model is discrete or continuous and second, how much the display attributes are given or chosen. Figure 4.2 shows the taxonomy structure of Tory and Möller [54]. According to Tory and Möller scientific visualization is placed on the top left area and information visualization on the bottom right. Middle areas cannot be classified to any of the both with assurance, which means that in the middle areas scientific visualization and information visualization are overlapping.



	Display Attributes		
	Given	***Constrained***	***Chosen***
Continuous	Images (e.g., medical) Fluid / gas flow, pressure distributions Molecular structures (distributions of mass, charge, etc.) Globe – distribution data (e.g., elevation levels)	Distortions of given / continuous ideas (e.g., flattened medical structures, 2D geographic maps, fish-eye lens views) Arrangement of numeric variable values	Continuous (high-dimensional) mathematical functions Continuous time-varying data, when time is mapped to a spatial dimension Regression analyses
Discrete	Classified data / images (e.g., segmented medical images) Air traffic positions Molecular structures (exact positions of components) Globe – discrete entity data (e.g., city locations)	Distortions of given / discrete ideas (e.g., 2D geographic maps, fish-eye lens views) Arrangement of ordinal or numeric variable values	Discrete time-varying data, when time is mapped to a spatial dimension Arbitrary entity-relationship data (e.g., file structures) Arbitrary multi-dimensional data (e.g., employment statistics)

Figure 4.2: High-level visualization taxonomy, illustrated by examples. Design models are classified based on whether they are discrete or continuous and by how much the algorithm designer chooses display attributes like spatialization, timing, colour and transparency [54].

4.3 Goals of Visualization Techniques

Beside the classifications by the data domain and the design model, Keim [31] defines three different approaches of how visualization techniques can be used.

- **Explorative Analysis**

  At the beginning there is no hypothesis about the data. The user performs interactive and usually undirected searches for structures, trends, etc. to explore relations between data attributes and to gain insight into the data. The aim is to get a visualization, which provides hypothesis about the data.

- **Confirmative Analysis**

  Starting with hypothesis about the data, the goal is a visualization, which allows the confirmation or rejection of the hypothesis. Therefore, the user examines the data in a goal-oriented manner.

- **Presentation**

  Presentation is a widely used visualization purpose and can be found in various fields. The facts to be presented are fixed at the beginning and the user has to choose an appropriate presentation technique as a post-processing step. The result should be a high-quality visualization of the data presenting the facts, whereby interaction is not important. The purpose is not to find relations, but to present them in the best way.

Chapter 5

Information Visualization

Research of the effective visualization of static presentations has been performed for a long time. The visualization of interactive visual representations has become more interesting after recent advances of computer hardware performance, especially after improvements in graphics power. As mentioned above, we differentiate between scientific visualization and information visualization. The goal of scientific visualization is to make important spatial structures visible, whereas the goal of information visualization is to find adequate and comprehensible visual metaphors for a particular problem and to provide optimal interaction possibilities. This includes the mapping of non-spatial data to spatial ones. Many visualization techniques are introduced and analyzed by researchers in this area. Some of them, which are relevant in the context of the visualization of n:m drug target interactions are described in the following. However, multiple view visualization, where at least two different views of the same data are showing different aspects or attributes of the data, is analyzed in chapter 6 in detail.

5.1 Heat Map

A *heat map* is the visual representation of data in 2D, where the color of a data item encodes a specific attribute of the data. The location of a data item in the 2D map results from one or more attributes of the data and should be intuitive to the user. Heat maps are widely used in molecular biology. An example of a gene expression during the yeast cell cycle is depicted in Figure 5.1. Genes correspond to the rows, and the time points of each experiment are the columns. The ratio of induction is shown for each gene such that the magnitude is indicated by the intensity of the

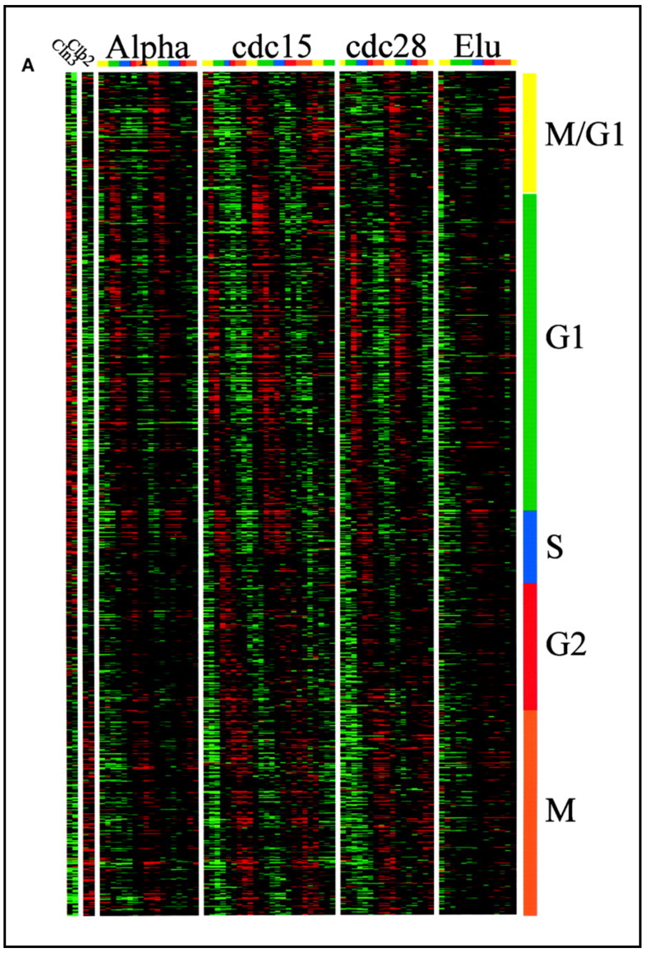


Figure 5.1: Gene expression during the yeast cell cycle [49].

colors displayed. If the color is black, then the ratio of control to experimental cDNA[1] is equal 1. Red indicates an increase in mRNA[2] abundance, whereas green indicates a decrease in abundance.

5.2 Height Map

Height maps are fairly similar to heat maps, while the visualization is 3D instead of 2D. The third dimension, the height (the z-axis) is used instead of color to encode a specific attribute of the data. Just as in a heat map, the location of a data element of the height map is determined by at least one attribute of the data. Height maps are commonly used in GIS'[3], where they are called DEM[4]. Figure 5.2 shows a height map rendered with Anim8or [4], which is a freeware OpenGL[5] based 3D modeling and animation program. The Figure shows a mountainous terrain.

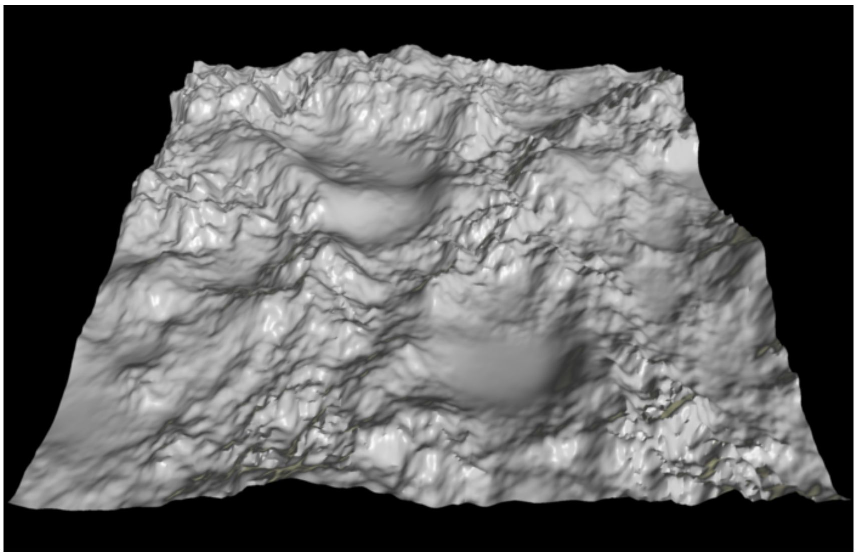


Figure 5.2: A height map rendered with Anim8or [4].

[1]Complementary Deoxyribonucleic acid

[2]Messenger Ribonucleic Acid

[3]Geographic Information System

[4]Digital Elevation Model

[5]Open Graphics Library is a standard defining an API for writing applications that profuce 2D and 3D computer graphics.

5.3 Clustering

A common issue of information visualization is that the amount of data to be visualized is huge. An effective technique to let the user analyze and discover interesting patterns in large datasets is to group data items which are similar in a particular property, which is called *clustering* [18, 24, 16]. A modern clustering approach is *hierarchical clustering* [30]. This approach finds pairs with the most similar data elements, and iteratively builds a hierarchy by pairing data items or existing clusters that are most similar. A powerful tool which makes use of this technique is HCE[6] [19] and was developed by Seo and Shneiderman [46]. HCE addresses a common issue of clustering, which is the determination of the amount of clusters. Previously there were two approaches for this issue. Either letting the users determine the number of clusters as an input or automatically determining them. The first one is often difficult to apply, because users often do not know the right number beforehand. The latter has the disadvantage that users cannot control the clustering process. HCE avoids this dilemma by applying the clustering algorithm without a predetermined number of clusters, and allows for user-effective interactive control of the grouping afterwards. Figure 5.3 shows HCE with its different views, which are described in the following.

Dendrogram View

Dendograms are trees often used to show the arrangement of clusters. The tree has joining points whose distance from the root indicates the similarity of subtrees. Highly similar nodes or subtrees have joining points that are farther from the root [46]. The “Minimum Similarity Bar” above the dendrogram of Figure 5.3 allows for an interactive change of the clusters.

Detail Views

The dataset is coupled with the detail views which lets the user easily find and examine high-level patterns and hot spots.

Scatterplots

The 2D scatterplots[7] are bi-directionally linked to the Dendrogram View and provide a different perspective of the data set.

[6]Hierarchical Clustering Explorer

[7]A scatterplot is a chart that uses cartesian coordinates to display values for two variables.

Profile Search

A profile search, where the user can easily identify genes with a certain temporal pattern is provided by a parallel coordinates[8] chart.

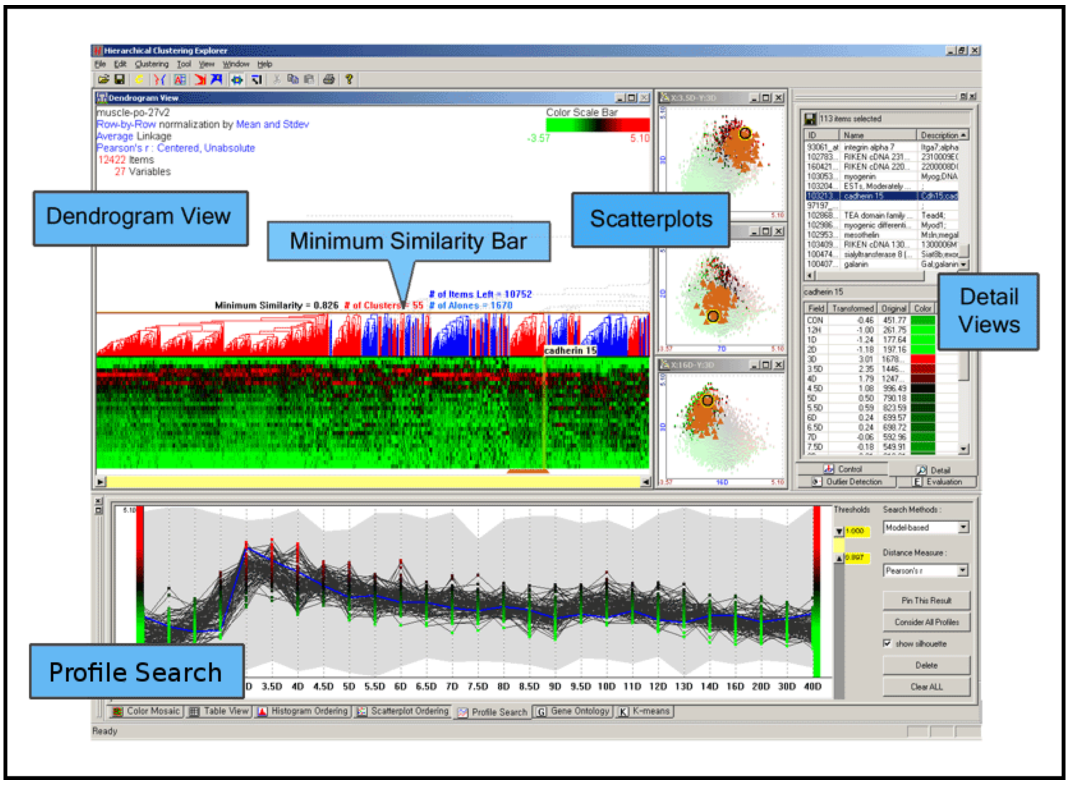


Figure 5.3: HCE - Hierarchical Clustering Explorer [19].

[8]Parallel coordinates is a common way of studying high-dimensional geometry. To show a set of points in an n-dimensional space, a backdrop is drawn consisting of n parallel lines, typically vertical and equally spaced. A point in n-dimensional space is represented as a polyline with vertices on the parallel axes. The position of the vertex on the i-th axis corresponds to the i-th coordinate of the point [27, 28].

Chapter 6

Multiple Views

Displaying data in such a form that the user can easily extract the information needed, is a difficult task. For different use cases the user often needs different details of the shown data. Putting all of the needed information into a single view often ends up with a view, which is not optimal for any of the use cases.

Multiple view visualization is a commonly used approach to overcome this problem.

6.1 Definition

Baldonado [7] defines a multiple view system as following:

A multiple view system uses two or more distinct views to support the investigation of a single conceptual entity.

A *view* is a set of data and the specification of how to display that data. Views are *distinct* if they differ in their data or in the visual representation of that data.

6.2 Drawbacks

The main drawbacks of multiple views are the additional complexity and the additional system requirements resulting from displaying multiple views. The system gets more complex for the user, because of the additional time and effort to learn the system and switch between the views and contexts. The additional system requirements are the resources needed to render the additional views and display space requirements for the additional display elements. There are three dimensions for the development of multiple view systems, which are described in the following section.

6.3 Dimensions

The development of multiple view systems consist of three dimensions: selection, presentation, and interaction [7].

6.3.1 Selection

In the selection phase the designer should identify a set of needed and useful views, which are distinct and can be combined for displaying data of a given task.

6.3.2 Presentation

After determining views to display, it has to be decided how to display them. For example, the designer has to choose between displaying the views sequentially or all at once. Another important decision is the visual presentation of the data (bar-charts, scatterplots, tabular,...).

6.3.3 Interaction

In the last phase the interaction methods have to be decided. Each view may have its own interface affordances for navigating through the data, selecting data, etc. In multiple view systems usually these affordances are tied together so that actions in one view are automatically propagated to other views. A common interaction technique is *navigational slaving* [7, 36], where movements in one view are propagated to other views. Another interaction method is *linking and brushing*

[22, 33, 36], where highlighting data in a view by user input (e.g. clicking on a visual object) results in highlighting that data in all views. In chapter 7 *Linking and brushing* is described in more detail.

In the next two sections rules for a successful implementation of these dimensions are described.

6.4 Rules for Selecting Views

In the following some rules for selecting views, which are mentioned by Baldonado et al. [7] are described.

6.4.1 Diversity

If there is a diversity of attributes, models, user profiles, levels of abstraction or genres multiple views should be preferred. Using a single view approach in this case often leads to a least-common-denominator view, that is insufficient for any needs. Figure 6.1 depicts a system that exemplifies the use of multiple views for data with different attributes and different levels of abstraction [39]. This tool shows legal information using the following views: query, query results in textual form, query results in a graphical visualization, and, for the selected case, overview, headnotes, and decision text, all in textual form [7].

6.4.2 Complementary

If there are correlations and/or disparities in the data, the user has to mentally extract and remember components to compare them, if there is only one single view. Since recognition is easier than recall, visual comparison is easier than memory-based comparison. Therefore, multiple views should be preferred.

6.4.3 Decomposition

Another case where the designer should choose multiple views is, if the data are very complex, for example a spreadsheet with many columns. A single view approach would be cognitively overwhelming to a user, but in a multiple view system the spreadsheet could be divided in multiple views (*divide and conquer*).

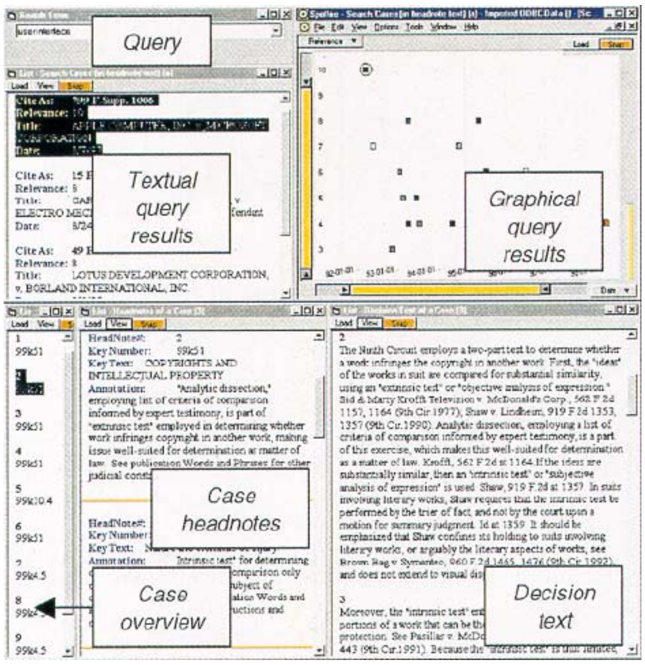


Figure 6.1: A multiple view system for data with different attributes and different levels of abstraction [7].

6.4.4 Parsimony

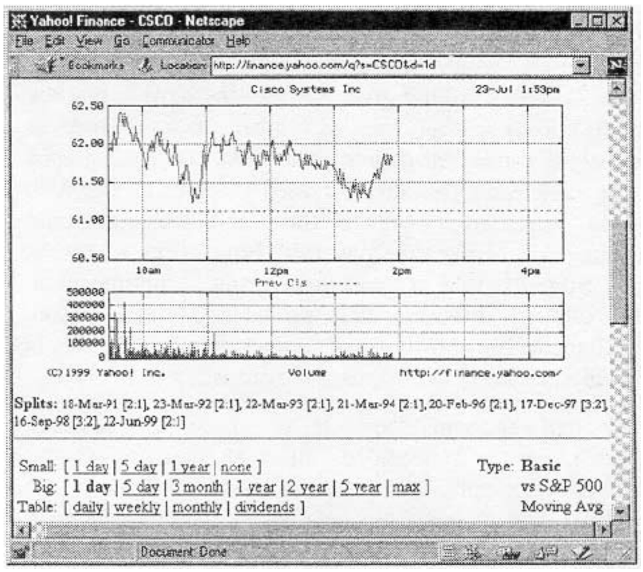
Multiple views incur the cost of context switching and introduce additional system complexity. Furthermore, more views lead to more learning costs for the end user. Finally, the computational costs increase and more display place is needed. Therefore, similar views should be merged to a single one.

6.5 Rules for Presentation and Interaction

After making the right decisions for selecting the views, it is important to choose the right visualization methods, to make extracting information easier for the end user and the right interface affordances, to minimize the learning costs of the application. Baldonado et al. [7] has described four rules which should help designers to make the right decisions.

6.5.1 Space/Time Resource Optimization

An important aspect in connection with the presentation of multiple views is space and time. A disadvantage of displaying views side-by-side is that it costs more space, so the visual elements of a view are smaller. However, if the views are shown sequentially, the user has to remember the data, if he or she wants to compare the views. Which also should be considered is the additional computing time of a side-by-side view. On the other hand the user saves time comparing the data. As we see, it is a trade-off between usability and space/time resources which strongly depends on the platform, e.g. on a Palm Pilot a sequential approach makes much more sense than the side-by-side one.



Splits: 18-Mar-91 [2:1], 23-Mar-92 [2:1], 22-Mar-93 [2:1], 21-Mar-94 [2:1], 20-Feb-96 [2:1], 17-Dec-97 [3:2], 16-Sep-98 [3:2], 22-Jun-99 [2:1]

Small: [1 day | 5 day | 1 year | none] Type: Basic
Big: [1 day | 5 day | 3 month | 1 year | 2 year | 5 year | max] vs S&P 500
Table: [daily | weekly | monthly | dividends] Moving Avg

Figure 6.2: Mutliple views of stack data, with a shared x-axis (time) to help the user easily compare the views [7].

Figure 6.2 shows an example where two views, the closing price view and the volume traded view are sharing the x-axis, which is an example for a side-by-side view. However, there are different views for 1-day, 5-day, 3-month, 1-year, 2-year, 5-year, and max time scales, which are shown only one at a time, because of the lack of display space.

6.5.2 Self-Evidence

The use of perceptual cues helps making the relationship between views more apparent to the end user. There are many types of perceptual cues. The most important ones are *highlighting*, *spatial arrangement*, and *coupled interaction*. When using perceptual cues there are some important issues to take care of:

- If the coupling is non-trivial, the mapping is hard to understand.
- Information needed to map between objects in all views has to be maintained.
- Visual and interactive components should not confuse the user. If two events occur within 100ms, the user perceives them as causally related, but if it takes longer, the user may fail to recognize to relationship. So, if computation takes too long, the views should be decoupled and this decoupling should be made evident to the user, e.g., by out-graying the decoupled view.

6.5.3 Consistency

System states and interface affordances should be consistent, because inconsistency can lead to false cognitive inferences by the user. For example, if one view highlights one particular object, related views should highlight the same object. The consistency in the interface affordances makes multiple view systems easier to learn.

6.5.4 Attention Management

The user interface designer should use perceptual techniques to focus the attention of the user to the right view at the right time. This can be accomplished by highlighting, for example.

Chapter 7

Linking and Brushing

Linking and brushing is a state of the art information visualization technique, often used in combination with multiple view systems, in case that it is necessary that a subset of the displayed data (the data of interest) is distinguishable from the rest of the data. *Brushing* [22, 33, 36] is the selection and highlighting of one or more objects in the visual representation of the data. *Linking* means that the selection of data performed by the user in one view is automatically applied in all other views, so that the relationship between the visual elements of the views gets apparent for the user. Figure 7.1 shows linking and brushing in a scatterplot matrix. The brush has been applied to the second plot of the first row by drawing a rectangle around the data to be highlighted. The selected data elements are highlighted (red) in all plots.

7.1 Brushing

In information visualization we often have a huge amount of data to show, but not all of the data is important all the time. Highlighting the data of interest is called brushing. Common information visualization tools offer the possibility of either selecting a single item or selecting multiple items, which is referred to as *multiple selection*. Multiple selection is easier to apply in 2D than in 3D. In 2D, for example, the user can simple draw a rectangle or lasso around the data to be highlighted. Advanced techniques have to be introduced for multiple selection in 3D, which are described in the following.

