

Artificial Intelligence Driven Drug Affinity Optimization

Robert Warmerdam

346462

BFV4

Supervisor: Alexander Dömling

Contact lecturer: Tsjerk Wassenaar

18-1-2018

Artificial Intelligence Driven Drug Affinity Optimization

Robert Warmerdam

346462

Bioinformatics

Institute for Life Science & Technology

Supervisor: Alexander Dömling

Contact lecturer: Tsjerk Wassenaar

18-1-2018

Foreword

Before you lies the report “Artificial Intelligence Driven Drug Affinity Optimization”. It has been written to full fill the internship requirements of the Bioinformatics bachelor that I attend at the Hanze University Of Applied Sciences. I was engaged in researching and writing this report from September 2018 to January 2019.

This report presents a project provided by the Drug Design research lab, where I undertook an internship under the supervision of prof. dr. A.S.S. (Alexander) Dömling, head of the research group. I would like to thank him for his support, his contagious enthusiasm and the accessible character of the research group.

I would also like to thank my other supervisors for their support during this process and the data that I received from group members.

Robert Warmerdam

Groningen, January 18, 2019

Abstract

Over the last years, there have been several developments in the rapid synthesis and screening of multi-component reaction compounds, which should allow for more efficient drug discovery. Through acoustic dispensing small volumes can be rapidly and precisely screened. However, the large amount of possible combinations of building blocks still poses a challenge. [1] In this report two aspects of this challenge are discussed. The first of these is drug affinity optimization driven by a genetic algorithm (GA) to find high scoring compounds. A new software package with a genetic algorithm, called compound evolver, is presented that uses a fitness function that is primarily based on the potency of compounds, which is represented by the score of various docking programs. The software package delivers a web interface for the interaction with the genetic algorithm that is easy to use for chemists. In addition to the methods employed for building the software package, the genetic algorithm results are also analysed, and parameter tuning is carried out. Various aspects of the genetic algorithm are demonstrated with two protein targets and accompanying multi-component reactions. In the experiments that were carried out it is shown that the genetic algorithm can converge to an optimum on which a large set of GA runs can agree. The second aspect of the project aims to decrease the time needed for analysing results of screening methods used to quantify the amount of product in a reaction mixture. More specifically; a program for automatic analysis and scoring of mass spectra is implemented that shows a more than adequate performance. In conclusion, the use of a genetic algorithm to drive drug affinity optimization is promising, as well as the use of software to perform automatic analysis of mass spectrometry results.

Abbreviations

3D	Three-dimensional
AI	Artificial Intelligence
ANOVA	Analysis of variance
API	Application programming interface
BB	Building block
CLI	Command line interface
EA	Evolutionary Algorithm
ES	evolution strategies
GA	Genetic algorithm
GBB	Groebke-Blackburn-Bienaymé
GLM	Generalized Linear Model
GRIP	Groningen Research Institute of Pharmacy
GUI	Graphical user interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JDK	Java Development Kit
JSP	JavaServer Pages
LE	Ligand efficiency
LLE	Ligand-lipophilic efficiency
MCR	Multi-component reaction
MSE	mean squared error
pdb	Protein Data Bank
PD-L1	programmed death-ligand 1
Q-Q plot	quantile-quantile plot
RMSD	Root-mean-square deviation
SFC	Supercritical fluid chromatography
SMARTS	SMILES arbitrary target specification
SMILES	Simplified molecular-input line-entry system
VBA	Visual Basic for Applications
XML	eXtensible Markup Language

Organisation

This research was carried out at the Drug Design research lab, a part of the Groningen Research Institute of Pharmacy (GRIP) located within the University of Groningen. The University of Groningen is a public research institute in the city of Groningen in the Netherlands. The university was founded in 1614 and is one of the oldest universities in the Netherlands still in operation. The University of Groningen has eleven faculties, one of them being the Faculty of Science and Engineering, the faculty wherein GRIP is located.

Drug Design is a research lab that is primarily focused on the issues in computational chemistry and drug discovery. The group tries to develop druglike molecules through multi-component reactions, as well as tools such as scaffolds to address novel biological targets or software to facilitate the design of drugs. The group of around thirty people is comprised of several PhD students, postdoctoral researchers and lab technicians. Head of the research group is prof. dr. A.S.S. (Alexander) Dömling.