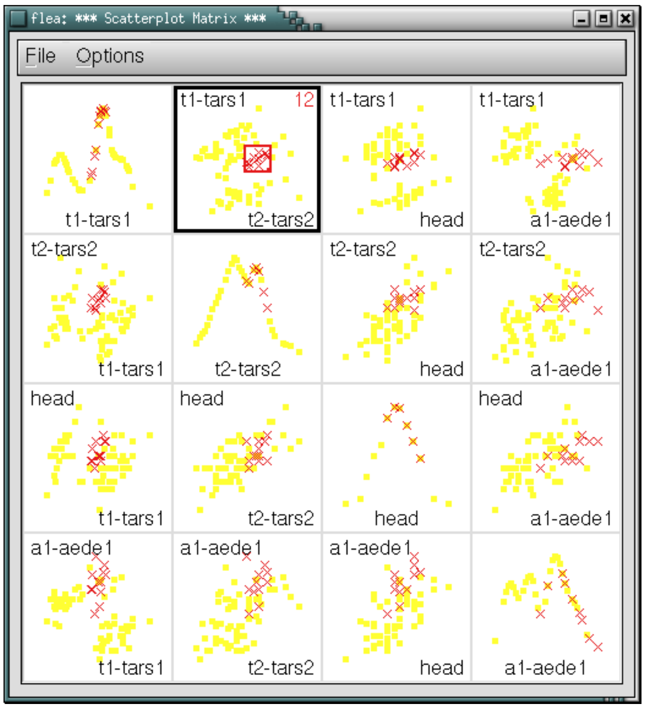


Figure 7.1: Linking and Brushing in a scatterplot matrix [56].

7.1.1 Range Brushing

Range brushing is a brushing technique introduced by Kosara et al. [33], where the user can define a brushing range in one or more axis. Objects that belong to that range are highlighted. Figure 7.2 shows high pressure in a catalytic converter dataset with a range slider in one dimension. In this example a range is defined for the x-axis (red). All elements within that range are highlighted (green).

7.1.2 Beam Brushing

Beam brushing, which was also introduced by Kosara et al. [33], is a more direct selection method. The selection happens by creating a cylinder perpendicular to the viewport having a radius specified by the user. All items inside the cylinder are selected. In Figure 7.3 such a selection is depicted.

7.1.3 Composite Brushing

Sometimes applying a set of single selections and combining them through logical operations is desired. *Composite brushing* [33] is such a technique where the user

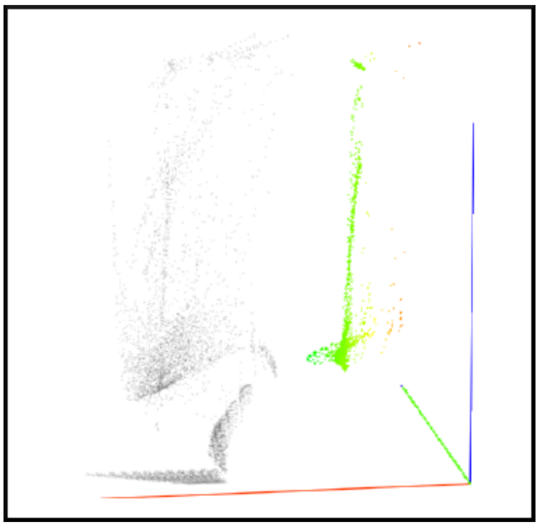


Figure 7.2: Selecting high pressure in a catalytic converter dataset with a range slider in one dimension [33].

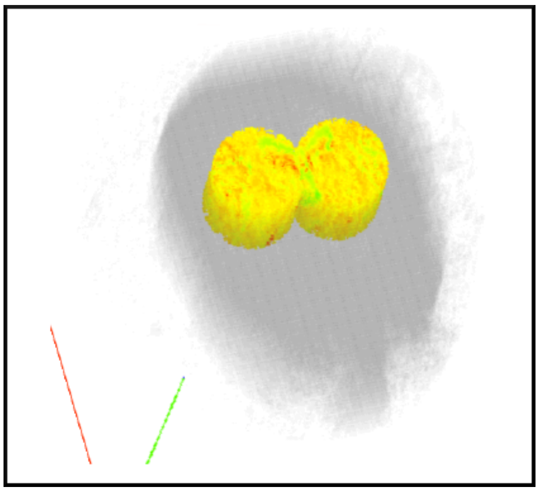


Figure 7.3: The result of two beam brushes through a dataset that represents a part of a CT scan of a head [33].

can select data and combine the selections by AND, OR, and Subselection operations. Composite brushing can also be used to combine different brushing methods.

7.1.4 Smooth Brushing

As mentioned before, brushing is a technique to highlight data of interest. But in some cases it is hard to decide if a data point is of interest or not. In such cases *smooth brushing* [33] can be used, where data points have an interest value of not only 0 or 1, but also in between. A DOI (degree of interest) function is used to calculate the interest value. The interest value can be used, for example, to specify the intensity of the color, which is used to highlight the data.

Chapter 8

Visualization of Activity Profiles

In the first two chapters virtual screening and in particular parallel screening were described, where n molecules are screened against m targets. Afterwards, some visualization techniques were discussed.

In the following two tools dealing with the problem of visualizing n:m (n to m) drug target interactions are discussed. Both are developed by students from the Vienna University of Technology[1] in cooperation with Inte:ligand, a company, which is developing software solutions for computer-driven drug research.

8.1 Ptft - Pharmacophore and Target Filtering Tool

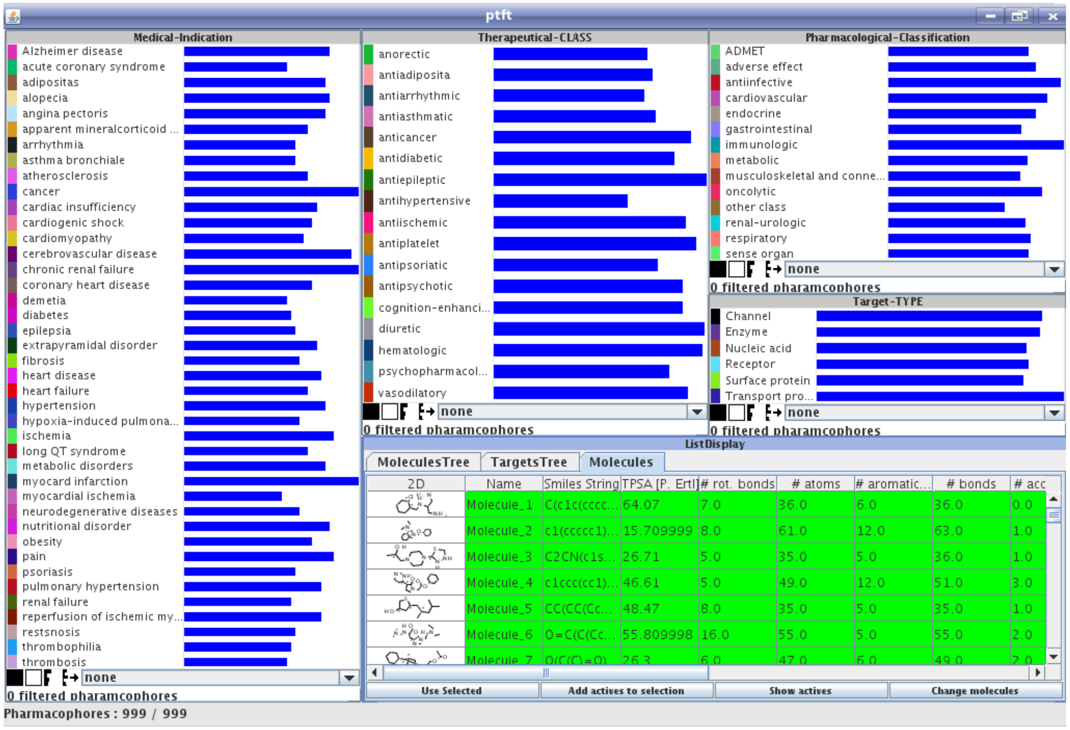
8.1.1 Overview

Ptft is the first of the two prototypes developed by Inte:ligand, where the problem of visualizing parallel screening results is analyzed and visualization techniques like multiple views (s. chapter 6) and linking and brushing (s. chapter 7) are used for the visualization.

In Ptft the screening is done statically, which means that the results have to be saved in a file and can be loaded by the program. The focus of Ptft is the presentation of these results and the exclusion of hits which are not of interest. The filtering can be done by disabling specific target attributes. Furthermore, details for ligands remaining after the filtering are provided, too.

Figure 8.1 provides an overview of Ptft. There are several views, which are described in the following.

[1]Ptft was developed 2005 by A. Liebig and the author of this work and Hitvis 2006 by V. Sladariu



Medical-Indication: Alzheimer disease, acute coronary syndrome, adipositas, alopecia, angina pectoris, apparent mineralcorticoid ..., arrhythmia, asthma bronchiale, atherosclerosis, cancer, cardiac insufficiency, cardiogenic shock, cardiomyopathy, cerebrovascular disease, chronic renal failure, coronary heart disease, demetia, diabetes, epilepsia, extrapyramidal disorder, fibrosis, heart disease, heart failure, hypertension, hypoxia-induced pulmona..., ischemia, long QT syndrome, metabolic disorders, myocard infarction, myocardial ischemia, neurodegenerative diseases, nutritional disorder, obesity, pain, psoriasis, pulmonary hypertension, renal failure, reperfusion of ischemic my..., restsnosis, thrombophilia, thrombosis
none
0 filtered pharamcophores

Therapeutical-CLASS: anorectic, antiadiposita, antiarrhythmic, antiasthmatic, anticancer, antidiabetic, antiepileptic, antihypertensive, antiischemic, antiplatelet, antipsoriatic, antipsychotic, cognition-enhanci..., diuretic, hematologic, psychopharmacol..., vasodilatory
none
0 filtered pharamcophores

Pharmacological-Classification: ADMET, adverse effect, antiinfective, cardiovascular, endocrine, gastrointestinal, immunologic, metabolic, musculoskeletal and conne..., oncolytic, other class, renal-urologic, respiratory, sense organ
none
0 filtered pharamcophores

Target-TYPE: Channel, Enzyme, Nucleic acid, Receptor, Surface protein, Transport pro...
none
0 filtered pharamcophores

2D	Name	Smiles String	TPSA [P. Ertl]	# rot. bonds	# atoms	# aromatic...	# bonds	# acc
	Molecule_1	C(c1c(cccc...	64.07	7.0	36.0	6.0	36.0	0.0
	Molecule_2	c1(ccccc1)...	15.709999	8.0	61.0	12.0	63.0	1.0
	Molecule_3	C2CN(c1s...	26.71	5.0	35.0	5.0	36.0	1.0
	Molecule_4	c1ccc(cc1)...	46.61	5.0	49.0	12.0	51.0	3.0
	Molecule_5	CC(CC(Cc...	48.47	8.0	35.0	5.0	35.0	1.0
	Molecule_6	O=C(C(Cc...	55.809998	16.0	55.0	5.0	55.0	2.0
	Molecule_7	O(C(C)=O)	26.3	6.0	47.0	6.0	49.0	2.0

Pharmacophores : 999 / 999

Figure 8.1: An overview of Ptft (Pharmacophore and Targets Filtering Tool).

8.1.2 Attribute Class

For a better overview, the target attributes are broken apart into different *attribute classes* (e.g., target type, pharmacological classification, therapeutical class, medical indication). Each class is displayed in a separate view, which allows the user to see the correlations between the different attributes of the targets. Figure 8.2 shows such a view. Attributes of a specific class (therapeutical class) are listed one below the other.

The box on the very left of a row indicates whether the corresponding attribute is enabled (box is filled) or not (box is not filled). If an attribute is disabled, all targets having this attribute are filtered from the system, which results in filtering all hits related to these targets. The filtering is done interactively by the user by clicking on the boxes. Changes are automatically applied to all views after user interaction.

An important aspect of this view is the amount of hits of each loaded target attribute, which is depicted by the bars. The width of the bars gives the user information about the activities caused by the ligands used for screening. The width of a bar is calculated relative to the attribute with the most hits of attributes of all classes.

On the bottom of the view the amount of filtered pharmacophores is shown. In

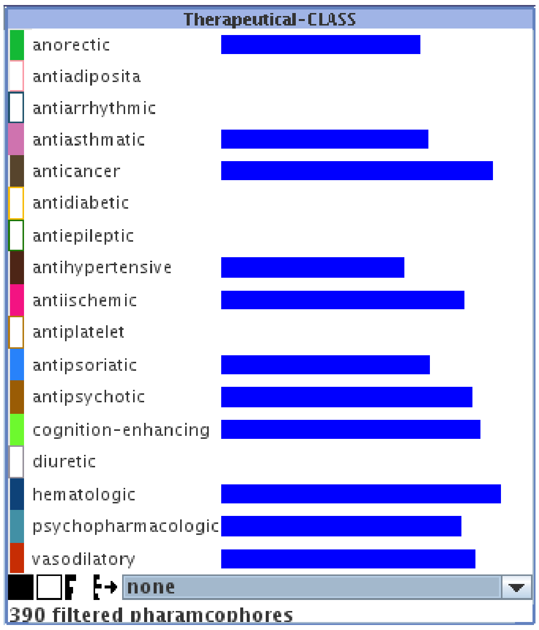


Figure 8.2: The attribute class view for *therapeutical class*.

combination with the total amount of filtered pharmacophores, which is avaiable on the statusbar of the application (see Figure 8.1), this information shows the proportion of the filtering of each class.

The buttons below the attributes[2] are, from left to right, for enabling and disabling all attributes, removing lines with disabled attributes and adjusting the widths of the bars relative to the attribute with the most hits of this class.

Another important part of this view is the selection of an attribute class by the combobox. After selecting an attribute class the bar of each attribute is split into multiple parts. Each part represents an attribute of the class selected by the combobox. This information is encoded by the color, which is the same as the color of the box of the attribute displayed in its own view. The width of each part shows the corresponding proportion of hits to targets having both attributes, the one, which is represented by the row and the one, which is related to. Figure 8.3 shows an example, where target type is selected in the combobox of the therapeutical class view.

[2]The black and white colored buttons left to the combobox

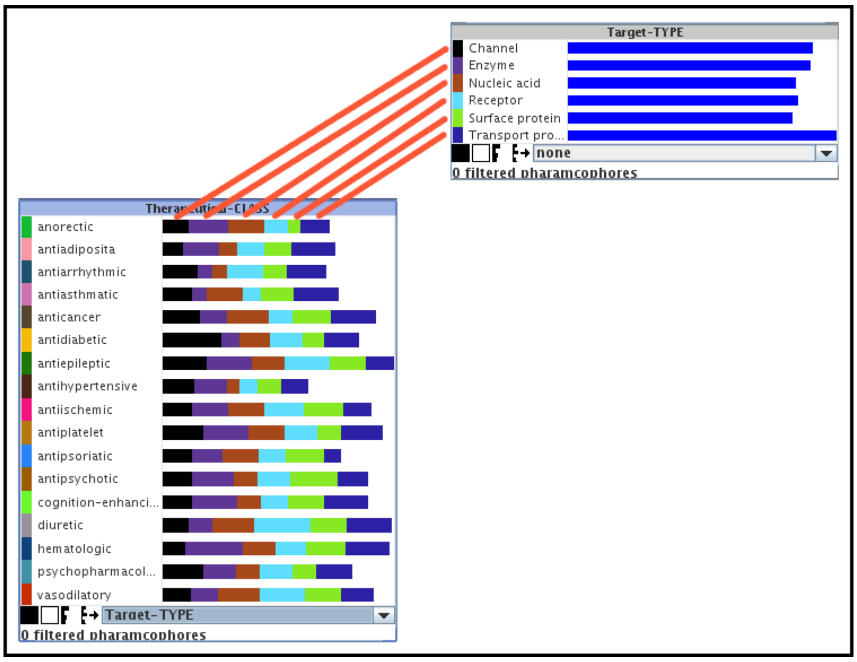


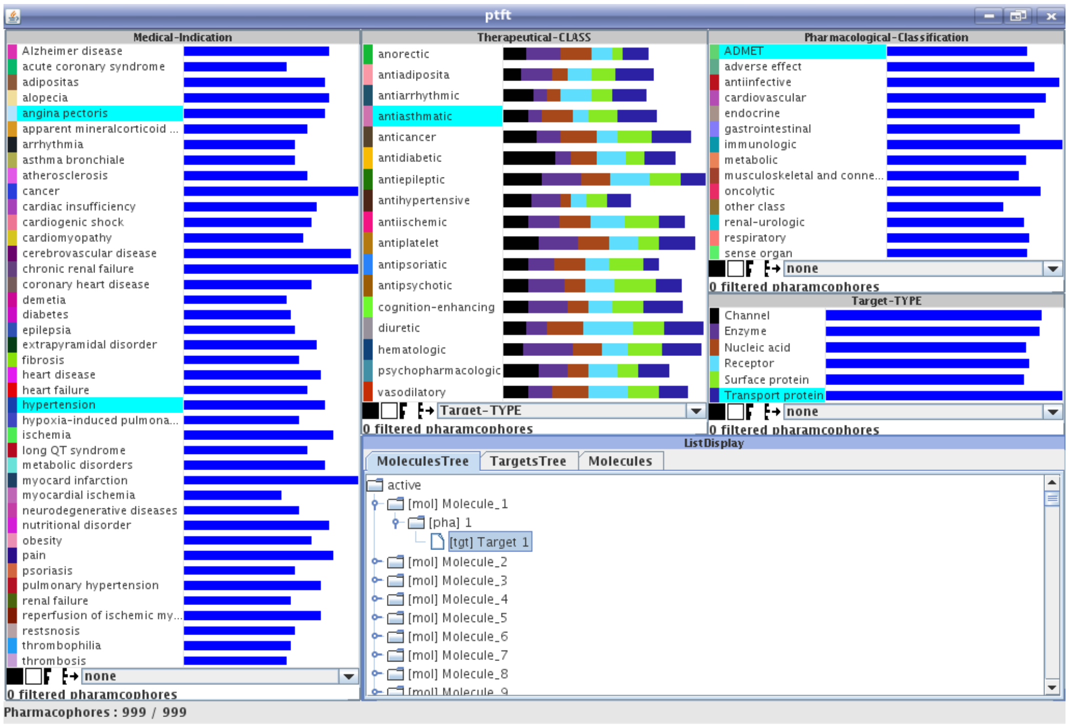
Figure 8.3: Correlation between attributes of *therapeutical class* and *target type*.

8.1.3 Tree Views

The relationship between molecules, pharmacophores and targets is shown in the two tree-views: *MoleculesTree* and *TargetsTree*. The difference between these two views is that the MoleculesTree is based on molecules and shows hierarchically for each molecule pharmacophores and targets which are hit by this molecule. By selecting a target in one of this views the corresponding attributes of the target can be highlighted in the attribute class views (see Figure 8.4).

8.1.4 Detail View

Finally, there is a *detail view* in Ptft, where molecules are listed one below the other. Each row represents one molecule. Details like 2D representation, name, and others are displayed for each molecule (see Figure 8.5).



ptft

Medical-Indication
- Alzheimer disease
- acute coronary syndrome
- adipositas
- alopecia
- angina pectoris
- apparent mineralcorticoid ...
- arrhythmia
- asthma bronchiale
- atherosclerosis
- cancer
- cardiac insufficiency
- cardiogenic shock
- cardiomyopathy
- cerebrovascular disease
- chronic renal failure
- coronary heart disease
- demetia
- diabetes
- epilepsia
- extrapyramidal disorder
- fibrosis
- heart disease
- heart failure
- hypertension
- hypoxia-induced pulmona...
- ischemia
- long QT syndrome
- metabolic disorders
- myocard infarction
- myocardial ischemia
- neurodegenerative diseases
- nutritional disorder
- obesity
- pain
- psoriasis
- pulmonary hypertension
- renal failure
- reperfusion of ischemic my...
- restsnosis
- thrombophilia
- thrombosis

none

0 filtered pharamcophores

Pharmacophores : 999 / 999

Therapeutical-CLASS
- anorectic
- antiadiposita
- antiarrhythmic
- antiasthmatic
- anticancer
- antidiabetic
- antiepileptic
- antihypertensive
- antiischemic
- antiplatelet
- antipsoriatic
- antipsychotic
- cognition-enhancing
- diuretic
- hematologic
- psychopharmacologic
- vasodilatory

Target-TYPE

0 filtered pharamcophores

Pharmacological-Classification
- ADMET
- adverse effect
- antiinfective
- cardiovascular
- endocrine
- gastrointestinal
- immunologic
- metabolic
- musculoskeletal and conne...
- oncolytic
- other class
- renal-urologic
- respiratory
- sense organ

none

0 filtered pharamcophores

Target-TYPE
- Channel
- Enzyme
- Nucleic acid
- Receptor
- Surface protein
- Transport protein

none

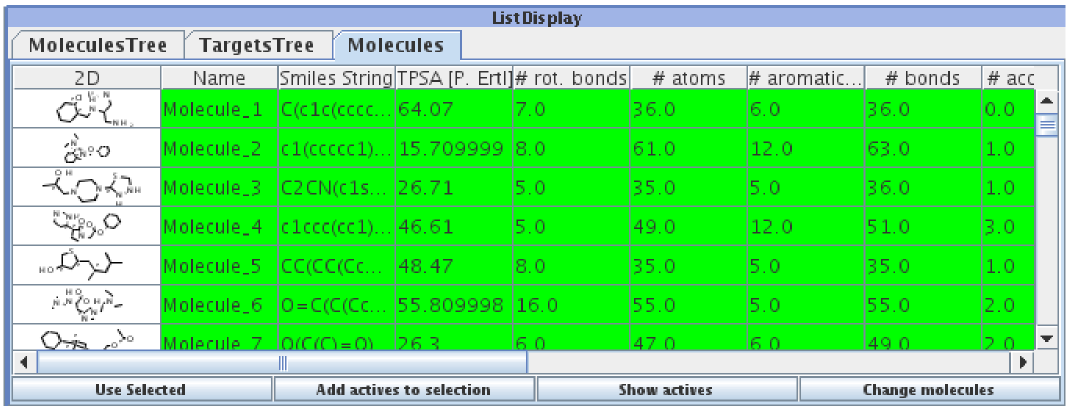
0 filtered pharamcophores

ListDisplay

MoleculesTree | TargetsTree | Molecules

```
active
  [mol] Molecule_1
    [pha] 1
      [tgt] Target 1
  [mol] Molecule_2
  [mol] Molecule_3
  [mol] Molecule_4
  [mol] Molecule_5
  [mol] Molecule_6
  [mol] Molecule_7
  [mol] Molecule_8
  [mol] Molecule_9
```

Figure 8.4: *Target 1* is selected in the *MoleculesTree* view. The Attributes of *Target 1* are highlighted in the attribute class views.



ListDisplay

MoleculesTree | TargetsTree | Molecules

2D	Name	Smiles String	TPSA [P. Ertl]	# rot. bonds	# atoms	# aromatic...	# bonds	# acc
	Molecule_1	C(c1c(cccc...	64.07	7.0	36.0	6.0	36.0	0.0
	Molecule_2	c1(ccccc1)...	15.709999	8.0	61.0	12.0	63.0	1.0
	Molecule_3	C2CN(c1s...	26.71	5.0	35.0	5.0	36.0	1.0
	Molecule_4	c1ccc(cc1)...	46.61	5.0	49.0	12.0	51.0	3.0
	Molecule_5	CC(CC(Cc...	48.47	8.0	35.0	5.0	35.0	1.0
	Molecule_6	O=C(C(Cc...	55.809998	16.0	55.0	5.0	55.0	2.0
	Molecule_7	O(C(O)=O)...	26.3	6.0	47.0	6.0	49.0	2.0

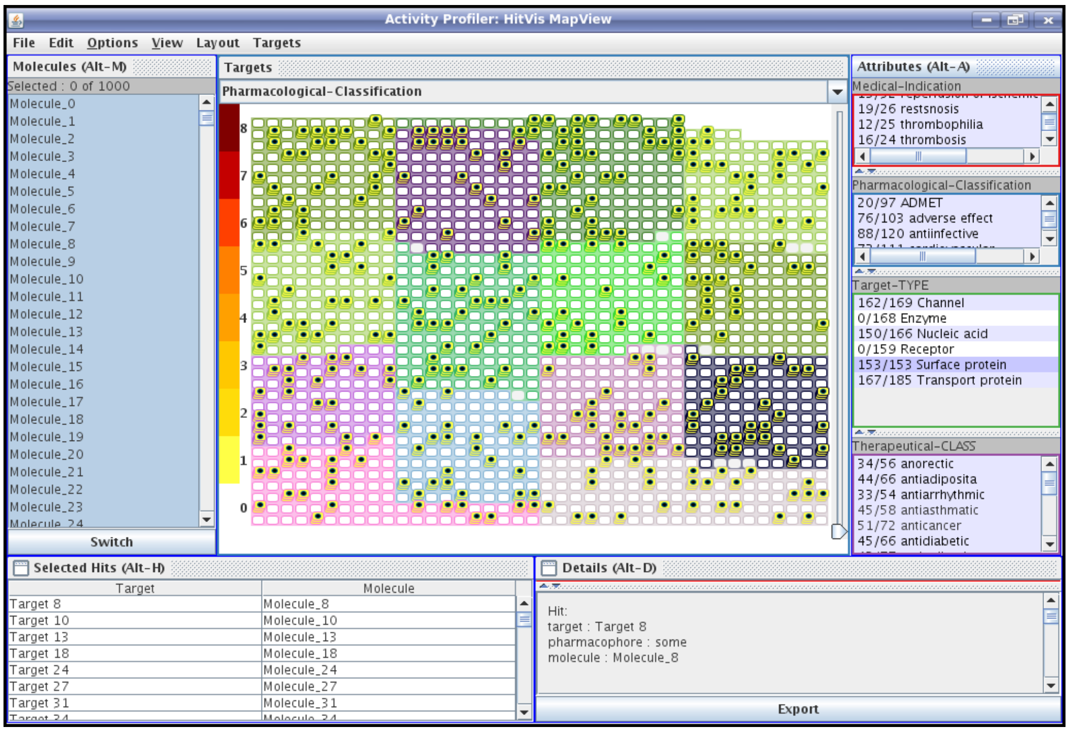
Use Selected | Add actives to selection | Show actives | Change molecules

Figure 8.5: Detail view of Ptft showing the 2D represention and attributes for each molecule.

8.2 HitVis - Hit Visualization

8.2.1 Overview

Inte:ligands second generation application for the visualization of n:m drug target interactions is Hitvis. In Figure 8.6 an overview of Hitvis is shown. Just as in Ptft,



Selected : 0 of 1000
Molecule_0
Molecule_1
Molecule_2
Molecule_3
Molecule_4
Molecule_5
Molecule_6
Molecule_7
Molecule_8
Molecule_9
Molecule_10
Molecule_11
Molecule_12
Molecule_13
Molecule_14
Molecule_15
Molecule_16
Molecule_17
Molecule_18
Molecule_19
Molecule_20
Molecule_21
Molecule_22
Molecule_23

Medical-Indication
19/26 restsnosis
12/25 thrombophilia
16/24 thrombosis

Pharmacological-Classification
20/97 ADMET
76/103 adverse effect
88/120 antiinfective

Target-TYPE
162/169 Channel
0/168 Enzyme
150/166 Nucleic acid
0/159 Receptor
153/153 Surface protein
167/185 Transport protein

Therapeutical-CLASS
34/56 anorectic
44/66 antiadiposita
33/54 antiarrhythmic
45/58 antiasthmatic
51/72 anticancer
45/66 antidiabetic

Selected Hits (Alt-H)

Target	Molecule
Target 8	Molecule_8
Target 10	Molecule_10
Target 13	Molecule_13
Target 18	Molecule_18
Target 24	Molecule_24
Target 27	Molecule_27
Target 31	Molecule_31

Details (Alt-D)

Hit:
target : Target 8
pharmacophore : some
molecule : Molecule_8

Figure 8.6: An overview of Inte:ligands second generation application for visualizing parallel screening results called HitVis.

the screening has to be done statically. The results are loaded from a file within the program, and the Ptft data structures were reused.

There are again multiple views for filtering, showing activity profiles and viewing details of drug candidates. These views are linked with each other, so actions performed in one view are automatically applied to all views. In the following each of the views is shortly summarized.

8.2.2 Target View

The target view (s. Figure 8.7) consists of a heat map including height information, so it is a mix of a heat- and height map. Each field in the map represents a target

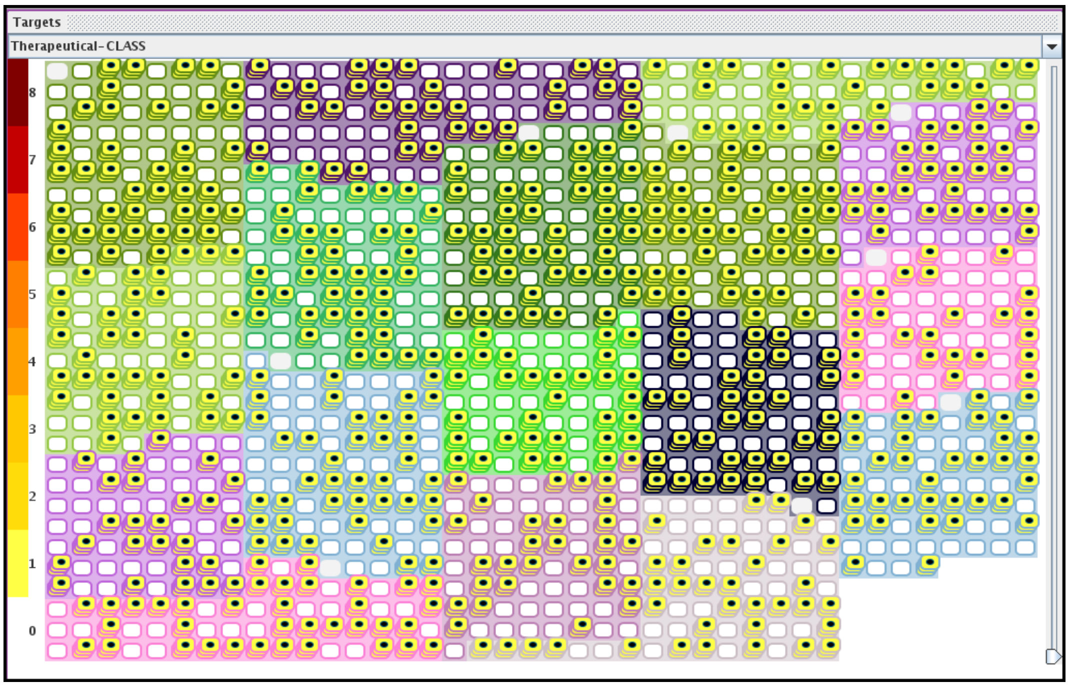


Figure 8.7: An overview of the *Target View* of Hitvis.

and is clustered by the attribute of the target. The attributes are split into separate classes of attributes just as in Ptft. Only attributes of one class are displayed in the target view, but the class to be shown can be changed by the combobox at the top of the view. To allow graphical clustering, which means arranging fields of the same attribute side by side, the position of each field is calculated at runtime.

The height of a field shows the total amount of hits of the corresponding target relative to the hit amount of other targets. The same information is encoded by the color of the field (white for no hits and yellow to red for one or more hits). The color of the field should not be mixed up with the color of the cluster grid, which represents a target attribute (see Figure 8.8).

Another feature of this view is that targets are highlighted when moving with the mouse over the view. Figure 8.9 shows an example, where three fields are highlighted at the same time. All of them are representing the same target, having three different attributes of the specified attribute class. The name of the target can be readout from the statusbar at the bottom of the view.

In addition to the target name, the target view of HitVis allows for showing or hiding names of attributes over the corresponding cluster by moving the slider on the very right of the view up or down.

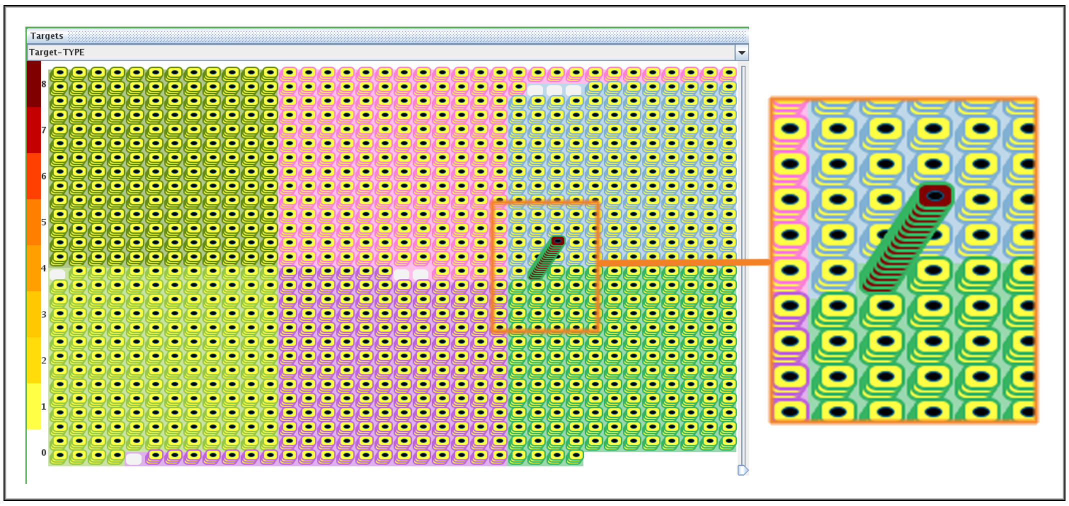
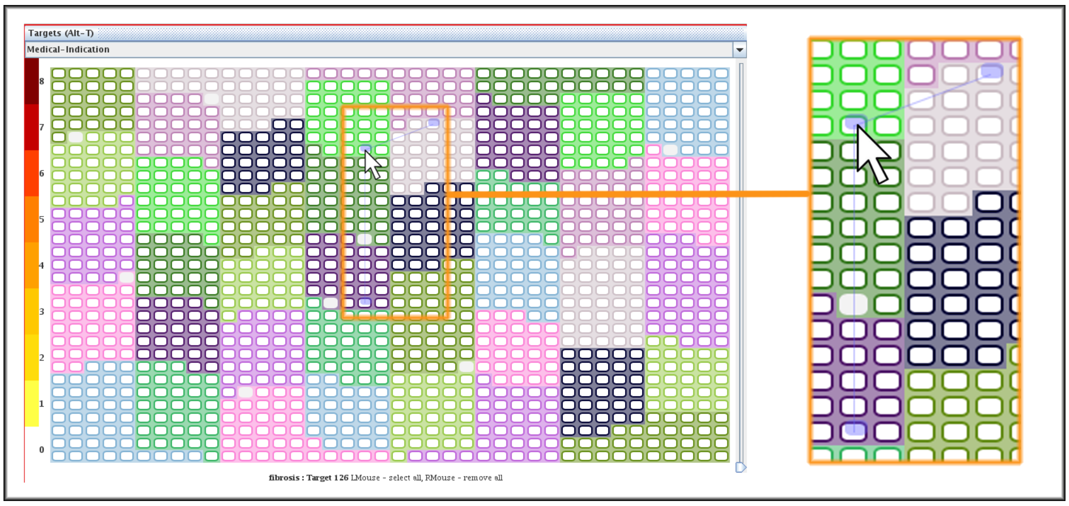


Figure 8.8: An example showing that the hits amount is encoded by both, heat- and height information. White fields for targets with no hits, yellow for targets with one hit and dark red for targets with eight hits.



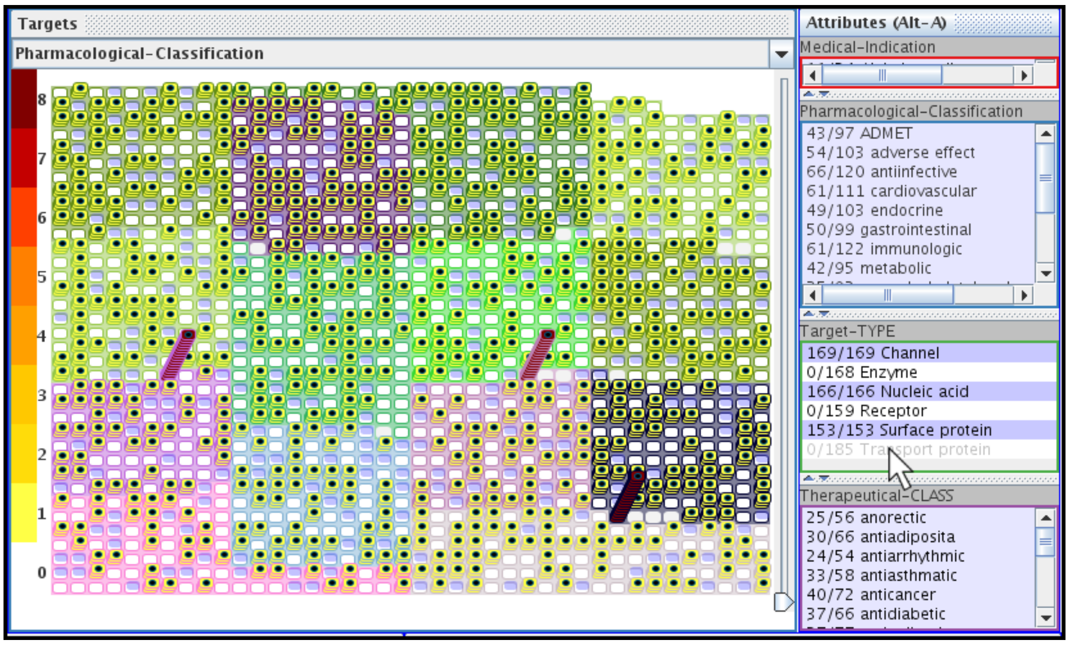
fibrosis : Target 126 LMouse - select all, RMouse - remove all

Figure 8.9: Targets are highlighted when moved with the mouse over the field. In this example *Target 126* is highlighted three times, because the target has three different attributes of medical indication.

8.2.3 Attribute View

The attribute view of Hitvis is similar to the attribute class views of Ptft if we imagine that all of them are put together in one view one below the other. Each row represents one attribute just as in Ptft. On the right the name of the attribute and on the left the total amount and the amount of unfiltered hits of targets having this attribute is displayed. The amount of unfiltered hits conforms to the bar width in Ptft. Technically each row is a tri-state button. In the first state hits for targets with this attribute are partially filtered by other constraints set in the program. The second state unfilters these hits and the third state filters all hits of the attribute.

Furthermore, the attribute view is linked to the target view. Therefore filtering in the attribute view is automatically applied to the target view. Another example for the linking is that moving the mouse over the attribute view results in highlighting of targets in the target view having the corresponding attribute. This is illustrated in Figure 8.10.



43/97 ADMET
54/103 adverse effect
66/120 antiinfective
61/111 cardiovascular
49/103 endocrine
50/99 gastrointestinal
61/122 immunologic
42/95 metabolic

169/169 Channel
0/168 Enzyme
166/166 Nucleic acid
0/159 Receptor
153/153 Surface protein
0/185 Transport protein

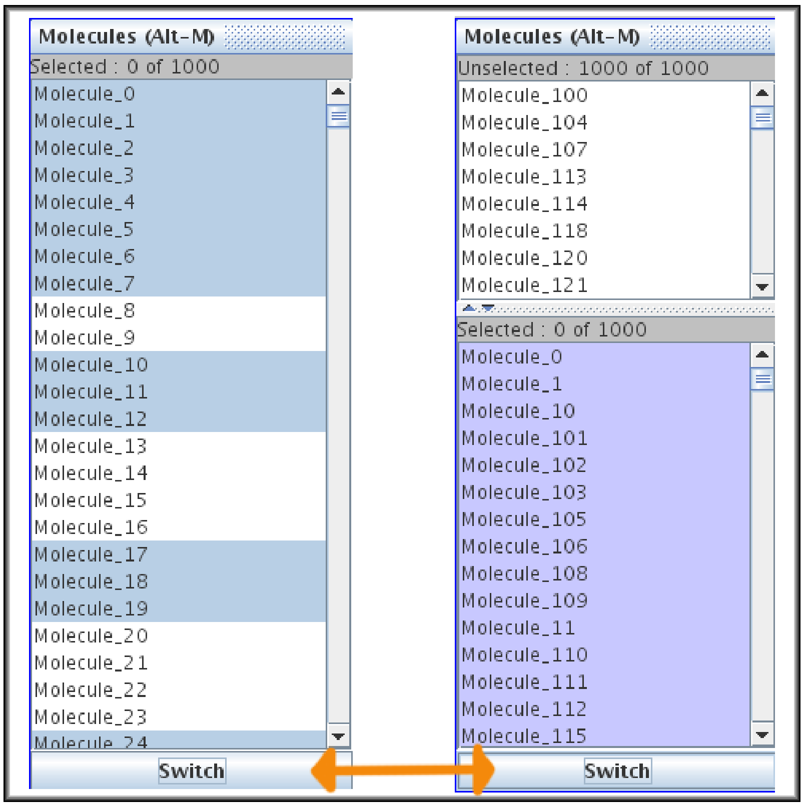
25/56 anorectic
30/66 antiadiposita
24/54 antiarrhythmic
33/58 antiasthmatic
40/72 anticancer
37/66 antidiabetic

Figure 8.10: In this example the cursor is over the target type *Transport protein* in the attributes view. Therefore, all targets of the target type *Transport protein* are highlighted in the targets view.

8.2.4 Molecule View

The molecule view lists all molecules used for screening one below the other. By selecting a molecule hits corresponding to the molecule become either filtered if they were unfiltered before or vice versa.

The *Switch* button on the bottom of the view can be used to split the view into two parts to separate filtered and unfiltered molecules (s. Figure 8.11).



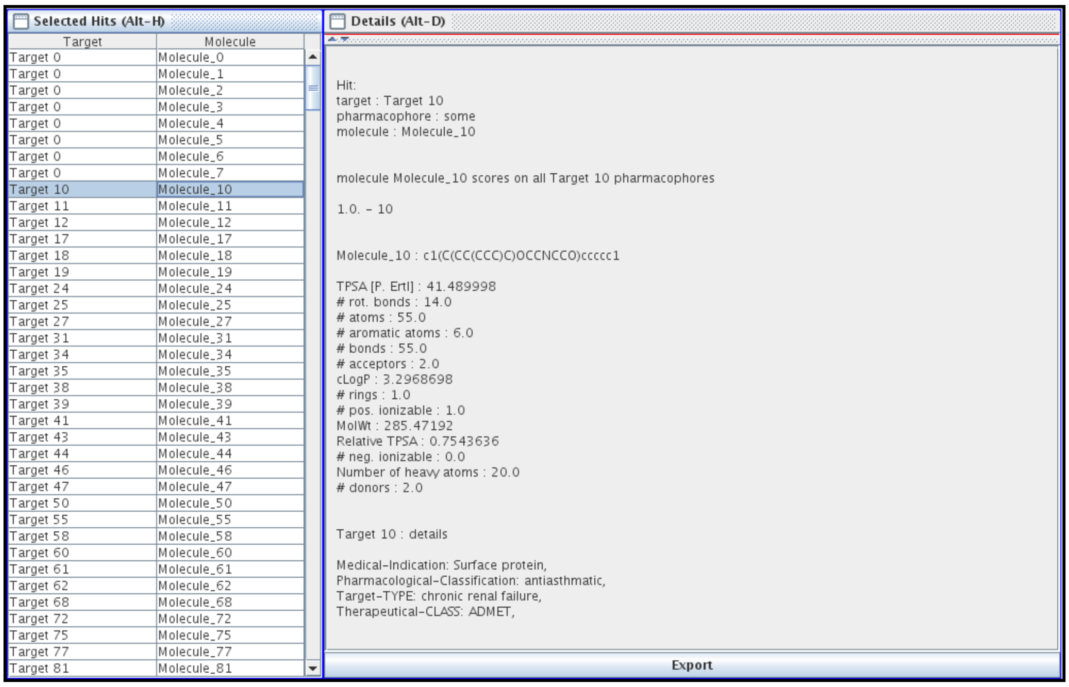
Molecules (Alt-M)
Selected : 0 of 1000
Molecule_0
Molecule_1
Molecule_2
Molecule_3
Molecule_4
Molecule_5
Molecule_6
Molecule_7
Molecule_8
Molecule_9
Molecule_10
Molecule_11
Molecule_12
Molecule_13
Molecule_14
Molecule_15
Molecule_16
Molecule_17
Molecule_18
Molecule_19
Molecule_20
Molecule_21
Molecule_22
Molecule_23
Molecule_24
Switch

Molecules (Alt-M)
Unselected : 1000 of 1000
Molecule_100
Molecule_104
Molecule_107
Molecule_113
Molecule_114
Molecule_118
Molecule_120
Molecule_121
Selected : 0 of 1000
Molecule_0
Molecule_1
Molecule_10
Molecule_101
Molecule_102
Molecule_103
Molecule_105
Molecule_106
Molecule_108
Molecule_109
Molecule_11
Molecule_110
Molecule_111
Molecule_112
Molecule_115
Switch

Figure 8.11: On the left unfiltered molecules are blue and filtered molecules white. On the right we have the situation after pressing the switch button, where filtered and unfiltered molecules are separated.