Table of Contents

Foreword.....	2
Abstract.....	3
Abbreviations	4
Organisation.....	5
Table of Contents	6
1. Introduction.....	7
1.1. Theoretical background.....	8
1.2. Goals.....	11
2. Materials and methods.....	12
2.1. Input data.....	12
2.2. Creating molecule libraries and the scoring pipeline	13
2.3. Genetic operators and parameters.....	17
2.4. Multi-reaction evolution.....	19
2.5. Web application.....	20
2.6. Genetic algorithm parameter tuning	21
2.7. Automatic mass spectra analysis	24
2.8. Performance evaluation.....	26
3. Results.....	27
3.1. Genetic algorithm	27
3.2. Web application.....	30
3.3. Genetic algorithm parameter tuning	31
3.4. Filtering effects.....	35
3.5. Average performance over time.....	35
3.6. Automatic mass spectra analysis	36
3.7. Performance evaluation.....	38
4. Conclusion and discussion.....	40
5. References	42
Appendices	45
A. Grid search parameter effects.....	45

1. Introduction

Multi-component reactions (MCR) play an important role in new drug discovery. [2] The main advantages of this technology are ease of automation and the ability to generate high diversity libraries. [2] Recently, there have been several developments in the ability to rapidly synthesize and screen new MCR compounds, which should allow for more efficient drug discovery. Through acoustic dispensing small volumes can be rapidly and precisely screened. However, the large amount of possible combinations of reactants still poses a challenge. [1] In this report two aspects of this challenge are discussed. The first of these focusses on driving the optimization process through artificial intelligence (AI) - the use of hardware or software to approximate human behaviours such as learning and problem solving. This technique is finding its way into more and more applications in daily life as well as in industries, including finance, automotive and healthcare.

Earlier work suggests that a genetic algorithm (GA) could be used to find high scoring molecules on topological structure-activity relationships. [3] Fabian Dey et al. have used a genetic algorithm in combination with a complementary search method for automatic fragment-based design of molecules within a protein binding site of known structure. [4] For this project, the hypothesis is that genetic algorithms can be used as a form of artificial intelligence to find high affinity compounds for a protein target. The population within such a genetic algorithm consist in this context of different chemical components comprised of different building blocks (BB). Prior to applying this procedure to an in vitro synthesis and screening setup, an in silico approach is first carried out, so a computational pipeline or workflow has to be setup.

In this report a new genetic algorithm is presented with a fitness function that is primarily based on the potency of compounds, represented by the score of various docking programs. New generations are produced by the common genetic operators, selection, crossing over and mutation. The choice of which methods to use and how to use them largely affects the success of an algorithm. The lack of diversity for example causes the algorithm to converge on a sub-optimal solution. [5] [6] To prevent this from happening, various parameters have to be optimized after the workflow and the GA is implemented. The parameters to optimize include for example the population or generation size, the selection, crossover and mutation rate, as well as the choice between various manners for creating new offspring and applying mutation. Besides this, a graphical user interface (GUI) also is presented for easy access to this algorithm, its options and its progress.

The second aspect that is discussed focusses on the screening of large amounts of compounds. At this moment, the quality of reactions that are carried out automatically by combining combinations of reactants are assessed by performing supercritical fluid chromatography (SFC), which is a method used to separate, identify and quantify components in a mixture. Mass spectra are currently manually assessed and classified, which takes a significant amount of time. This aspect of the project aims to decrease the time needed by writing a program for automatic analysis and scoring of these mass spectra. Therefore a graphical user interface was built. In addition, this also offers some flexibility for the user in terms parameters to specify.

Reporting results of analysed mass spectra also takes up much time currently. This process also has to be sped up. This means that the analysis should output a set of heatmaps showing the quantity of products both with and without chemical structures.

1.1. Theoretical background

Multi-component reactions

In chemistry a multi-component reaction (MCR) is a chemical reaction where three or more compounds react to form a single product. The reason why they have gained much attention in drug discovery is that they have several advantages over classical chemical synthesis. [7]

Druglikeness

Druglikeness is a qualitative concept used for how druglike a substance is. A traditional method to assess the druglikeness is to check compliance of the Lipinski's Rule of Five, which limits the number of hydrophilic groups, molecular weight and hydrophobicity. [8] Druglikeness can also be achieved by using measurements that incorporate important properties. Examples of these measurements are the ligand efficiency, which is the binding energy per non-hydrogen atom, [9] and the ligand-lipophilic efficiency, which links the potency with the lipophilicity. [10]

Optimization and search algorithms

Optimization problems typically consists of finding the best available outcome of a function by systematically choosing input values from within an allowed set and computing the outcome. Various optimization algorithms exist for solving optimization problems. In computational optimization techniques, iterative methods can be used that converge to a solution.