8.2.5 Selected Hits and Details

Finally, there are the selected hits view and the detail view. The selected hits view lists all unfiltered hits one below the other and is automatically updated if any filters are changed. This view conforms to the tree views in Ptft. By selecting a row, which represents a hit, details for the corresponding target and molecule are displayed in the details view. The details view is a pure textual view and offers an export functionality of the displayed information. Figure 8.12 depicts both views.



Target	Molecule
Target 0	Molecule_0
Target 0	Molecule_1
Target 0	Molecule_2
Target 0	Molecule_3
Target 0	Molecule_4
Target 0	Molecule_5
Target 0	Molecule_6
Target 0	Molecule_7
Target 10	Molecule_10
Target 11	Molecule_11
Target 12	Molecule_12
Target 17	Molecule_17
Target 18	Molecule_18
Target 19	Molecule_19
Target 24	Molecule_24
Target 25	Molecule_25
Target 27	Molecule_27
Target 31	Molecule_31
Target 34	Molecule_34
Target 35	Molecule_35
Target 38	Molecule_38
Target 39	Molecule_39
Target 41	Molecule_41
Target 43	Molecule_43
Target 44	Molecule_44
Target 46	Molecule_46
Target 47	Molecule_47
Target 50	Molecule_50
Target 55	Molecule_55
Target 58	Molecule_58
Target 60	Molecule_60
Target 61	Molecule_61
Target 62	Molecule_62
Target 68	Molecule_68
Target 72	Molecule_72
Target 75	Molecule_75
Target 77	Molecule_77
Target 81	Molecule_81

```
Hit:
target : Target 10
pharmacophore : some
molecule : Molecule_10


molecule Molecule_10 scores on all Target 10 pharmacophores

1.0. - 10


Molecule_10 : c1(C(CC(CCC)C)OCCNCCO)ccccc1

TPSA [P. Ertl] : 41.489998
# rot. bonds : 14.0
# atoms : 55.0
# aromatic atoms : 6.0
# bonds : 55.0
# acceptors : 2.0
cLogP : 3.2968698
# rings : 1.0
# pos. ionizable : 1.0
MolWt : 285.47192
Relative TPSA : 0.7543636
# neg. ionizable : 0.0
Number of heavy atoms : 20.0
# donors : 2.0


Target 10 : details

Medical-Indication: Surface protein,
Pharmacological-Classification: antiasthmatic,
Target-TYPE: chronic renal failure,
Therapeutical-CLASS: ADMET,
```

Figure 8.12: The hit where *Molecule_10* fits to *Target_10* is selected in the selected hits view. The details view shows the details of both, *Molecule_10* and *Target_10*.

8.3 Evaluation

Together with the Institute of Pharmacy of the Leopold-Franzens-University of Innsbruck the pros and cons of both applications, Ptft and Hitvis have been discussed and analyzed. The meeting took about five hours and was made up of two parts. In the first part the two applications were introduced, and in the second part they were evaluated. The goal was on the one hand to extract and combine the advantages of each approach and on the other hand to discuss the shortcomings and problems of them and to find solutions. A total of eleven people took part of the moderated meeting. Ten of them are chemical experts and do research in this area. In the following the results of the evaluation of Ptft and HitVis are described.

8.3.1 Ptft - Pharmacophore and Target Filtering Tool

Pros

- The representation of the amount of hits by bars allows an easy comparison.
- The bar visualization is simple and intuitive.
- The partition of the bars allows the user to see correlations between two attribute classes.
- The 2D representation of the molecule in the detail view gives the user information about the structure of the drug candidate.

Cons

- The attributes are sorted by name, but there is no sorting of the attributes by amount of hits.
- When changing the size of an attribute class view, the height of each row changed, which often results in too great or too small rows. Constant heights with a scroll possibility would allow for better quantitative estimation.
- Hiding attributes with a hits amount lower than a minimum set by the user is not supported.
- The tree views are not linked with the detail view.
- The tree views and the detail view are put into a tabbed pane. Therefore, they can only be accessed sequentially and not all at once, which makes this part of the program circuitous.
- Each view is put into a separate inner frame. This makes the resizing, moving and rechanging the position of views very flexible. But the analyses have shown that this feature is not needed by the user. The need to resize adjacent views when resizing a particular view makes this method very circuitous.

8.3.2 HitVis - Hit Visualization

Pros

- The heat map is very powerful, showing the amount of hits for each target through heat and height information, the amount of hits for each attribute via the size of the clusters, and, furthermore, allows interaction for filtering.
- Different layouts, which can be accessed through the menu or via keyboard shortcuts allow the user to hide views which are not of interest and at the same time there is more place for the views of interest.

Cons

- If too many targets and hits are displayed the performance of Hitvis becomes bad.
- The correlation between attributes of different attribute classes is missing.
- The amount of hits of each target is numerically displayed and not graphically, which makes it difficult to compare them.
- Too much information like attributes, hits amount and highlighting is encoded by color. Because of this overloading of colors extracting the right information takes more time.
- The sorting of the attributes by the hits amount is missing.
- The hiding of attributes in the attributes view is completely missing.
- In connection with highlighting targets, the attribute view is linked to the target view only in one direction.
- Highlighting targets by moving the mouse in the molecules view is not supported.
- Just as in Ptft, the use of inner frames is very circuitous.

8.3.3 Conclusions

1. The analysis and discussion has shown that the heat map with height information is a very powerful tool for such a problem. A drawback of the heat map in Hitvis is the performance. The application gets very slow when the amount of items to display increases. The main reason for the performance problems is the use of Java2D, which is not hardware accelerated. Using OpenGL for the map will dramatically increase the performance and enable the use of 3D graphics, which can be very interesting in connection with the height information of the map.

2. To solve the problems arisen from encoding too much information with colors, the clusters should be spatially separated from each other and in addition, the heat information should be encoded by the intensity of the attribute color.

3. The discussion also showed that there is no need for the rounded rectangles within the fields. Removing them will result in less visual objects, which should help the user to concentrate on the ones that are remaining.

4. The visualization of attributes in Ptft is the one which should be preferred. The numerical representation of the hits amount in Hitvis can be used as an additional information, but the bars used in Ptft allow an easier comparison and have the bonus of showing the correlation to attributes of another class.

5. Problems existing in both of the applications are that sorting of lists or tables and that hiding of filtered elements are not supported. These problems should be addressed to allow the user to have a better overview of the relevant data.

6. Working with split panes is much faster than with inner frames.

7. Finally, the layout concept of Hitvis, where the user can hide views, which are not of interest, is a powerful mechanism to overcome the lack of display space.

Chapter 9

Apt (Activity Profiling Tool)

In the prior chapters state of the art techniques and tools for the visualization of n:m drug target interactions were described and analyzed. Together with the Institute of Pharmacy of the Leopold-Franzens-University of Innsbruck the tools Ptft and Hitvis were analyzed and improvement potentials were discussed. The third generation activity profiling tool of Inte:ligand is developed as part of this work and is described in this chapter.

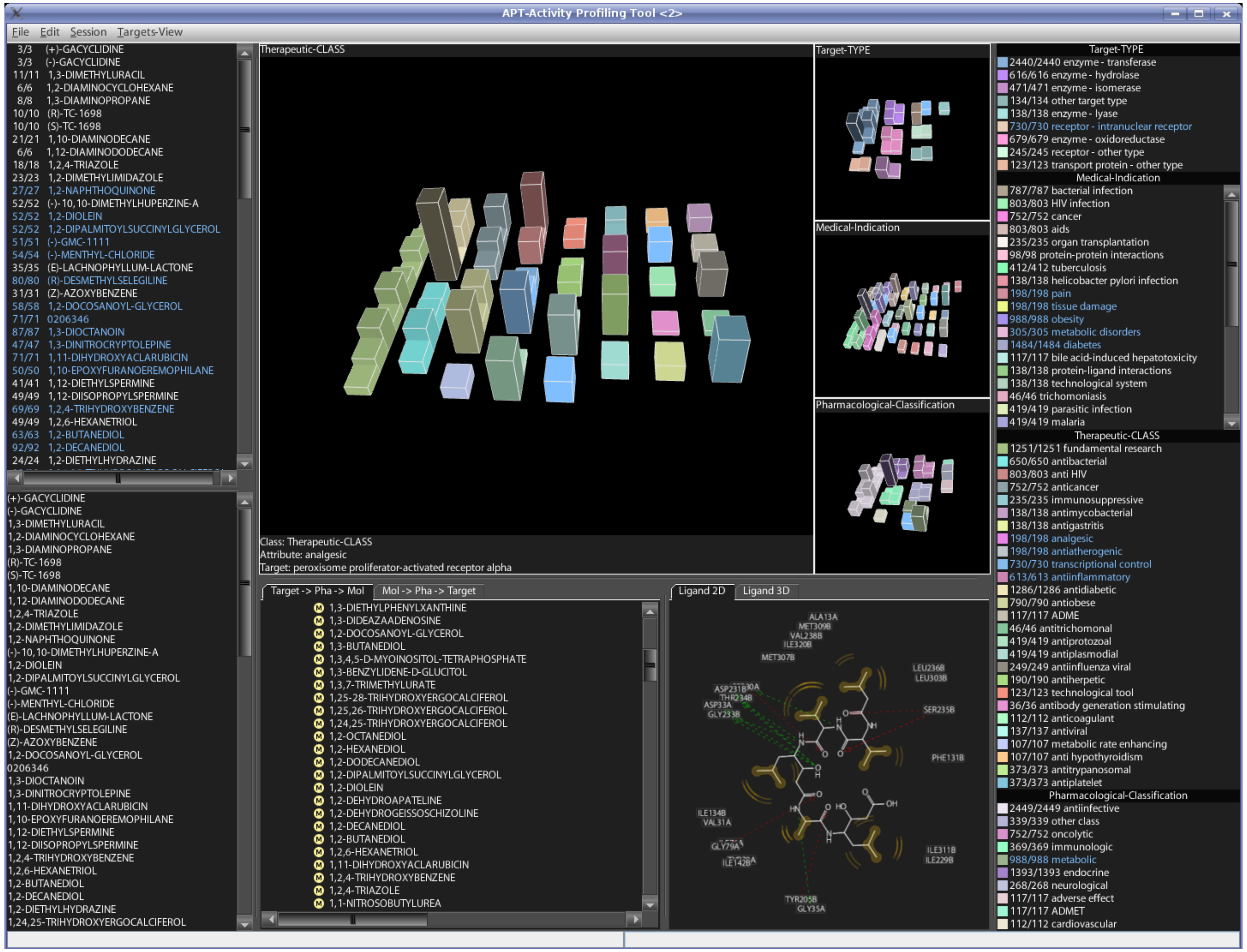
9.1 Overview

Apt is written in Java SE 5 [1] and is not compatible to older versions, because of the use of generic types. Since some imports are reorganized in Java SE 6 [2], Apt is also incompatible to Java SE 6, but is prepared for upgrade to Java SE 6.

In the following an overview of the main features of Apt (see Figure 9.1) is provided.

9.1.1 Screening

A drawback of Ptft and Hitvis is that screening itself is not supported. The user has to screen in a separate application and save the results in a specific format, which can be used by Hitvis and Ptft to load the results. The reason for this is on the one hand that screening thousands of molecules against hundreds of targets can take a very long time. On the other hand at that time when these two programs were developed the technical infrastructure, like pharmacophore databases and matching libraries were not as sophisticated as they are today.



3/3	(+)-GACYCLIDINE
3/3	(-)-GACYCLIDINE
11/11	1,3-DIMETHYLURACIL
6/6	1,2-DIAMINOCYCLOHEXANE
8/8	1,3-DIAMINOPROPANE
10/10	(R)-TC-1698
10/10	(S)-TC-1698
21/21	1,10-DIAMINODECANE
6/6	1,12-DIAMINODODECANE
18/18	1,2,4-TRIAZOLE
23/23	1,2-DIMETHYLIMIDAZOLE
27/27	1,2-NAPHTHOQUINONE
52/52	(-)-10,10-DIMETHYLHUPERZINE-A
52/52	1,2-DIOLEIN
52/52	1,2-DIPALMITOYLSUCCINYLGLYCEROL
51/51	(-)-GMC-1111
54/54	(-)-MENTHYL-CHLORIDE
35/35	(E)-LACHNOPHYLLUM-LACTONE
80/80	(R)-DESMETHYLSELEGILINE
31/31	(Z)-AZOXYBENZENE
58/58	1,2-DOCOSANOYL-GLYCEROL
71/71	0206346
87/87	1,3-DIOCTANOIN
47/47	1,3-DINITROCRYPTOLEPINE
71/71	1,11-DIHYDROXYACLARUBICIN
50/50	1,10-EPOXYFURANOEREMOPHILANE
41/41	1,12-DIETHYLSPERMINE
49/49	1,12-DIISOPROPYLSPERMINE
69/69	1,2,4-TRIHYDROXYBENZENE
49/49	1,2,6-HEXANETRIOL
63/63	1,2-BUTANEDIOL
92/92	1,2-DECANEDIOL
24/24	1,2-DIETHYLHYDRAZINE

(+)-GACYCLIDINE
(-)-GACYCLIDINE
1,3-DIMETHYLURACIL
1,2-DIAMINOCYCLOHEXANE
1,3-DIAMINOPROPANE
(R)-TC-1698
(S)-TC-1698
1,10-DIAMINODECANE
1,12-DIAMINODODECANE
1,2,4-TRIAZOLE
1,2-DIMETHYLIMIDAZOLE
1,2-NAPHTHOQUINONE
(-)-10,10-DIMETHYLHUPERZINE-A
1,2-DIOLEIN
1,2-DIPALMITOYLSUCCINYLGLYCEROL
(-)-GMC-1111
(-)-MENTHYL-CHLORIDE
(E)-LACHNOPHYLLUM-LACTONE
(R)-DESMETHYLSELEGILINE
(Z)-AZOXYBENZENE
1,2-DOCOSANOYL-GLYCEROL
0206346
1,3-DIOCTANOIN
1,3-DINITROCRYPTOLEPINE
1,11-DIHYDROXYACLARUBICIN
1,10-EPOXYFURANOEREMOPHILANE
1,12-DIETHYLSPERMINE
1,12-DIISOPROPYLSPERMINE
1,2,4-TRIHYDROXYBENZENE
1,2,6-HEXANETRIOL
1,2-BUTANEDIOL
1,2-DECANEDIOL
1,2-DIETHYLHYDRAZINE
1,24,25-TRIHYDROXYERGOCALCIFEROL

Class: Therapeutic-CLASS
Attribute: analgesic
Target: peroxisome proliferator-activated receptor alpha

1,3-DIETHYLPHENYLXANTHINE
1,3-DIDEAZAADENOSINE
1,2-DOCOSANOYL-GLYCEROL
1,3-BUTANEDIOL
1,3,4,5-D-MYOINOSITOL-TETRAPHOSPHATE
1,3-BENZYLIDENE-D-GLUCITOL
1,3,7-TRIMETHYLURATE
1,25-28-TRIHYDROXYERGOCALCIFEROL
1,25,26-TRIHYDROXYERGOCALCIFEROL
1,24,25-TRIHYDROXYERGOCALCIFEROL
1,2-OCTANEDIOL
1,2-HEXANEDIOL
1,2-DODECANEDIOL
1,2-DIPALMITOYLSUCCINYLGLYCEROL
1,2-DIOLEIN
1,2-DEHYDROAPATELINE
1,2-DEHYDROGEISSOSCHIZOLINE
1,2-DECANEDIOL
1,2-BUTANEDIOL
1,2,6-HEXANETRIOL
1,11-DIHYDROXYACLARUBICIN
1,2,4-TRIHYDROXYBENZENE
1,2,4-TRIAZOLE
1,1-NITROSOBUTYLUREA

Target-TYPE

2440/2440	enzyme - transferase
616/616	enzyme - hydrolase
471/471	enzyme - isomerase
134/134	other target type
138/138	enzyme - lyase
730/730	receptor - intranuclear receptor
679/679	enzyme - oxidoreductase
245/245	receptor - other type
123/123	transport protein - other type

Medical-Indication

787/787	bacterial infection
803/803	HIV infection
752/752	cancer
803/803	aids
235/235	organ transplantation
98/98	protein-protein interactions
412/412	tuberculosis
138/138	helicobacter pylori infection
198/198	pain
198/198	tissue damage
988/988	obesity
305/305	metabolic disorders
1484/1484	diabetes
117/117	bile acid-induced hepatotoxicity
138/138	protein-ligand interactions
138/138	technological system
46/46	trichomoniasis
419/419	parasitic infection
419/419	malaria

Therapeutic-CLASS

1251/1251	fundamental research
650/650	antibacterial
803/803	anti HIV
752/752	anticancer
235/235	immunosuppressive
138/138	antimycobacterial
138/138	antigastritis
198/198	analgesic
198/198	antiatherogenic
730/730	transcriptional control
613/613	antiinflammatory
1286/1286	antidiabetic
790/790	antiobese
117/117	ADME
46/46	antitrichomonal
419/419	antiprotozoal
419/419	antiplasmodial
249/249	antiinfluenza viral
190/190	antiherpetic
123/123	technological tool
36/36	antibody generation stimulating
112/112	anticoagulant
137/137	antiviral
107/107	metabolic rate enhancing
107/107	anti hypothyroidism
373/373	antitrypanosomal
373/373	antiplatelet

Pharmacological-Classification

2449/2449	antiinfective
339/339	other class
752/752	oncolytic
369/369	immunologic
988/988	metabolic
1393/1393	endocrine
268/268	neurological
117/117	adverse effect
117/117	ADMET
112/112	cardiovascular

Figure 9.1: An overview of Apt (Activity Profiling Tool).

However, Apt supports screening in a dynamic way. The user can choose molecule libraries and target sets for screening. The screening is performed in the background and the GUI is updated whenever new incoming results are available. The status of the overall screening process is shown by a progress bar, which is updated regularly. Furthermore, new incoming results are highlighted to inform the user about the updates. Since the overall screening of huge libraries can take several days or weeks Apt supports pausing a screening session. The user can continue the session whenever he or she wants. Another feature of Apt in connection with screening is that already finished results can be loaded at any time.

9.1.2 Multiple Views

Just as in Ptft and Hitvis the concept of multiple views combined with linking and brushing is used for a user-effective interactive visualization and filtering of screening results. An important improvement is that more views are linked with each other and that the highlighting and filtering of data is much more consistent in Apt. Multiple split panes are used instead of inner frames to divide the views, which allows for a very quick resizing of the views. The arrangement of the views can be defined by the user making the layout of Apt very flexible. Almost all views can be displayed at once, only detail views and tree views are tabbed and can be accessed sequentially.

9.1.3 Highlighting and Filtering

In Apt there are three views that support filtering: the target view, the attribute view, and the molecule view. The filtering is done based on hits, which means selecting a molecule, a target or an target attribute for filtering results in filtering all hits corresponding to the selected element. Moving the mouse over an element of one of these views highlights all elements, which would be affected if that element is selected for filtering. The same applies for the unfiltering of hits.

9.1.4 Target View

For the visualization of the activity profiles a heat map with height information, similar to the one in Hitvis, is used. A drawback of the one in Hitvis is that the use of Java2D slows down the application, if too many targets and hits have to be displayed on the map. The heat map in Apt is also written in Java, but by using JOGL (Java bindings for OpenGL) [41, 65, 6, 20, 32, 37, 47, 48, 43] Apt takes full advantage of the hardware acceleration to dramatically increase the performance of the application. Furthermore OpenGLs 3D API allows a perspective representation of the map, which helps the user examining the height information on the map. Moreover the target view in Apt consists of two parts. One showing the map of the attribute class selected by the user and allowing user interaction and the other one displaying an overview of the other maps. The layout of the map and the representation of the fields has also changed in Apt. Free space is left between the clusters to clearly separate them from each other and the heat information of the fields are now shown by the intensity of the grid color.

9.1.5 Attribute View

The attribute view shows target attributes separated into different classes. The name of the attribute and numerical information about hits from targets having this attribute are displayed in a tabular form. Furthermore the attribute view allows for filtering and highlighting hits and is linked with the target view and molecule view in both directions.

9.1.6 Molecule View

The molecule view is separated into two parts. One displaying all molecules and information about the amount of hits in which they are occurring and the other part which is a list of molecules remaining after the filtering (*"result list"*). Both parts are linked to the filtering mechanism of Apt, but only the first part allows user interaction and highlighting of hits.

9.1.7 Tree Views

Apt offers two tree views placed in a tabbed pane. Both show the relationship between molecules, pharmacophores and targets of unfiltered hits. One with molecules as top level nodes and the other one with targets as top level nodes. The trees are linked to the other views for receiving filtering information to dynamically update the trees when constraints have been added or removed. Selecting a hit in one of the trees shows the corresponding ligand in the detail views.

9.1.8 Detail Views

There are two detail views in Apt which are separated on different tabs, too. One shows the 2D representation and the other one shows the 3D representation of the molecule of the corresponding hit, that was selected in one of the tree views.

9.2 Screening

This section describes the screening concept of Apt with a special focus on the interfaces allowing an easy exchange of the screening client of Apt.

9.2.1 The Screening Concept

The screening concept of Apt is illustrated in Figure 9.2. An IHitHandler is regis-

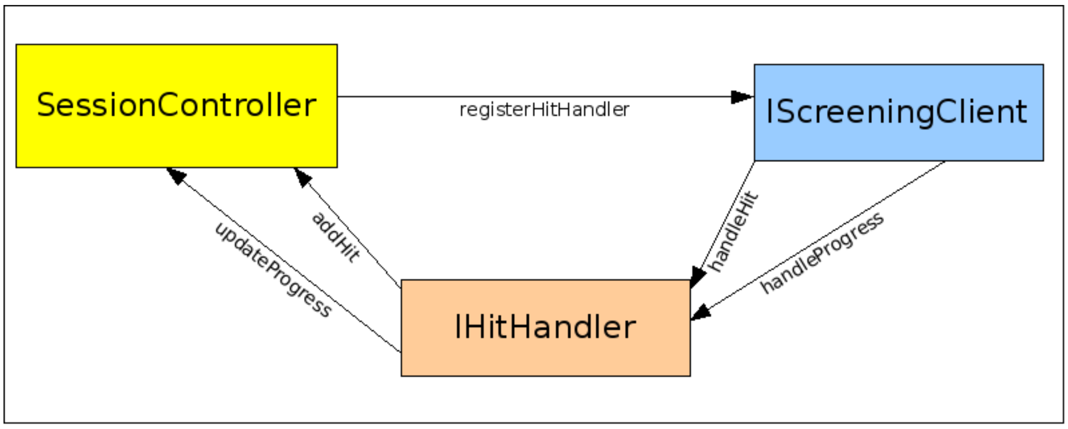


Figure 9.2: Screening concept of Apt.

tered to the IScreeningClient (see Appendix A.1.1). The IScreeningClient sends hits to the handler whenever new displayable hits are available. The IHitHandler (see Appendix A.1.3) uses the SessionController to add the new incoming hits to the system and saves the hits in the molecule file (see table 9.1 on page 62). Furthermore the IScreeningClient regularly sends progress information to the IHitHandler. The handler uses the SessionController to save the new progress status in the session information file and updates the progress bar in the GUI.

The use of the IHitHandler interface in between the SessionController and IScreeningClient allows to easily change the handling of new hits and progress updates. For example, one could save the hits in a different file format or in a database.

The reason for using an interface for the screening client is to allow different client solutions. The default screening client in Apt is the *LocalScreeningClient* (see Appendix A.1.2), where a separate thread is started to perform screening on the local machine.

9.2.2 Peer-To-Peer Screening

A high-performance implementation of the IScreeningClient is provided by Liebig [38]. He makes use of the peer-to-peer framework JXTA to use multiple clients to enhance the screening speed. The so called *JxtaScreeningClient* only works in a LAN and only makes sense, if there are at least three machines available. An schematic overview of the peer-to-peer approach is depicted in Figure 9.3. At the

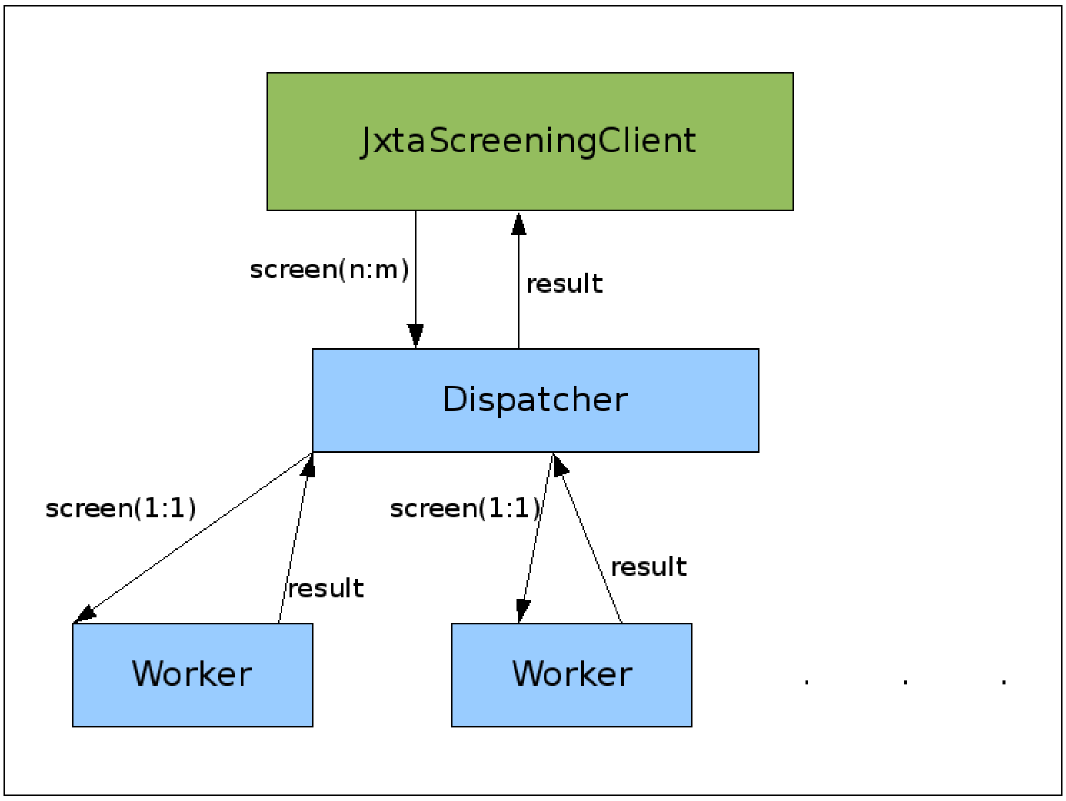


Figure 9.3: A schematic overview of the peer-to-peer screening approach introduced by Liebig [38]. On the top is the peer-to-peer client implementing the IScreeningClient interface of Apt. The blue boxes are representing peers in the LAN.

top there is the JxtaScreeningClient which implements the IScreeningClient interface. After starting a screening session the screening client sends all molecules and pharmacophores to be screened against each other to the *Dispatcher*, which is a JXTA-Peer. The *Dispatcher* splits this work-unit into multiple work-units. In a loop the Dispatcher sends these work-units to *Worker* peers. Whenever a worker peer finishes its work and sends back its result it gets a new work-unit. The Dispatcher queues all results until the screening client polls them. The blue boxes in Figure 9.3 represent JXTA peers. The fact that a Dispatcher is needed and that there is a little overhead of the peer-to-peer framework shows that at least two Worker peers, which means a total of three peers, is needed to be profitable. However, the parallel working of multiple clients can dramatically speed up the overall screening performance of Apt. Another important advantage of this peer-to-peer approach is that the screening process itself is completely decoupled from the GUI, which means that closing Apt after a screening session is started does not affect the screening process at all. The Dispatcher continues queuing screening results, which can be retrieved, for example, when Apt is started next time.

9.2.3 Starting and Stopping a Screening Session

To start a new screening session the user needs to select molecules and targets. The supported file format for molecules is Inte:ligands internal molecule binary format CPB[1], where molecules are listed one below the other and can be accessed by an index. Targets and pharmacophores are stored in a database. The screening information of pharmacophores is stored in PMZ[2] format in the database. To start a screening session the user needs to select a molecule library (CPB-file) and a set of targets. The default target set in Apt is the "All-targets" set which represents all targets stored in the database having pharmacophores with a pmz file. Finally the user has to specify a file for saving the new session. Figure 9.4 shows the GUI for starting a new session. The import and export of all objects in Apt (molecules, targets, pharmacophores,...) is done by the use of interfaces which let developers easily change the file format or add support for other file formats.

To stop a started screening session one has to select "Stop Session" in the session menu.

[1]Compressed Pharmaceutical Binary

[2]Compressed Pharmaceutical Markup Language

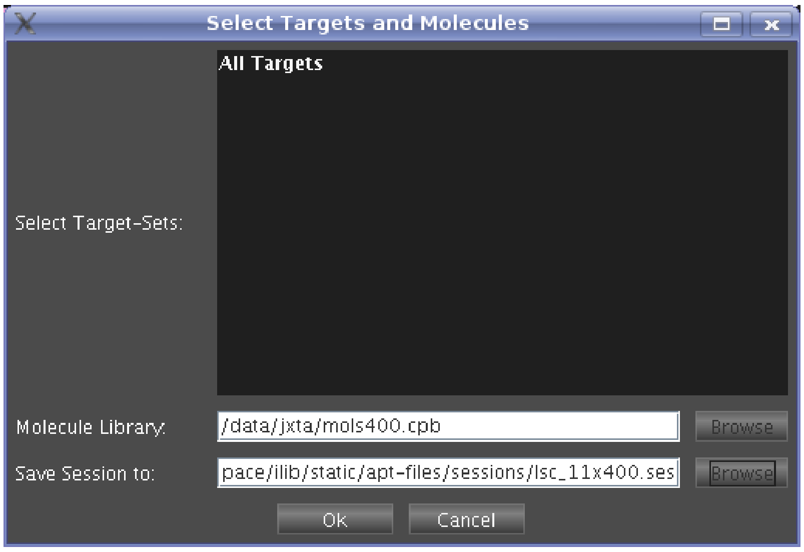


Figure 9.4: The GUI for starting a new screening session.

9.2.4 Saving a Screening Session

After starting a new session Apt saves the session information in files described in table 9.1. The **Session File** is the file, which the user selects, when loading a prior started session. It saves the Session-ID, which is retrieved from the screening client, when starting a session. Furthermore, progress information is kept in the Session File, too. Information about targets, pharmacophores and molecules are saved in separate files. The Session file saves only the filepath to these files.
The **Targets File** saves the ID, name, and attributes of all targets that are selected for the screening session.
Information about pharmacophores is stored in the **Pharmacophores File**. The ID, name and target-IDs of all corresponding targets are provided for each pharmacophore. Finally, molecules are stored in the **Molecule File**. For each molecule the ID, name and all pharmacophore-ID's, the molecule fits to, are saved in the file.
All files are accessed by import, and export interfaces (see Appendix A.3).

9.3 Linking and Brushing in Apt

In Apt there are several views which are linked for highlighting and filtering of hits. Moreover, there are views that are linked for showing molecule and pharmacophore details.

Session File	.ses	Session-ID, progress information, and the filepath to the Target-, Pharmacophore- and Molecule File is specified.
Target File	.ata	IDs, names, and attributes of all targets are listed one below the other.
Pharmacophore File	.pha	IDs, names, and the IDs of the corresponding targets for all pharmacophores are listed one below the other.
Molecule File	.aml	IDs, names, and matched pharmacophores for all molecules are specified.

Table 9.1: Session files used by Apt.

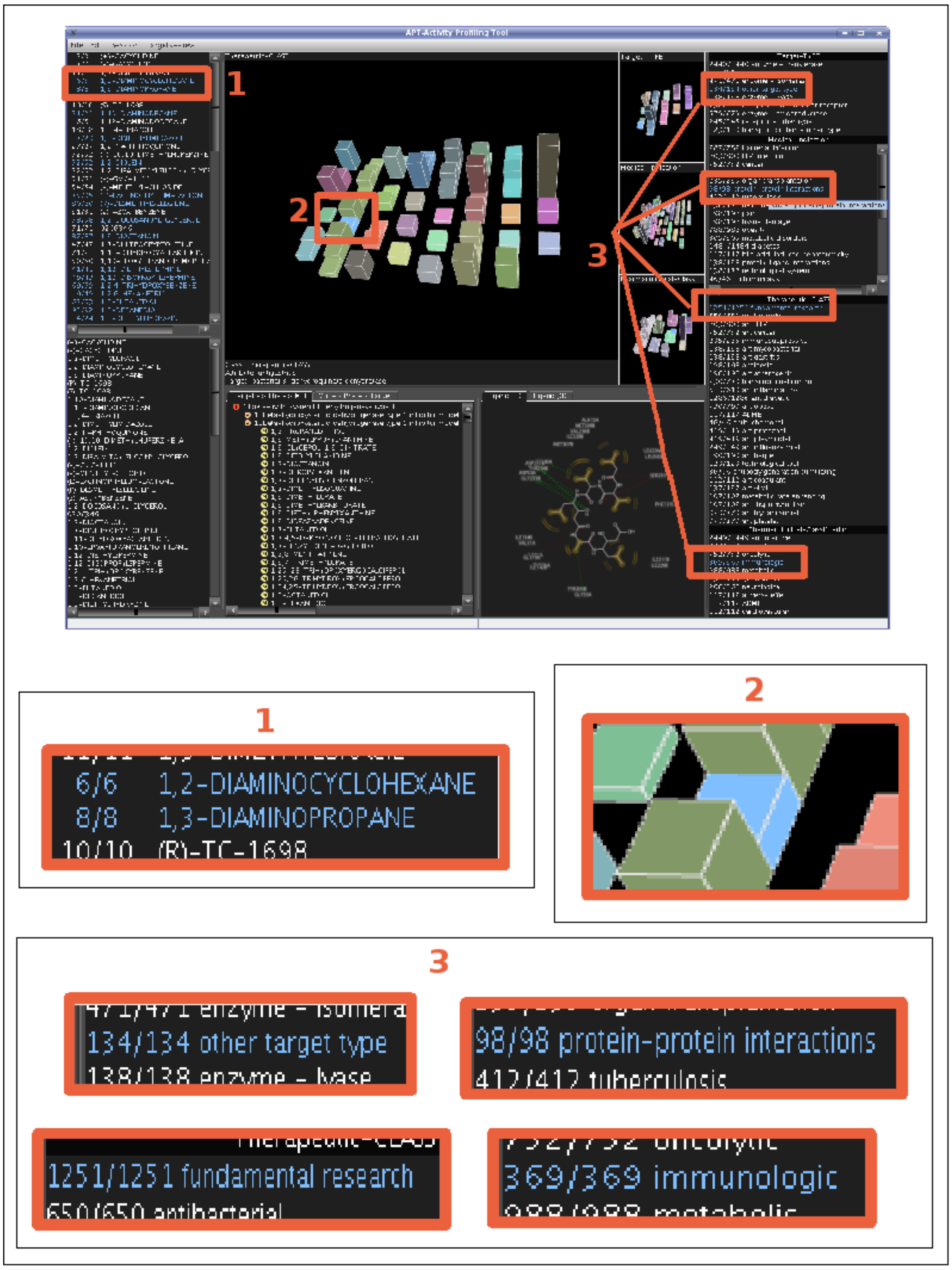
9.3.1 Highlighting and Filtering

There are three views supporting highlighting and filtering of hits, the Target View, the Attribute View, and the Molecule View. Whenever the user moves the mouse over data objects like targets or molecules, Apt highlights all data objects which would be affected by the filtering of this data object. An example is shown in Figure 9.5 and Figure 9.6. Figure 9.5 shows the situation when the user moves the mouse over the Attribute View and is above the Medical Indication *"protein-protein interactions"*. Filtering this attribute would result in filtering all hits of targets having the Medical Indication *"protein-protein interactions"*. This filtering would also affect some molecules, targets, and other attributes like *immunologic* (Pharmacological-Classification). Therefore, all affected objects are highlighted.

Figure 9.6 shows the situation after selecting the Medical Indication *"protein-protein interactions"* in the Attribute View. All hits of targets having the Medical Indication *"protein-protein interactions"* are automatically filtered. Figure 9.6 shows clearly that all highlighted objects are affected from this filtering.

9.3.2 Molecule Details

The two Tree Views and the Ligand-2D/3D views are linked with each other. To show the 2D or 3D representation of an molecule the appropriate hit in one of the



6/6 1,2-DIAMINOCYCLOHEXANE
8/8 1,3-DIAMINOPROPANE
10/10 (R)-TC-1698

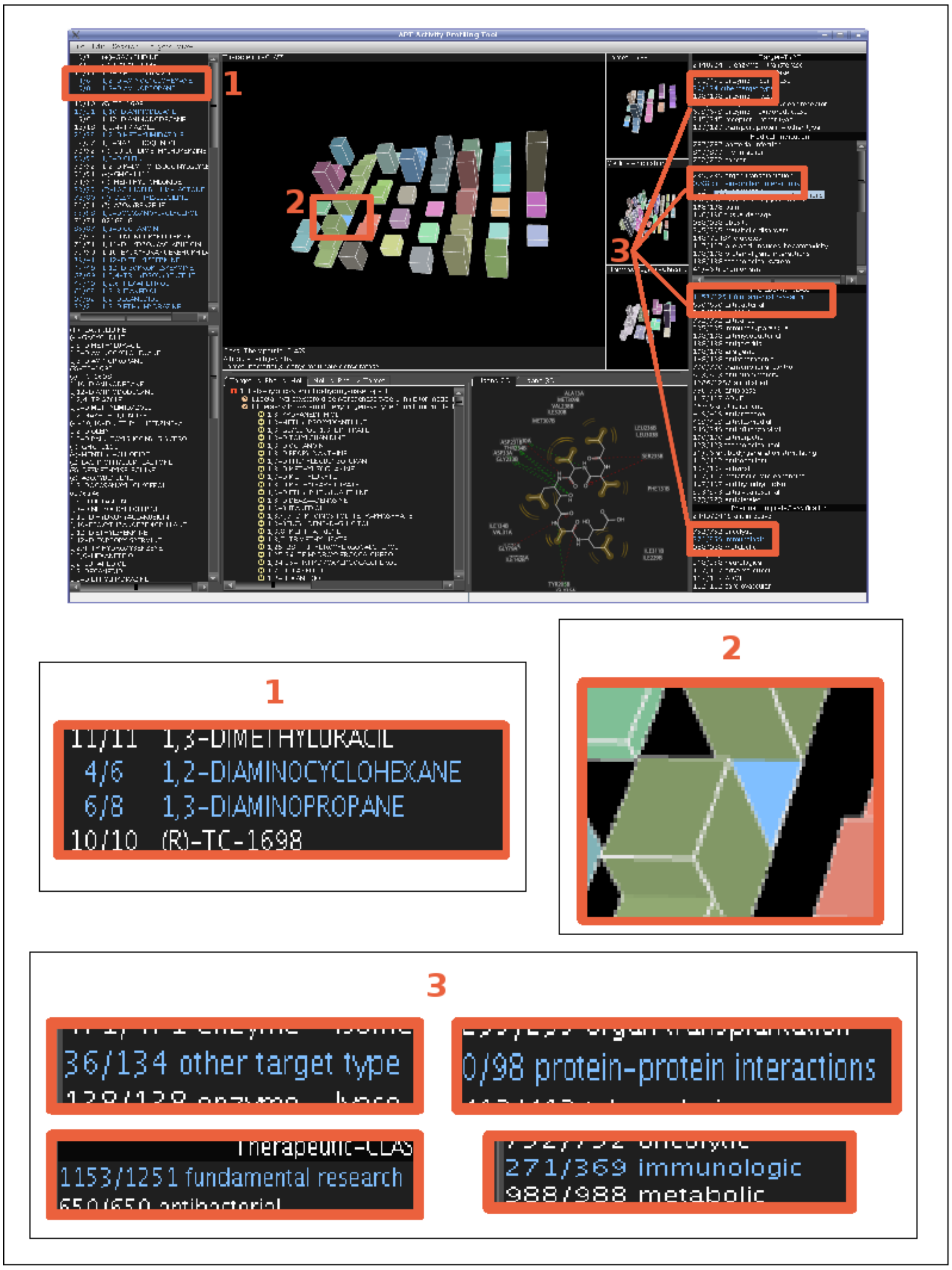
134/134 other target type
138/138 enzyme - lyase

98/98 protein-protein interactions
412/412 tuberculosis

1251/1251 fundamental research
650/650 antibacterial

369/369 immunologic

Figure 9.5: An example where the user moves with the mouse over “protein-protein interactions” in the Attribute View. In the general view the affected objects are surrounded with an orange box. These boxes are drawn to a larger scale below.

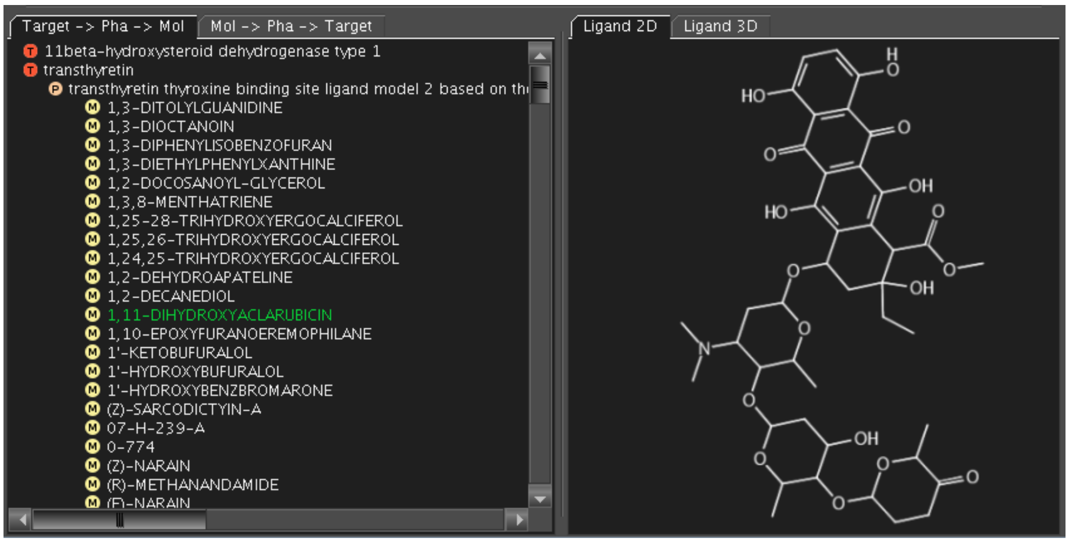


11/11 1,3-DIMETHYLURACIL
4/6 1,2-DIAMINOCYCLOHEXANE
6/8 1,3-DIAMINOPROPANE
10/10 (R)-TC-1698

36/134 other target type
0/98 protein-protein interactions
Therapeutic-CLAS
1153/1251 fundamental research
271/369 immunologic
988/988 metabolic

Figure 9.6: After selecting "protein-protein interactions" in the Attribute View all hits of targets having this attribute are filtered. In the general view the affected objects are surrounded with an orange box. These boxes are drawn to a larger scale below.

tree views can be selected. Figure 9.7 shows an example where the molecule *"1,11-DIHYDROXYACLARUBICIN"* was selected in the Tree View. The Ligand-2D view on the right shows the 2D structure of the molecule.



```
T 11beta-hydroxysteroid dehydrogenase type 1
T transthyretin
  P transthyretin thyroxine binding site ligand model 2 based on th
      M 1,3-DITOLYLGUANIDINE
      M 1,3-DIOCTANOIN
      M 1,3-DIPHENYLISOBENZOFURAN
      M 1,3-DIETHYLPHENYLXANTHINE
      M 1,2-DOCOSANOYL-GLYCEROL
      M 1,3,8-MENTHATRIENE
      M 1,25-28-TRIHYDROXYERGOCALCIFEROL
      M 1,25,26-TRIHYDROXYERGOCALCIFEROL
      M 1,24,25-TRIHYDROXYERGOCALCIFEROL
      M 1,2-DEHYDROAPATELINE
      M 1,2-DECANEDIOL
      M 1,11-DIHYDROXYACLARUBICIN
      M 1,10-EPOXYFURANOEREMOPHILANE
      M 1'-KETOBUFURALOL
      M 1'-HYDROXYBUFURALOL
      M 1'-HYDROXYBENZBROMARONE
      M (Z)-SARCODICTYIN-A
      M 07-H-239-A
      M 0-774
      M (Z)-NARAIN
      M (R)-METHANANDAMIDE
      M (E)-NARAIN
```

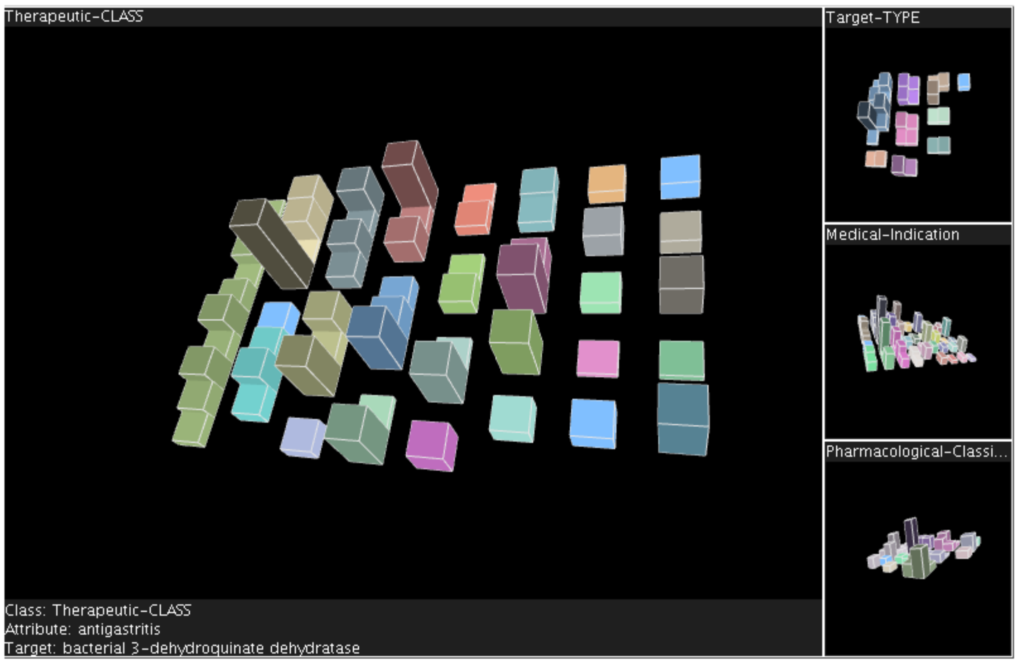
Figure 9.7: The green highlighted molecule is selected in the Tree View. The Ligand-2D View shows the 2D structure of the ligand.

9.4 Target View

The Target View of Apt is a very powerful tool for displaying activity profiles. For an effective visualization of parallel screening results a Heat- and Height Map has been combined (see Figure 9.8).

The view consists of multiple maps, one for each attribute class. Only one is displayed large and allows for user interaction. The others provide an overview of the activity profiles of their attribute class and are linked to the highlighting and filtering mechanism of Apt. Selecting one of these maps exchanges it with the large one.

Each of the maps are composed of multiple field clusters. A cluster is representing a target attribute and is made up of fields, one for each target having the corresponding attribute. Each target attribute and therefore also each cluster of fields has its own unique color. The intensity of the color of each field, which is the heat information of the map, indicates the amount of hits matching to the corresponding target. The darker the color of the field, the more hits the target has. The same



Class: Therapeutic-CLASS
Attribute: antigastritis
Target: bacterial 3-dehydroquinate dehydratase

🟩 fundamental research
🟦 antibacterial
🟥 anti HIV
⬜ anticancer
🟦 immunosuppressive
🟪 antimycobacterial
🟨 antigastritis
🟨 86 antidiabetic
🟩 antiobese
⬜ ADME
🟩 titrichomonal
🟩 antiprotozoal
🟦 antiplasmodial
⬛ antiinfluenza viral
🟩 antiherpetic
🟥 technological tool
🟪 tibody generation stimulating
🟩 anticoagulant
🟦 antiviral
🟪 metabolic rate enhancing
🟧 anti hypothyroidism
🟩 antitrypanosomal
🟦 antiplatelet

Figure 9.8: An example showing the powerful Target View of Apt and the color legend.

information is encoded through the height of each field. The higher the cuboid of the field, the more hits the target has.

In the main map of the Target View hits of a particular target can be filtered or unfiltered by selecting the appropriate field. Selecting a cluster results in filtering hits of all targets within the cluster. This is equivalent to selecting the attribute represented by the cluster in the Attribute View. When moving the mouse over a field of the main map the information panel below the map shows information about the field (attribute class, target attribute, and target name).

The maps are written in Java using JOGL (Java bindings for OpenGL) for hardware acceleration. One of the most important drawbacks of Hitvis, where a similar solution is used to display activity profiles, is that it gets performance problems if large molecule and target libraries are loaded. JOGL enables a high-performance implementation of the maps. Furthermore it offers a 3D API for a perspective visualization of the maps. Zooming and rotating the maps is also supported by Apt, which allows the user to navigate through the map and to have a different view of the activity profiles.

Another interesting aspect of the map is the layout of the clusters. The development of an applicable algorithm for the layout (see Appendix A.2) was also part of this work and is explained in the following. In Figure 9.9 the meaning of some terms, which are used in the explanation, are illustrated by example.

The steps of the algorithm are:

1. Sort the list of clusters by their amount of fields in descending order.

2. Initially set MaxFieldsPerRow and MaxFieldsPerColumn to the square root of the total amount of fields in the map rounded up.

3. Set MaxFieldsPerRowInCluster to MaxFieldsPerRow divided by the size of clusters to ensure that there are enough columns for the worst case where only one cluster fits to each column. Furthermore, if the largest cluster does not fit into the map because of insufficient rows, increase MaxFieldsPerRowInCluster until it does.

4. For each column of the map iterate through the sorted list of clusters and insert clusters, fitting into the column, starting from the top of the column. The fact that the list is sorted ensures that the largest possible clusters are always inserted first.

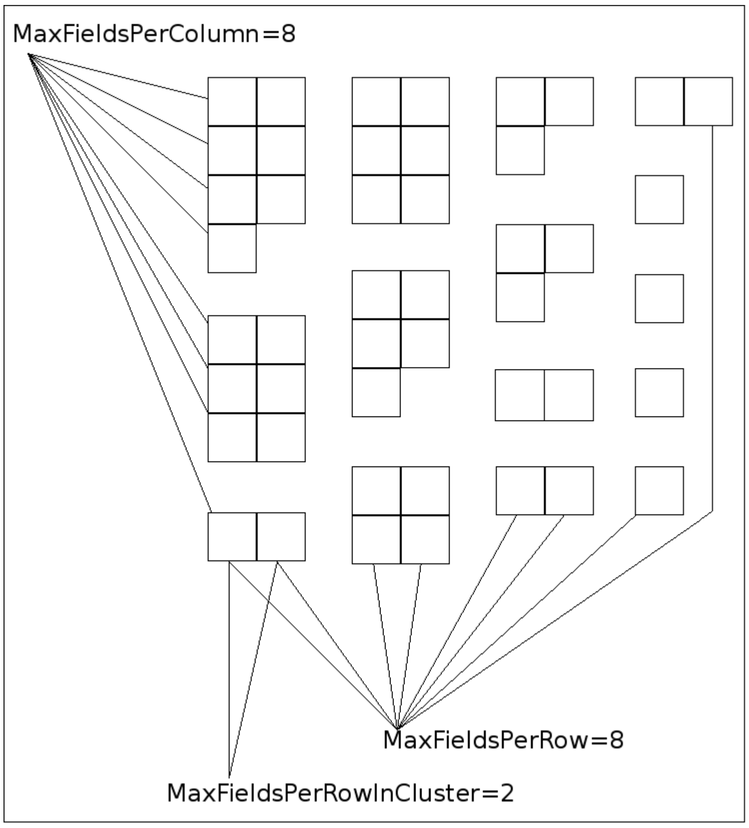


Figure 9.9: Terms used in the Layout-Algorithm of the maps in the Target View.

5. Now, after all columns are iterated through, check if there are any clusters left, which are not placed on the map. If true, increase MaxFieldsPerRow and MaxFieldsPerColumn by one and continue with step 3, otherwise the creation of the map is finished.

Implementation

The implementation of the Target View is split into a logic- and a user interface part.

The logic part (*TargetViewController*) handles the logic for the backend and sets the data to be shown in the user interface components.

In the user interface part of the Target View there is the class *TargetView*. Every change in the data is forwarded from the *TargetViewController* to the *TargetView*. The *TargetView* itself consists of multiple *TargetMap*s, one for each attribute class. The *TargetMap* implements the *GLEventListener* interface which is needed to add it to the GLCanvas, which is a heavyweight container of the JOGL API. A schematic overview of the implementation of the Target View is shown in Figure 9.10.

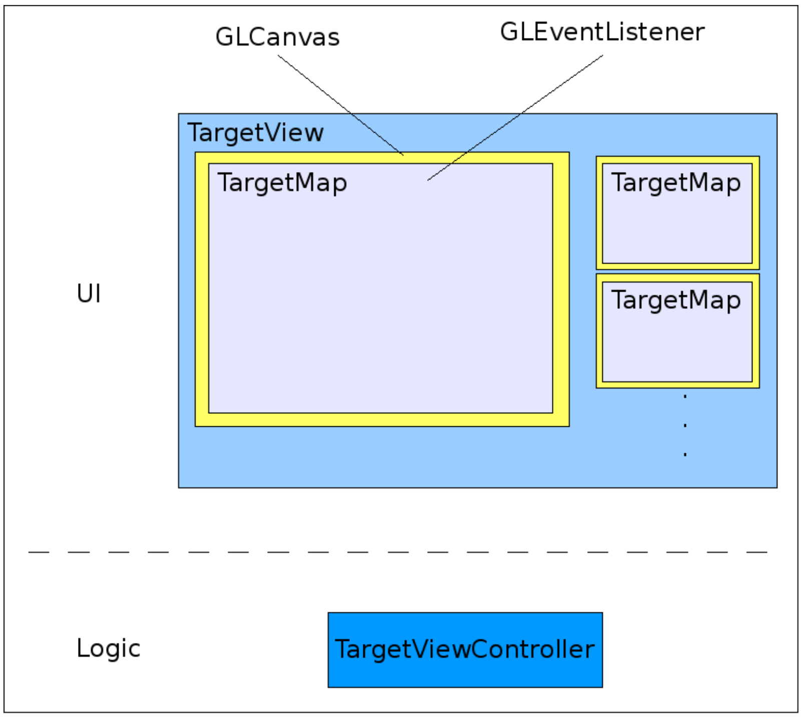


Figure 9.10: A schematic overview of the implementation of the Target View.

9.5 Attribute View

The Attribute View is a tabular view of all target attributes classified by attribute type/class (see Figure 9.11 on page 71).

Starting from the left, each row shows the following information of a target attribute:

- The unique color of the attribute, which is the same used for the corresponding cluster in the Target View.
- The amount of unfiltered hits of targets having this attribute.
- The total amount of hits of targets having this attribute.
- The name of the target attribute.

The Attribute View allows for an interactive highlighting and filtering of hits like the Target- and Molecule View. Moving the mouse over attributes highlights all objects in the mentioned views that would be affected by the filtering caused by the selection of the attribute. All views are updated if a filtering action is performed in the Attribute View and vice versa.

The attributes of each attribute class are put into separate scroll panes and each scroll pane is resizeable on the y-axis. This allows the user to analyze more attributes of an attribute class by enlarging its view.

Implementation

Like the implementation of the Target View, the implementation of the Attribute View is divided into a logic- and a user interface part (see Figure 9.12).

The *AttributeViewController* handles the backend operations and the *Attribute-View* is responsible for drawing the user interface components of the Attribute View. The *AttributeView* is composed of multiple *AttributeClassPanel*s, one for each attribute class. Each AttributeClassPanel uses an *IlibScrollPane* to display the target attributes of the class.

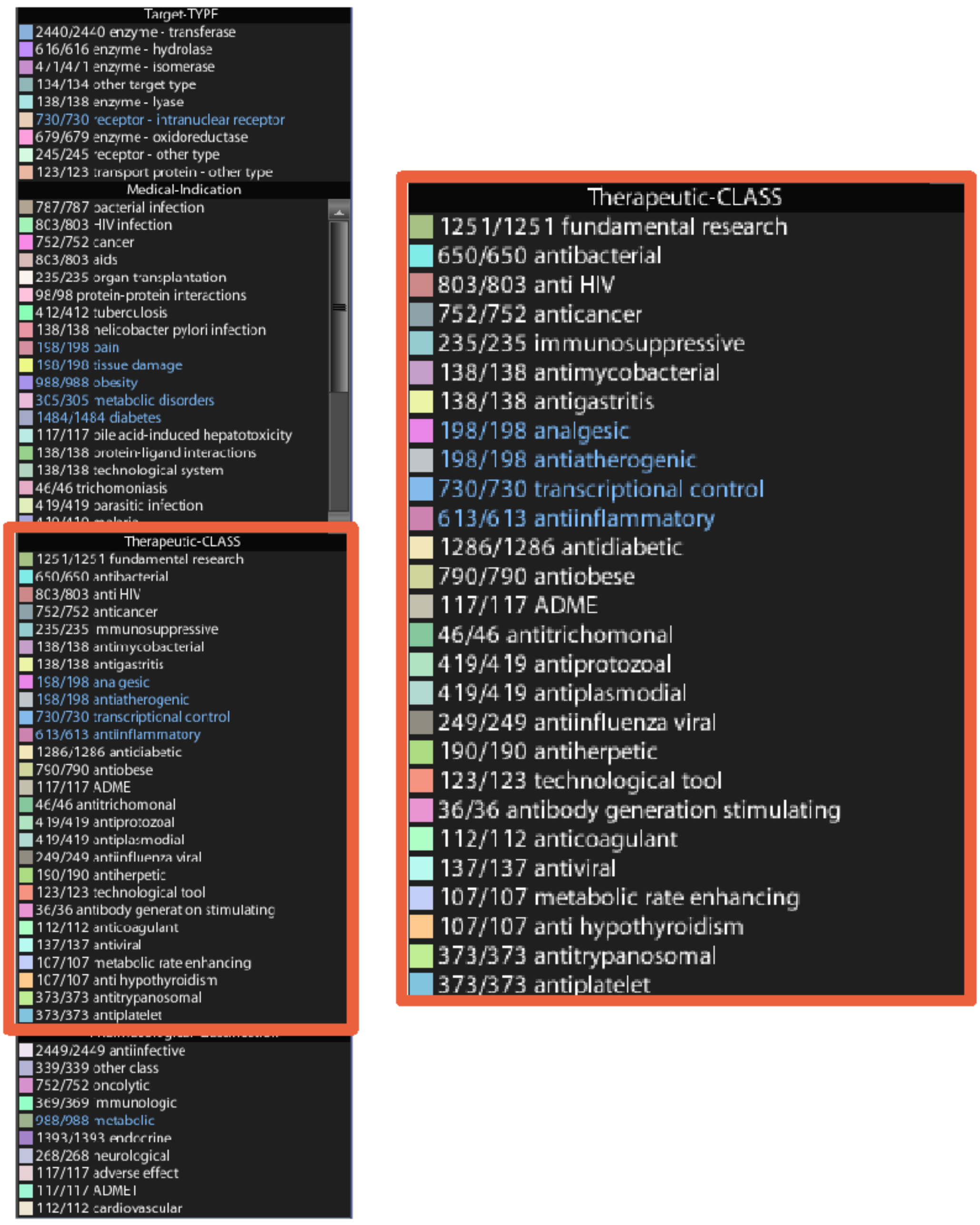
9.6 Molecule View

The Molecule View is made up of two separate views. One is showing all molecules which have fit to any of the targets of the set selected at the very beginning of the session. The other is only showing the molecules remaining after all filters have been applied and is therefore the result list of the virtual screening and activity profiling process. The first one is shown in Figure 9.13 on the left and is build up very similar to the Attribute View. Starting from the left each row contains the following information of a molecule:

- The amount of unfiltered hits of the molecule.
- The total amount of hits of the molecule.
- The name of the molecule.

This view also supports the interactive highlighting and filtering of hits, which works exactly the same as in the Target View and the Attribute View.

The second view is depicted in Figure 9.13 on the right and displays only the names of the molecules. No user interaction is supported, but the view is immediately updated whenever hits are filtered in any of the views.



Target-TYPF

2440/2440 enzyme - transferase
616/616 enzyme - hydrolase
471/471 enzyme - isomerase
134/134 other target type
138/138 enzyme - lyase
730/730 receptor - intranuclear receptor
679/679 enzyme - oxidoreductase
245/245 receptor - other type
123/123 transport protein - other type

Medical-Indication

787/787 bacterial infection
803/803 HIV infection
752/752 cancer
803/803 aids
235/235 organ transplantation
98/98 protein-protein interactions
412/412 tuberculosis
138/138 helicobacter pylori infection
198/198 pain
198/198 tissue damage
988/988 obesity
305/305 metabolic disorders
1484/1484 diabetes
117/117 bile acid-induced hepatotoxicity
138/138 protein-ligand interactions
138/138 technological system
46/46 trichomoniasis
419/419 parasitic infection
419/419 malaria

Therapeutic-CLASS

1251/1251 fundamental research
650/650 antibacterial
803/803 anti HIV
752/752 anticancer
235/235 immunosuppressive
138/138 antimycobacterial
138/138 antigastritis
198/198 analgesic
198/198 antiatherogenic
730/730 transcriptional control
613/613 antiinflammatory
1286/1286 antidiabetic
790/790 antiobese
117/117 ADME
46/46 antitrichomonal
419/419 antiprotozoal
419/419 antiplasmodial
249/249 antiinfluenza viral
190/190 antiherpetic
123/123 technological tool
36/36 antibody generation stimulating
112/112 anticoagulant
137/137 antiviral
107/107 metabolic rate enhancing
107/107 anti hypothyroidism
373/373 antitrypanosomal
373/373 antiplatelet

2449/2449 antiinfective
339/339 other class
752/752 oncolytic
369/369 immunologic
988/988 metabolic
1393/1393 endocrine
268/268 neurological
117/117 adverse effect
117/117 ADMET
112/112 cardiovascular

Therapeutic-CLASS

1251/1251 fundamental research
650/650 antibacterial
803/803 anti HIV
752/752 anticancer
235/235 immunosuppressive
138/138 antimycobacterial
138/138 antigastritis
198/198 analgesic
198/198 antiatherogenic
730/730 transcriptional control
613/613 antiinflammatory
1286/1286 antidiabetic
790/790 antiobese
117/117 ADME
46/46 antitrichomonal
419/419 antiprotozoal
419/419 antiplasmodial
249/249 antiinfluenza viral
190/190 antiherpetic
123/123 technological tool
36/36 antibody generation stimulating
112/112 anticoagulant
137/137 antiviral
107/107 metabolic rate enhancing
107/107 anti hypothyroidism
373/373 antitrypanosomal
373/373 antiplatelet

Figure 9.11: An overview of the Attribute View is shown on the left, the attribute class Therapeutic-Class on the right.

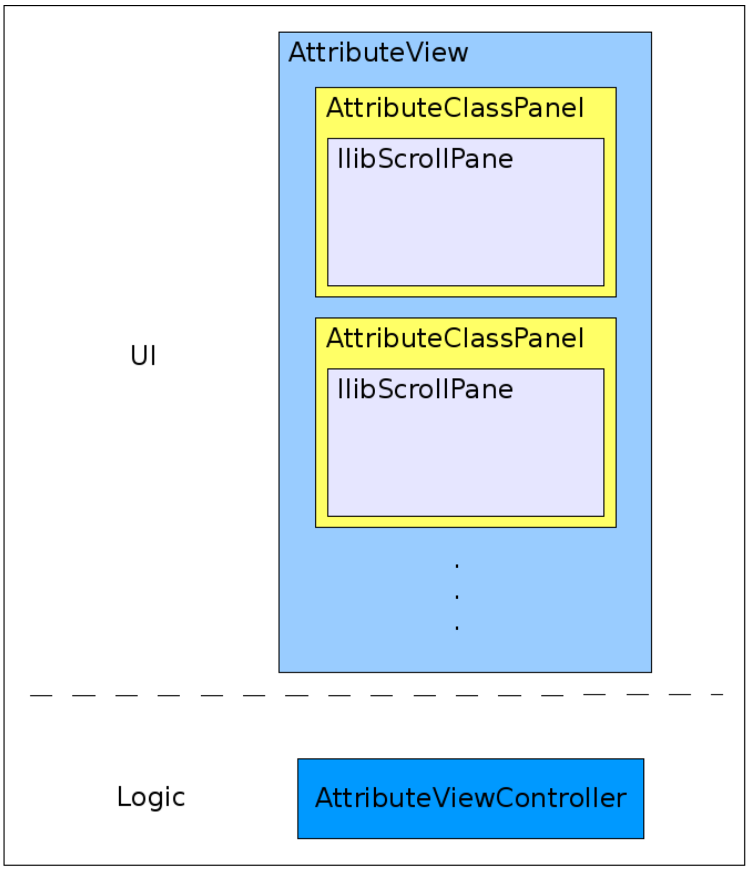
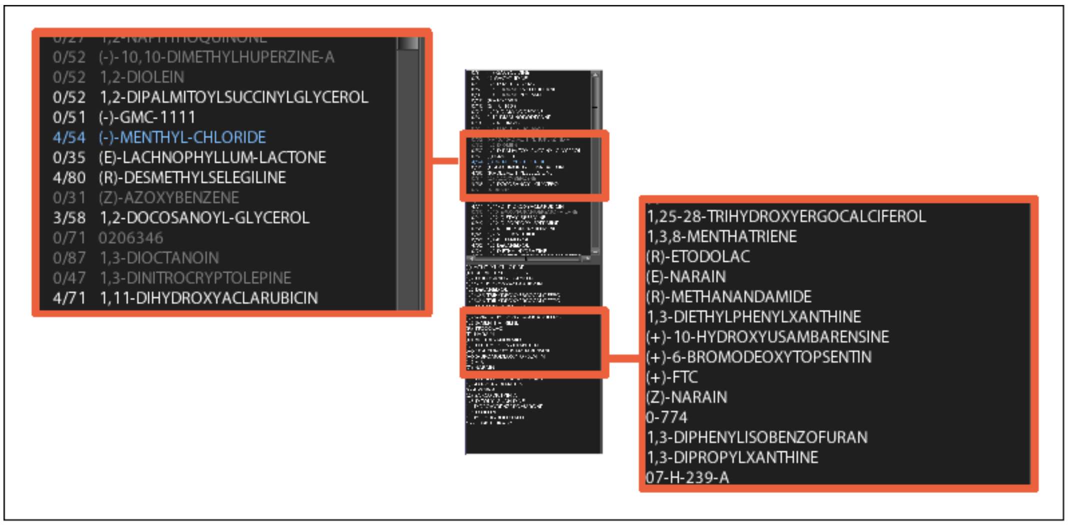


Figure 9.12: A schematic overview of the implementation of the Attribute View.



```
0/52  (-)-10,10-DIMETHYLHUPERZINE-A
0/52  1,2-DIOLEIN
0/52  1,2-DIPALMITOYLSUCCINYLGLYCEROL
0/51  (-)-GMC-1111
4/54  (-)-MENTHYL-CHLORIDE
0/35  (E)-LACHNOPHYLLUM-LACTONE
4/80  (R)-DESMETHYLSELEGILINE
0/31  (Z)-AZOXYBENZENE
3/58  1,2-DOCOSANOYL-GLYCEROL
0/71  0206346
0/87  1,3-DIOCTANOIN
0/47  1,3-DINITROCRYPTOLEPINE
4/71  1,11-DIHYDROXYACLARUBICIN
```

```
1,25-28-TRIHYDROXYERGOCALCIFEROL
1,3,8-MENTHATRIENE
(R)-ETODOLAC
(E)-NARAIN
(R)-METHANANDAMIDE
1,3-DIETHYLPHENYLXANTHINE
(+)-10-HYDROXYUSAMBARENSINE
(+)-6-BROMODEOXYTOPSENTIN
(+)-FTC
(Z)-NARAIN
0-774
1,3-DIPHENYLISOBENZOFURAN
1,3-DIPROPYLXANTHINE
07-H-239-A
```

Figure 9.13: The two views of the Molecule View.

Implementation

The implementation of the Molecule View is also split up into a logic- and a user interface part. The *MoleculeViewController* handles the logic and the *MoleculeView* the user interface. The user interface is build up of two containers, one for the interactive view which is an *IlibScrollPane* with a tabular visualization of the molecules and their hit amounts. The other container is also an *IlibScrollPane* and contains *MoleculeNameLabel*s. Figure 9.14 shows a schematic overview of the implementation of the Molecule View.

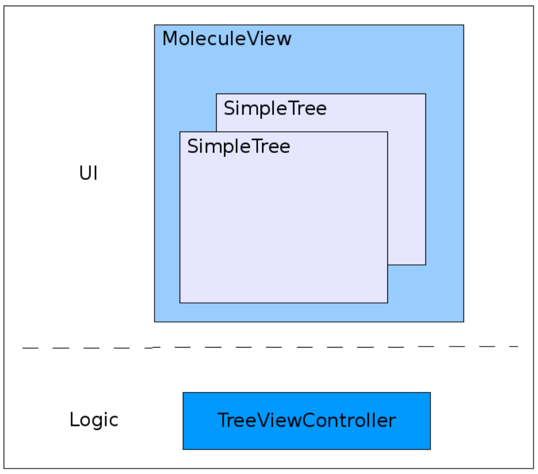
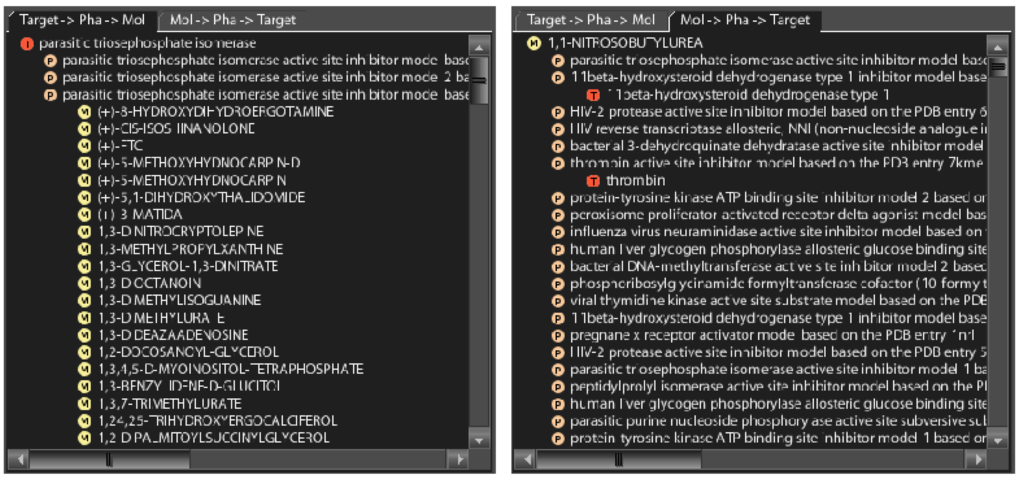


Figure 9.14: A schematic overview of the implementation of the Molecule View.

9.7 Tree Views

In Apt there are two hierarchical views, which are showing: molecules, pharmacophores, and targets of unfiltered hits. The difference between the two views is the different point of view, respectively. In one of the two views the top level nodes are molecules and in the other view the top level nodes are targets. They are placed into a tabbed pane and can only be displayed sequentially. Figure 9.15 shows both views side by side. The views do not support highlighting and interactive filtering of hits, but changes in other views are immediately updated in the trees. Furthermore, the views are linked with the detail views. Selecting a hit in one of the tree views shows the appropriate molecule in the detail views.



```
Target -> Pha -> Mol | Mol -> Pha -> Target
parasitic triosephosphate isomerase
    parasitic triosephosphate isomerase active site inhibitor model base
    parasitic triosephosphate isomerase active site inhibitor model 2 ba
    parasitic triosephosphate isomerase active site inhibitor model base
        (+)-8-HYDROXYDIHYDROERGOTAMINE
        (+)-CIS-ISOSHINANOLONE
        (+)-FTC
        (+)-5-METHOXYHYDNOCARPIN-D
        (+)-5-METHOXYHYDNOCARPIN
        (+)-5,1-DIHYDROXYTHALIDOMIDE
        (+)-3-MATIDA
        1,3-DINITROCRYPTOLEPINE
        1,3-METHYLPROPYLXANTHINE
        1,3-GLYCEROL-1,3-DINITRATE
        1,3-DIOCTANOIN
        1,3-DIMETHYLISOGUANINE
        1,3-DIMETHYLURACILE
        1,3-DIDEAZAADENOSINE
        1,2-DOCOSANOYL-GLYCEROL
        1,3,4,5-D-MYOINOSITOL-TETRAPHOSPHATE
        1,3-BENZYLIDENE-D-GLUCITOL
        1,3,7-TRIMETHYLURATE
        1,24,25-TRIHYDROXYERGOCALCIFEROL
        1,2-DIPALMITOYLSUCCINYLGLYCEROL
```

```
Target -> Pha -> Mol | Mol -> Pha -> Target
1,1-NITROSOBUTYLUREA
    parasitic triosephosphate isomerase active site inhibitor model base
    11beta-hydroxysteroid dehydrogenase type 1 inhibitor model base
        11beta-hydroxysteroid dehydrogenase type 1
    HIV-2 protease active site inhibitor model based on the PDB entry 6
    HIV reverse transcriptase allosteric, NNI (non-nucleoside analogue i
    bacterial 3-dehydroquinate dehydratase active site inhibitor model
    thrombin active site inhibitor model based on the PDB entry 7kme
        thrombin
    protein-tyrosine kinase ATP binding site inhibitor model 2 based or
    peroxisome proliferator activated receptor delta agonist model bas
    influenza virus neuraminidase active site inhibitor model based on
    human liver glycogen phosphorylase allosteric glucose binding site
    bacterial DNA-methyltransferase active site inhibitor model 2 based
    phosphoribosylglycinamide formyltransferase cofactor (10 formyl t
    viral thymidine kinase active site substrate model based on the PDE
    11beta-hydroxysteroid dehydrogenase type 1 inhibitor model base
    pregnane x receptor activator model based on the PDB entry 1nrl
    HIV-2 protease active site inhibitor model based on the PDB entry 5
    parasitic triosephosphate isomerase active site inhibitor model 1 ba
    peptidylprolyl isomerase active site inhibitor model based on the PI
    human liver glycogen phosphorylase allosteric glucose binding site
    parasitic purine nucleoside phosphorylase active site subversive sul
    protein tyrosine kinase ATP binding site inhibitor model 1 based or
```

Figure 9.15: The *Tree Views* of Apt, which are showing the molecules, pharmacophores, and targets of all unfiltered hits.

Implementation

The implementation of the tree views is similar to other views of Apt. The *TreeViewController* manages the logic of the trees and the *TreeView* class handles the user interface of the trees. For the visualization of the trees *SimpleTree* was developed and used instead of the default *JTree*. The reason for this is the lack of performance of the *JTree* when dynamically inserting or removing thousands of nodes. The *SimpleTree*s are both placed in *IlibTabbedPane*s. The implementation of the tree views is illustrated in Figure 9.16.

9.8 Detail Views

There are two detail views in Apt. One is showing the 2D structure and the other one the 3D structure of a ligand. The view is linked with the tree views. If any hit is selected in one of the tree views the corresponding ligand is shown in the two detail views. Just as the tree views, the detail views are placed in a tabbed pane. Both views are part of Inte:ligands API. Figure 9.17 on page 76 shows them side by side.

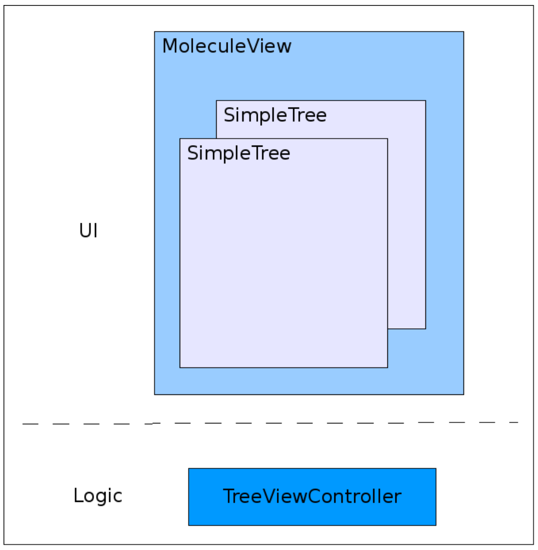


Figure 9.16: A schematic overview of the implementation of the hierarchical views of Apt.

Implementation

As mentioned above, the views are part of Inte:ligand's API [63], only a controller for backend handling and access to the components was implemented (*DetailsViewController*).

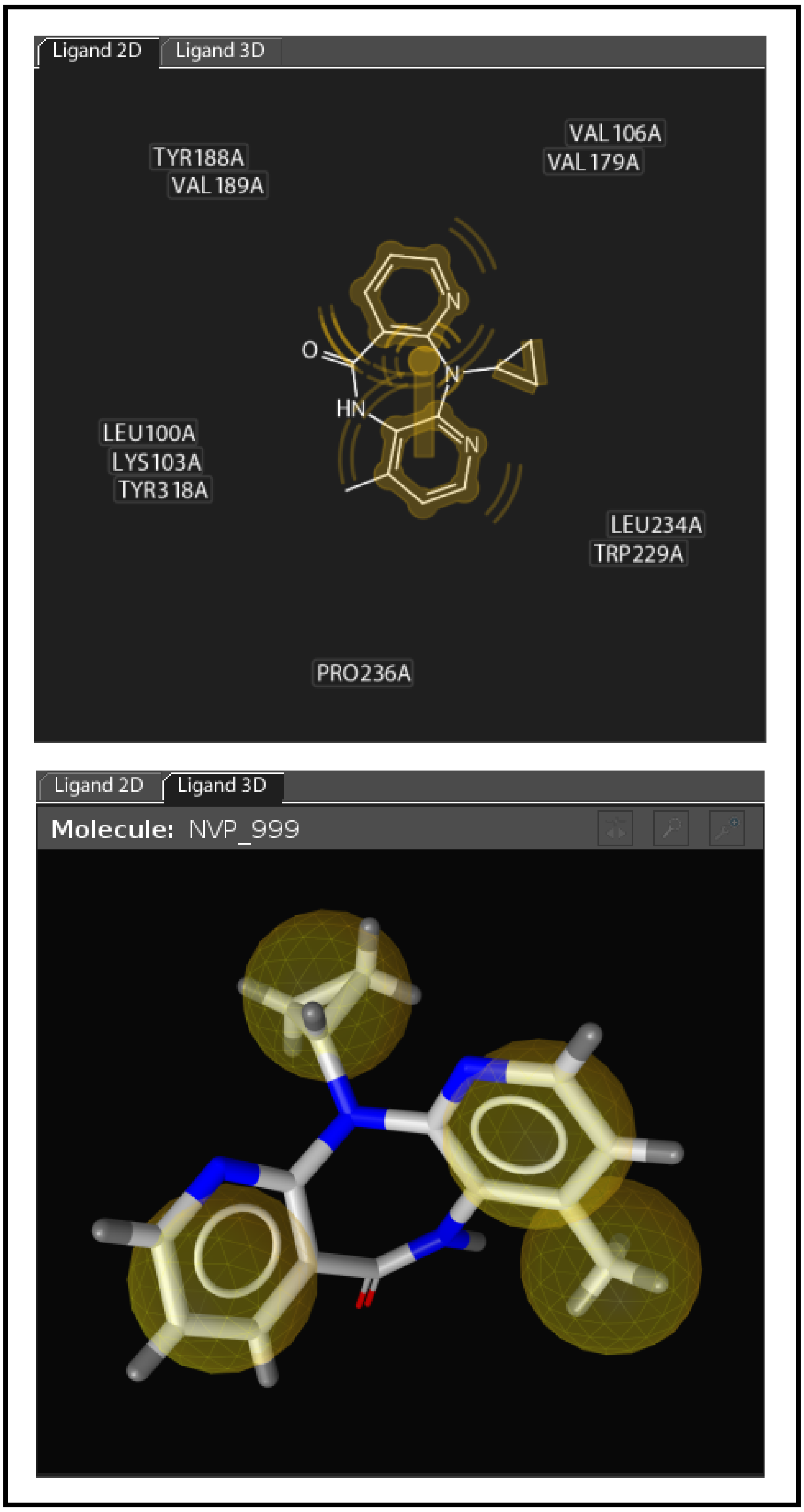


Figure 9.17: The detail views of Apt, which are showing the 2D and 3D structure of molecules.

Chapter 10

Summary and Outlook

In the following a summary of this work is given, together with the results of the evaluation of our tool called Apt showing the benefits and drawbacks of the new approach. Based on the results of the evaluation an outlook for future improvements is given afterwards.

Recent advances in informatics have induced many research fields to use computer-aided methods for their purposes. One of these fields is drug discovery. The goal of drug discovery is to find chemicals, which cause a particular biological activity in human body that helps human beings to cure from diseases. A computer-based approach to achieve this is virtual screening, where large libraries of molecules are screened virtually against a drug target. But screening molecule libraries against a single target does not show side effects. Therefore, tests are applied after the screening process, where drug candidates are tested against multiple targets. In activity profiling, which combines these two methods, molecule libraries are screened against target sets. To analyze the huge amount of results of this process advanced visualization techniques are required.

Ptft and Hitvis are two applications developed by Inte:ligand which are dealing with this problem. Both are implemented in multiple view systems (see chapter 6) using user-interactive linking and brushing (see chapter 7) methods for filtering and visualizing parallel virtual screening results (see chapter 3). The pros and cons of both approaches were discussed in chapter 8. These two tools formed the starting point for the implementation of a new tool: Apt (Activity Profiling Tool).

The development of Apt, which should combine advantages of both approaches, Ptft and HitVis, and address the drawbacks of them, was part of this work and is described in chapter 9. Especially the 3D map with heat and height information of Apt seems to be very promising. Apt was evaluated by two members of the Institute of Pharmacy of the Leopold-Franzens-University of Innsbruck, who are experts in this field: Dr. Thierry Langer and Dr. Daniela Schuster. The evaluation took three hours, was not moderated and was done in the form of brainstoarming. The results of the evaluation are:

- The spaces left between the clusters lead to a better separation of the attributes.
- The use of the color intensity as heat information avoids that too much information is encoded by colors, which was an important point of criticism of Hitvis.
- The heat information is now local and not global anymore. The reason therefore is the use of different colors in each cluster, so heat information is hard to compare through multiple clusters. Therefore the height information got more important.
- The user can access the height information of the map much better by zooming and rotating the 3D map.
- The performance of Apt seems to be good even though thousands of molecules and targets are screened.
- The visualization of new incoming hits, by highlighting them, is very intuitive, but may disturb the user if the frequency of new incoming hits is too high.
- This disturbance factor has been addressed by introducing a pause function for pausing a screening session.
- Only the tree views are linked to the detail views. Additional links to the Ligand 2D/3D view are desired.

The evaluation shows that the improvements of Apt are very valuable and promising. But it also shows that some issues remain. Together with features, which were not implemented yet, some further possible improvements remain, which are described in the following:

- When moving the mouse over the Target-, Attribute- or Molecule View Apt highlights all elements that would be affected from filtering. However, the extent of the filtering is not shown. Smooth Brushing, which is described in chapter 7 could be a method to extract this information.
- The bars in Ptft provide a very fast way for comparing hit amounts and are able to show the correlation between attributes of two different attribute classes. They could be integrated in Apt by combining them with the numerical representation of the hit amounts in the Attribute View.
- The detail views are only linked to the Tree Views and could be linked to other views as well, especially to the Molecule View.

Chapter 11

Conclusions

The visualization of n:m drug target interactions is a difficult task since it involves huge amounts of data. The fact that the information to be displayed is very complex requires the use of advanced visualization techniques. But hardware and software limits are a challenge for developers trying to implement user-friendly software. Therefore, often trade-offs between usability and performance have to be done and to make the right decisions it is crucial to consult the users. This was an important aspect of this thesis. Together with chemists from the University of Innsbruck Ptft and Hitvis were analyzed in detail. The insights gained from these discussions were an important starting point for developing Apt.

The most important improvements of Apt are:

- Dynamic screening
- Hardware acceleration by using OpenGL
- Improved layout of the heat map
- 3D visualization of the heat map, which supports zooming, and rotating the map
- Improved highlighting and filtering of hits
- Ligand-2D/3D view

Finally, there is an important insight I have gained working on this thesis, which I would like to note.

Information visualization is a research field where permanently new techniques are introduced. It is important to know how and when to use them. But, in my opinion, it is even more important to be creative enough to combine these techniques and assemble a custom visualization form, which exactly fits to the dealt problem.

Acknowledgments

There are some people I would like to thank. First, my supervisors Monika Lanzenberger and Gerhard Wolber. **Monika Lanzenberger** provided me a lot of useful and interesting information and material about visualization. **Gerhard Wolber** made the infrastructure of Inte:ligand available to me. Furthermore, he motivated and supported me throughout my thesis.

Special thanks go to **Fabian Bendix** who introduced me to Inte:ligands API and helped me whenever I had problems with OpenGL.

Furthermore, I would like to thank **Andreas Liebig**, who helped me implementing the interface for the screening client.

Moreover, thanks to Christine Buu, Philipp Adaktylos and Thomas Seidel. **Christine Buu** explained me a lot of biological and chemical staff in an articulate and precise way. **Philipp Adaktylos** and **Thomas Seidel** provided me some useful information about Inte:ligands API.

I am also thankful to all members of the Institute of Pharmacy of the Leopold-Franzens-University of Innsbruck, who helped me evaluating Ptft, HitVis and Apt: **Thierry Langer**, **Daniela Schuster**, **Klaus Liedl**, **Gudrun Hackspiel**, **Patrick Markt**, **Christian Laggner**, **Johannes Kirchmair**, **Hannes Wallnöfer** and **Simona Distinto**.

Finally, I would like to thank my friends and family for their patience and support.

This work is supported by **Inte:ligand** Software-Entwicklungs und Consulting GmbH.

Appendix A

Code Samples

A.1 Screening Client

In the following the source code of the interfaces and classes of the screening client implementation of Apt is provided.

A.1.1 IScreeningClient

```java
public interface IScreeningClient {
  public String start(String mollibFilePath, Vector<String>
      pharmacophores) throws ScreeningClientException;

  public void stop();

  public void reconnect(String id) throws ScreeningClientException;

  public void addHitHandler(IHitHandler handler);

  public void removeHitHandler(IHitHandler handler);
}
```

A.1.2 LocalScreeningClient

```java

public class LocalScreeningClient implements IScreeningClient{

  private class ScreeningThread extends Thread {

    private CompressedMoleculeLibraryAccessor moleculeLibraryAccessor;
    private int totalWorkUnits;
    private long startTime;
    private boolean stop;

    private ScreeningThread() throws ScreeningClientException {
      try {
        moleculeLibraryAccessor = new
            CompressedMoleculeLibraryAccessor(mflFilePath);
      } catch (Exception e) {
        e.printStackTrace();
        throw new ScreeningClientException(e.getMessage());
      }
      totalWorkUnits=moleculeLibraryAccessor.getSize()*pharmacophores.
          size();
      startTime=System.currentTimeMillis();
      stop=false;
    }

    public void run() {
      try {
        long lastReturnedResult=startTime;
        int mIndex = molIndex;
        for(int j=phaIndex; j<pharmacophores.size(); j++) {
          prefs.storePreference(PHA_INDEX, new Integer(j).toString());
          String phaId = pharmacophores.get(j);
          Pharmacophore pha = importPharmacophore(phaId);
          for(int i=mIndex; i<=moleculeLibraryAccessor.getSize(); i++)
              {
            if(stop)
              return;
```

```
                prefs.storePreference(MOL_INDEX, new Integer(i).toString()
                    );
                System.out.println("Screening:␣phaIndex="+j+"␣molIndex="+i
                    );
                for(IHitHandler hitHandler : hitHandlers)
                  hitHandler.progressUpdate(new Progress(totalWorkUnits,j*
                      moleculeLibraryAccessor.getSize()+(i-1),startTime,
                      lastReturnedResult));
                Molecule mol = moleculeLibraryAccessor.getMoleculeAt(i);
                if(match(mol, pha)) {
                  MoleculeLibraryAccessorMoleculeInfo molInfo = new
                      MoleculeLibraryAccessorMoleculeInfo();
                  molInfo.setMoleculeName(mol.getName());
                  molInfo.setLibraryPath(mflFilePath);
                  molInfo.setPosition(i);
                  ScreeningWorkResultData result = new
                      ScreeningWorkResultData(molInfo, phaId,true);
                  for(IHitHandler hitHandler : hitHandlers)
                    hitHandler.resultReceived(result);
                  lastReturnedResult=System.currentTimeMillis();
                }
              }
              mIndex=1;
            }
            for(IHitHandler hitHandler : hitHandlers)
              hitHandler.progressUpdate(new Progress(totalWorkUnits,
                  totalWorkUnits,startTime,lastReturnedResult));
            prefs.getFile().delete();
          }catch (Exception e) {
            e.printStackTrace();
          }
      }

      private void stopThread() {
        this.stop=true;
      }
    }

```

```
private static final String PHAS = "phas";
private static final String MOL_INDEX = "molIndex";
private static final String PHA_INDEX = "phaIndex";
private static final String FILE = "file";
private static final String SEPERATOR = ",";
private DBData dbData;
private Vector<IHitHandler> hitHandlers;
private String localScreeningClientsDir;
private Preferences prefs;
private ScreeningThread screeningThread;
private String mflFilePath;
private Vector<String> pharmacophores;
private int phaIndex;
private int molIndex;

public LocalScreeningClient(DBData dbData) throws IOException {
  this.dbData = dbData;
  hitHandlers = new Vector<IHitHandler>();
  AptApplicationController controller = (AptApplicationController)
     RuntimeContext.getRuntimeContext().getApplicationController();
  localScreeningClientsDir = controller.getLocalScreeningClientsDir
     ();
}

public void addHitHandler(IHitHandler handler) {
  hitHandlers.add(handler);
}

public void reconnect(String id) throws ScreeningClientException {
  try {
    prefs = new Preferences(localScreeningClientsDir+"/"+id);
  } catch (IOException e) {
    e.printStackTrace();
    throw new ScreeningClientException(e.getMessage());
  }
  this.mflFilePath = prefs.getPreference(FILE);
  this.pharmacophores = getPhaIds(prefs.getPreference(PHAS));
  this.phaIndex = Integer.parseInt(prefs.getPreference(PHA_INDEX));
```

```java
    this.molIndex = Integer.parseInt(prefs.getPreference(MOL_INDEX));
    screeningThread = new ScreeningThread();
    screeningThread.start();
  }

  public void removeHitHandler(IHitHandler handler) {
    hitHandlers.remove(handler);
  }

  public String start(final String mflFilePath, final Vector<String>
      pharmacophores) throws ScreeningClientException {
    String id = this.getClass().getSimpleName()+"-"+System.
        currentTimeMillis();
    try {
      prefs = new Preferences(localScreeningClientsDir+"/"+id);
      prefs.storePreference(FILE, mflFilePath);
      prefs.storePreference(PHA_INDEX, new Integer(0).toString());
      prefs.storePreference(MOL_INDEX, new Integer(1).toString());
      prefs.storePreference(PHAS, getPhaStrings(pharmacophores));
    } catch (IOException e) {
      e.printStackTrace();
      throw new ScreeningClientException(e.getMessage());
    }
    this.mflFilePath = mflFilePath;
    this.pharmacophores = pharmacophores;
    this.phaIndex=0;
    this.molIndex=1;
    screeningThread = new ScreeningThread();
    screeningThread.start();
    return id;
  }

  private String getPhaStrings(Vector<String> pharmacophores) {
    StringBuffer buffer = new StringBuffer();
    boolean appendComma = false;
    for(String phaId : pharmacophores) {
      if(appendComma)
        buffer.append(SEPERATOR);
        else
            appendComma=true;
        buffer.append(phaId);
    }
    return buffer.toString();
}

private Vector<String> getPhaIds(String phaString) {
    Vector<String> phaIds = new Vector<String>();
    for(String phaId : phaString.split(SEPERATOR))
        phaIds.add(phaId);
    return phaIds;
}

public void stop() {
    screeningThread.stopThread();
    prefs.getFile().delete();
}

private Pharmacophore importPharmacophore(String pharmacophoreId)
throws Exception {
    try {
        return new DBPharmacophoreAccessor(dbData)
        .getPharmacophore(pharmacophoreId);
    } catch (Exception e) {
        System.err.println("Error␣importing␣pharmacophore:␣"
            + e.getMessage());
        e.printStackTrace();
        throw e;
    }
}

private boolean match(Molecule molecule, Pharmacophore pharmacophore
    )
throws Exception {
    for(int i=0; i<molecule.getNumberOfConformations(); i++) {
        if(match(molecule, pharmacophore, i))
            return true;
    }
    return false;
  }

  private boolean match(Molecule molecule, Pharmacophore pharmacophore
      , int index)
  throws Exception {
    AlignmentElement moleculeAlignmentElement;
    molecule.setActiveConformation(index);
    moleculeAlignmentElement = new AlignmentElement(molecule);
    AlignmentElement pharmacophoreAlignmentElement = new
        AlignmentElement(pharmacophore);
    Alignment alignment = new Alignment(moleculeAlignmentElement,
        pharmacophoreAlignmentElement, AlignmentMode.ALIGN_BY_FEATURES
            );
    alignment.align();

    return alignment.hasSolutions();
  }
}
```

A.1.3 IHitHandler

```
public interface IHitHandler {
    public void progressUpdate(Progress progress);
    public void resultReceived(ScreeningWorkResultData result);
}
```

A.2 Layout Algorithm of the Target View

The layout algorithm of the Target View is implemented in the *TargetMap* class. The entry point for the algorithm is the method: *initClusters()*. The following code snippet shows an extract of the *TargetMap* class.

```java
public class TargetsMap implements GLEventListener {

  private Hashtable<String, Cluster> attribute2cluster;

  private Hashtable<String, ClusterPosition> clusterPositions;

  private int size;

  private double z;

  private int clusterPerRow;

  public void initClusters() {
    int fieldsPerColumn, fieldsPerRow, fieldsPerRowCluster;
    Vector<Cluster> clusters = getSortedClusters(attribute2cluster.
        values());

    fieldsPerColumn=fieldsPerRow=(int) Math.sqrt(size);
    do {
      fieldsPerColumn=++fieldsPerRow;

      fieldsPerRowCluster = (int) (fieldsPerRow / (double) clusters.
          size());
      while(clusters.firstElement().getFieldsAmount() / (double)
          fieldsPerRowCluster > fieldsPerColumn) {
        fieldsPerRowCluster++;
      }

      clusterPerRow = (int) (fieldsPerRow / (double)
          fieldsPerRowCluster);
    }while (!calculateClusterPositions(fieldsPerRow, fieldsPerColumn,
        fieldsPerRowCluster, clusters));

    double fieldWidth = calculateFieldWidth(fieldsPerRowCluster);
    double fieldHeight = calculateFieldHeight(fieldsPerColumn);
    double[] xValues = calculateClusterXValues(fieldWidth,
        fieldsPerRowCluster);
    double[] yValues = initClusterYValues(fieldHeight);
    for (Cluster cluster : clusters) {
      ClusterPosition position = clusterPositions.get(cluster.
          getAttribute().getId());
      double x = xValues[position.getCol()];
      double y = yValues[position.getCol()];
      cluster.initFields(new ClusterData(x,y,z,fieldWidth,fieldHeight,
          fieldsPerRowCluster));
      double clusterHeight = 1+cluster.getFieldsAmount()/(double)
          fieldsPerRowCluster;
      yValues[position.getCol()] -= clusterHeight*fieldHeight;
    }
  }

  private boolean calculateClusterPositions(int fieldsPerRow, int
      fieldsPerColumn,int fieldsPerRowCluster, Vector<Cluster> clusters
      ) {
    clusterPositions = new Hashtable<String, ClusterPosition>();

    int[] columnRemainingHeights = initColumnRemainingHeights(
        clusterPerRow, fieldsPerColumn);
    HashSet<String> placedClusterIds = new HashSet<String>();
    for(int row=0, col=0; col<columnRemainingHeights.length; col++,row
        =0) {
      if(clusters.size()==placedClusterIds.size()) {
        clusterPerRow = col;
        break;
      } else if(columnRemainingHeights[col]<getClusterHeight(clusters.
          lastElement(),fieldsPerRowCluster)) {
        continue;
      }
      for(Cluster cluster : clusters) {
        if(placedClusterIds.contains(cluster.getAttribute().getId()))
          continue;
```

```

            double clusterHeight = getClusterHeight(cluster,
                fieldsPerRowCluster);
            if(columnRemainingHeights[col] >= clusterHeight) {
                clusterPositions.put(cluster.getAttribute().getId(), new
                    ClusterPosition(col,row++));
                placedClusterIds.add(cluster.getAttribute().getId());
                columnRemainingHeights[col] -= clusterHeight;
            }
        }
    }

    if(clusters.size() != placedClusterIds.size())
        return false;
    return true;
}

private double calculateFieldWidth(int fieldsPerRowCluster) {
    return (width/(double)height) * (1/(double) (fieldsPerRowCluster*
        clusterPerRow+clusterPerRow -1));
}

private boolean calculateClusterPositions(int fieldsPerRow, int
    fieldsPerColumn,int fieldsPerRowCluster, Vector<Cluster> clusters
    ) {
    clusterPositions = new Hashtable<String, ClusterPosition>();

    int[] columnRemainingHeights = initColumnRemainingHeights(
        clusterPerRow, fieldsPerColumn);
    HashSet<String> placedClusterIds = new HashSet<String>();
    for(int row=0, col=0; col<columnRemainingHeights.length; col++,row
        =0) {
        if(clusters.size()==placedClusterIds.size()) {
            clusterPerRow = col;
            break;
        } else if(columnRemainingHeights[col]<getClusterHeight(clusters.
            lastElement(),fieldsPerRowCluster)) {
            continue;
```

```
      }
      for(Cluster cluster : clusters) {
        if(placedClusterIds.contains(cluster.getAttribute().getId()))
          continue;

        double clusterHeight = getClusterHeight(cluster,
            fieldsPerRowCluster);
        if(columnRemainingHeights[col] >= clusterHeight) {
          clusterPositions.put(cluster.getAttribute().getId(), new
              ClusterPosition(col,row++));
          placedClusterIds.add(cluster.getAttribute().getId());
          columnRemainingHeights[col] -= clusterHeight;
        }
      }
    }

    if(clusters.size() != placedClusterIds.size())
      return false;
    return true;
  }

  private double getClusterHeight(Cluster cluster, int
      fieldsPerRowCluster) {
    return 1+cluster.getFieldsAmount()/(double)fieldsPerRowCluster; //
        TODO freespace instead of "1+"
  }

  private double[] calculateClusterXValues(double fieldWidth, int
      fieldsPerRowCluster) {
    double[] xValues = new double[clusterPerRow];
    double x=fieldWidth/2d;
    for(int i=0; i<clusterPerRow; i++) {
      xValues[i]=x;
      x+=(1+fieldsPerRowCluster)*fieldWidth;
    }
    return xValues;
  }

  private double calculateFieldHeight(int fieldsPerColumn) {
    return 1/(double)fieldsPerColumn;
  }

  private double[] initClusterYValues(double fieldHeight) {
    double[] yValues = new double[clusterPerRow];
    for(int i=0; i<clusterPerRow; i++) {
      yValues[i]=-fieldHeight/2d;
    }
    return yValues;
  }

  private int[] initColumnRemainingHeights(int clusterPerRow,int
      fieldsPerColumn) {
    int[] columnRemainingHeights = new int[clusterPerRow];
    for (int i = 0; i < clusterPerRow; i++) {
      columnRemainingHeights[i] = fieldsPerColumn;
    }
    return columnRemainingHeights;
  }
}
```

A.3 Import and Export

This section provides the source code of interfaces used for data import and export.

A.3.1 IMoleculesImport

```
public interface IMoleculesImport {
  public Hashtable<String, AptMolecule> loadMolecules() throws
      ImportException;
}
```

A.3.2 IPharmacophoresImport

```
public interface IPharmacophoresImport {
  public Hashtable<String,AptPharmacophore> loadAllPharmacophores()
      throws ImportException;
  public Hashtable<String,AptPharmacophore> loadPharmacophores(
      Collection<AptTarget> targets) throws ImportException;
}
```

A.3.3 ISessionImport

```
public interface ISessionImport {
  public SessionInfo loadSession() throws ImportException;
}
```

A.3.4 ITargetImport

```
public interface ITargetsImport {
  public Hashtable<String,AptTarget> loadTargets() throws
      ImportException;
}
```

A.3.5 IHitExport

```java
public interface IHitExport {
  public void saveHit(AptHit hit) throws ExportException;
}
```

A.3.6 IMoleculeExport

```java
public interface IMoleculeExport {
  public void saveMolecules(Hashtable<String, AptMolecule> mols)
      throws ExportException;
}
```

A.3.7 IPharmacophoresExport

```java
public interface IPharmacophoresExport {
  public void savePharmacophores(Hashtable<String, AptPharmacophore>
      pharmacophores) throws ExportException;
}
```

A.3.8 IProgressExport

```java
public interface IProgressExport {
  public void progressUpdate(Progress progress) throws ExportException
      ;
}
```

A.3.9 ISessionExport

```
public interface ISessionExport {
  public ISessionIdExporter saveSession(Hashtable<String, AptTarget>
      targets,
       Hashtable<String, AptPharmacophore> phas) throws ExportException
          ;

  public IHitExport getHitExport();
}
```

A.3.10 ISessionIdExport

```
public interface ISessionIdExporter {
  public void setId(String sessionId) throws ExportException;
}
```

A.3.11 ITargetsExport

```
public interface ITargetsExport {
  public void saveTargets(Hashtable<String, AptTarget> targets) throws
       ExportException;

}
```

List of Figures

List of Tables

Bibliography

[1] Java SE 5. http://java.sun.com/javase/downloads/index_jdk5.jsp. (accessed on October 13, 2007).

[2] Java SE 6. http://java.sun.com/javase/. (accessed on October 13, 2007).

[3] E. Anderson, G. D. Veith, and D. Weininger. SMILES: A line notation and computerized interpreter for chemical structures. Technical Report EPA/600/M-87/021, U.S. EPA, Environmental Research Laboratory-Duluth, 1987.

[4] Anim8or. http://www.anim8or.com/. (accessed on October 13, 2007).

[5] S. Ash, M. A. Cline, R. W. Homer, T. Hurst, and G. B. Smith. SYBYL Line Notation (SLN): A Versatile Language for Chemical Structure Representation. *Journal of Chemical Information and Computer Sciences*, 37:71–79, 1997.

[6] D. Astle and K. Hawkins. *Beginning OpenGL Game Programming*. Premier Press, 2004.

[7] M.Q. Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for Using Multiple Views in Information Visualization. In *Advanced Visual Interfaces (AVI2000)*, pages 110–119. ACM Press, 2000.

[8] F. Bendix, V. Sladariu, T. Langer, and G. Wolber. Visualizing biological activity profiles using target affinity maps. http://oasys2.confex.com/acs/234nm/techprogram/P1109537.HTM, August 2007. (accessed on October 13, 2007).

[9] H. M. Berman, K. Henrick, and H. Nakamura. Announcing the worldwide Protein Data Bank. *Nature Structural Biology*, 10(12):980, 2003.

[10] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[11] F. C. Bernstein, T. F. Koetzle, G. J. Williams, E. F. Meyer, M. D. Brice, J. R. Rodgers, O. Kennard, T. Shimanouchi, and M. Tasumi. The Protein Data Bank: a computer-based archival file for macromolecular structures. *Journal of Molecular Biology*, 112(3):535–542, 1977.

[12] H.-J. Böhm, G. Klebe, and H. Kubinyi. *Grundlagen der Arzneimittelforschung, Wirkstoffdesign*. Spektrum, 1996.

[13] T. H. T. H. Buu. An Enhanced Scoring Function for Pharmacophore Elucidation in LigandScout and its Application to H1 Antagonists. Master's thesis, Fachhochschule Hagenberg, 2007.

[14] S.K. Card, J.D. Mackinlay, and B. Shneiderman, editors. *Readings in Information Visualization*, chapter 1, pages 1–34. Morgan Kaufman, 1999.

[15] A. Dalby, J. G. Nourse, W. D. Hounshell, and A. K. I. Gushurst ad D. L. Grier. Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited. *Journal of Chemical Information and Computer Sciences*, 32:244–255, 1992.

[16] G. S. Davidson, B. N. Wylie, and K. W. Boyack. Cluster stability and the use of noise in interpretation of clustering. In *INFOVIS 2001*, pages 23–30. IEEE Symposium on Information Visualization, ACM Press, 2001.

[17] Inc. DAYLIGHT Chemical Information Systems. SMILES - A Simplified Chemical Language. http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html. (accessed on October 13, 2007).

[18] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. In *Proceedings of the National Academy of Sciences of the USA*, volume 95, pages 14863–14868, December 1998.

[19] HCE Hierarchical Clustering Explorer. http://www.cs.umd.edu/hcil/multi-cluster/. (accessed on October 13, 2007).

[20] R. Fosner. *OpenGL Programming for Windows 95 and Windows NT*. Addison Wesley, 1996.

[21] A. C. Good, J. S. Mason, and S. D. Pickett. Pharmacophore Pattern Application in Virtual Screening, Library Design and QSAR. In *Virtual Screening for Biactive Molecules*, volume 10, pages 131–159. WILEY-VCH, 2000.

[22] D.L. Gresh, B.E. Rogowitz, R.L. Winslow, D.F. Scollan, and C.K. Yung. WEAVE: A System for Visually Linking 3-D and Statistical Visualizations, Applied to Cardiac Simulation and Measurement Data. In *Proceedings of the 2000 IEEE Conference on Visualization (Vis'2000)*, pages 489–492. IEEE Computer Society Press, 2000.

[23] Chemical Computing Group. Suite 910 - 1010 Sherbrooke St. W, Montreal, Quebec, Canada H3A 2R7. http://www.chemcomp.com/. (accessed on October 13, 2007).

[24] D. Harel and Y. Koren. Clustering Spatial Data Using Random Walks. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 281–286. ACM Press, 2001.

[25] Accelrys Inc. 10188 Telesis Court, Suite 100, San Diedo, CA 92121 USA. http://www.accelrys.com/. (accessed on October 13, 2007).

[26] SciTegic Inc and Accelrys Inc. 10188 Telesis Court, Suite 100, San Diedo, CA 92121 USA. http://www.scitegic.com/. (accessed on October 13, 2007).

[27] A. Inselberg. The Plane with Parallel Coordinates. *Visual Computer*, 1(4):69–91, 1985.

[28] A. Inselberg. *Parallel Coordinates: VISUAL Multidimensional Geometry and its Applications*. Springer, 2007.

[29] Inte:ligand. Clemens Maria Hofbauer-H. 6, A-2344 Maria Enzersdorf Austria. http://www.inteligand.com/. (accessed on October 13, 2007).

[30] S. C. Johnson. Hierarchical Clustering Schemes. *Psychometrika*, 32(3):241–254, September 1967.

[31] D.A. Keim. Visualization techniques for exploring databases. Invited Tutorial, Int. Conference on Knowledge Discovery in Databases (KDD'97), 1997.

[32] M. Kilgard. *OpenGL for the X Window System*. Addison Wesley, 1996.

[33] R. Kosara, G. N. Sahling, and H. Hauser. Linking Scientific and Information Visualization with Interactive 3D Scatterplots. In *Short Communication Papers Proceedings of the 12th International Conference in Central Europe on Computer Graphics, Visualization, and Computer Vision (WSCG)*, pages 133–140. UNION Agency - Science Press, 2004.

[34] T. Langer. Pharmacophore based Parallel Screening. A Tool For Activity Profiling Of Virtual Libraries.

[35] T. Langer and G. Wolber. Pharmacophore definition and 3D searches. *Drug Discovery Today: Technologies*, 1(3):203–207, 2004.

[36] M. Lanzenberger. *The Interactive Stardinates*. PhD thesis, Vienna University of Technology, 2003.

[37] E. Lengyel. *The OpenGL Extensions Guide*. Charles River Media, 2003.

[38] A. Liebig. JXTA Peer-To-Peer Distributed Computing Network for Virtual Screening. Master's thesis, Vienna University of Technology, 2007.

[39] C. North and B. Shneiderman. Snap-together visualization: A user interface for coordinating visualizations via relational schemata. In *Proc. of Advanced Visual Interfaces AVI 2000, Palermo, Italy*, pages 128–135, 2000.

[40] L. Pauling. *The nature of the chemical bond*. Cornell University Press, 1948.

[41] Java Community Process. JSR 231: JavaTM Binding for the OpenGL API. http://www.jcp.org/en/jsr/detail?id=231. (accessed on October 13, 2007).

[42] T.-M. Rhyne, M. Tory, T. Munzner, M. Ward, C. Johnson, and D. H. Laidlaw. Information and Scientific Visualization: Separate but Equal or Happy Together at Last. In *IEEE Visualization 2003*, pages 611–614. Panel - North Caroline University, 2003.

[43] R. J. Rost. *OpenGL Shading Language*. Addison Wesley, 2004.

[44] Schrödinger. 101 SW Main Street, Suite 1300, Portland, OR 97204 USA. http://www.schrodinger.com/. (accessed on October 13, 2007).

[45] D. Schuster, C. Laggner, and T. Langer. Why Drugs Fail - A Study on Side Effects in New Chemical Entities. *Current Pharmaceutical Design*, 11(27):3545–3559, October 2005.

[46] J. Seo and B. Shneiderman. Interactively Exploring Hierarchical Clustering Results. *IEEE Computer*, 35(7):80–86, July 2002.

[47] D. Shreiner. *OpenGL Reference Manual: The Official Reference Document to OpenGL, Version 1.4*. Addison Wesley, 2004.

[48] D. Shreiner, M. Woo, and J. Neider. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2*. Addison Wesley, 2005.

[49] P. T. Spellman, G. Sherlock, M. Zhang, R. Vishwanath, K. Anders, M. B. Eisen, B. Michael, P. O. Brown, D. Botstein, and B. Futcher. Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast Saccharomyces cerevisiae by Microarray Hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, 1998.

[50] T. M. Steindl, D. Schuster, C. Laggner, K. Chuang, R. D. Hoffmann, and T. Langer. Parallel Screening and Activity Profiling with HIV Protease Inhibitor Pharmacophore Models. *ChemInform*, 38(24):563–571, May 2007.

[51] T. M. Steindl, D. Schuster, G. Wolber, C. Laggner, and T. Langer. High-throughput structure-based pharmacophore modelling as a basis for successfull virtual screening. *Journal of Computer-Aided Molecular Design*, 20:703–715, September 2006.

[52] J. L. Sussman, D. Lin, J. Jiang, N. O. Manning, J. Prilusky, O. Ritter, and E. E. Abola. Protein Data Bank (PDB): database of three-dimensional structural information of biological macromolecules. *Acta Cryst*, 54(1):1078–1084, 1998.

[53] R. Todeschini and V. Consonni. *Handbook of Molecular Descriptors*. Wiley-VCH, 2002.

[54] M. Tory and T. Möller. Rethinking Visualization: A High-Level Taxonomy. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'04)*, pages 151–158. IEEE Computer Society Press, 2004.

[55] Tripos. 1699 South Hanley Road, St. Louis, MO 63144-2319 USA. http://www.tripos.com/. (accessed on October 13, 2007).

[56] Robert Voigt. An Extended Scatterplot Matrix and Case Studies in Information Visualization. Master's thesis, Hochschule Magdeburg-Stendal, 2002.

[57] W. Patrick Walters, Matthew T. Stahl, and Mark A. Murcko. Virtual Screening - an overview. *Research Focus*, 3(4):160–178, April 1998.

[58] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers, 2000.

[59] D. Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Computer Sciences*, 28:31–36, 1988.

[60] G. Wolber, A. Dornhofer, and T. Langer. Efficient overlay of small organic molecules using 3D pharmacophores. *Journal of Compututer Aided Molecular Design*, 20(12):773–788, 2007.

[61] G. Wolber, H. Kirchmair, and T. Langer. Structure-Based 3D Pharmacophores: An Alternative to Docking? Plenary talk at the 7th International Conference on Chemical Structures in Noordwijkerhout, 2005.

[62] G. Wolber and R. Kosara. *Pharmacophores and Pharmacophore Searches*, chapter Pharmacophores from Macromolecular Complexes with LigandScout, pages 131–150. Wiley-VCH, 2006.

[63] G. Wolber and T. Langer. CombiGen: A novel software package for the rapid generation of virtual combinatorial libraries. In *Rational approaches to drug design*, pages 390–399. Prous Science, 2000.

[64] G. Wolber and T. Langer. LigandScout: 3-D Pharmacophores Derived from Protein-Bound Ligands and Their Use as Virtual Screening Filters. *Journal of Chemical Information and Modeling*, 45(1):160–169, 2005.

[65] R. S. Wright and B. Lipchak. *OpenGL SuperBible, Third Edition*. Sams Publishing, 2005.