Grid search

A traditional way of performing optimization of a set of parameters is a grid search, or parameter sweep. This is an method that scores every combination within a manually specified subset of the parameter space. The method suffers from the curse of dimensionality; each additional variation that is added increases the effort needed to try all combinations radically. [11]

Genetic Algorithms

A genetic algorithm belongs to the larger class of evolutionary algorithms (EA), which are generic population based algorithms. These algorithms are inspired by the process of biological evolution and the genetic algorithm especially by natural selection, the process by which forms of life having a better fitness will tend to survive and reproduce in greater numbers than others of their kind. Genetic algorithms are commonly used for solving optimization and search problems, generating solutions with high quality. [12]

Genetic algorithms contain a set of possible candidate solutions. The first population is randomly generated. It is an iterative process in which for each generation the fitness of the candidates is evaluated. Subsequently new offspring is generated from the preceding generation. This is done by use of biology inspired operators such as selection, crossover and mutation for producing offspring. Each candidate solution has along with a phenotype, on which natural selection acts, a set of properties that together can be called the genotype. This genotype is traditionally represented in binary as a sequence of 0s and 1s, but other encodings or real value approaches are also possible. It is the genotype that is passed to offspring.

A genetic algorithm is a metaheuristic which sample a set of solutions too large to be completely sampled, as is the case with the drug affinity optimization that is worked on in this project. Metaheuristics can often find a good solution with much less computational effort. A metaheuristic is however not guaranteed to find the optimal solution as it is when sampling the entire search space. The traditional theory of GAs [13] assumes that genetic algorithms work by discovering

and recombining good building blocks, which builds on the idea that good solutions tend to consist of good building blocks.

Genetic operators

For optimal performance of the genetic algorithm a wide range of parameters and methods for genetic operators have to be analysed and optimized. When optimizing these parameters a balance has to be found between diversity in the population and convergence. Without the correct parameters and balance the genetic algorithm might have low diversity and converge to a local minimum. This procedure is thus not guaranteed to find the optimal solution as it is when sampling the entire chemical space. [5] [6]

Population size

The amount of candidate solutions within a single generation is called the population size. Population size is one important parameter for a genetic algorithm, and good sizes are reported to vary from 30 individuals to more than a hundred. [14] In the following sections the different methods and parameters that can be applied are listed.

Selection

The first genetic operator applied is selection which consist of choosing individuals from a population for producing offspring. This selection operator picks individuals from the population based on their fitness. Various procedures exist for selection, and a parameter that applies to this genetic operator is the number of selected parents for producing offspring. A lower number of parents selected generally means that convergence is quicker.

Truncated selection

The first and simplest selection method that will be discussed in this report is truncated selection. In this procedure only the best individuals of a certain portion from the preceding generation is selected. In practice this method is less used due to the high selection pressure.

Fitness proportionate selection

The second selection procedure is a probabilistic method that makes sure higher scoring solutions are more likely to be selected than lower scoring solutions. The probability of an individual being chosen is proportionate to its fitness value. [15] For example, consider a selection event in which each individuals fitness score is scaled so that the sum of all fitness scores is between 0 and 1. The chance of each individual in this case being picked is equal to the scaled fitness score: an individual with a scaled fitness score of 0.6 will have a probability of being picked equal to 0.6.

Tournament selection

The third selection method is tournament selection, which selects the best individual(s) from tournaments. A tournament consists of randomly selected individuals that participate for victory in the tournament. The participant with the best fitness score is the winner, which will be the picked individual. The amount of individuals that comprise a tournament, called the tournament size, can be varied. In large tournaments, weak individuals have a smaller chance of being selected, because there is a higher chance in large tournaments that a better individual is in the same tournament as weak individuals. [16]

Genetic representation

In evolutionary computational methods the genetic representation is the way of representing the individuals, and on which genetic operators act. In genetic algorithms this is often an array of bits. The genetic representation of an individual is divided into several genes, that can contain possible values of these genes called alleles. Other representation types can also be used and real value approaches are used in evolution strategies (ES). [17]

Crossover

The crossover operator in GAs is a method that combines the genotypes of two selected parents in order to produce new offspring, analogous to sexual reproduction in biology. This operator and the amount of times it is applied, often determined by a user definable parameter, effects the convergence. Crossover generally decreases the amount of generations needed for convergence to occur. With enough diversity present in a GA the use of crossover can speed up the expected optimization time. [18] Various crossover implementations exist, the traditional methods for bit array genetic representations being single, two or k-point crossover. In these methods k crossover points are picked randomly in the parents chromosomes. Subsequently, the bits between crossover points are swapped between parent organisms. In uniform crossover, each bit of the chromosome that will be passed on to offspring is independently chosen from the two parents.

Mutation

In genetic algorithms the mutation operator can be implemented in several ways. Bit string mutation introduces variations in the genome by inducing the flipping of a bit at a random position in the genome. This method is thus only applicable to binary string genetic representations. Other methods that can be applied to integer and floating-point values include replacing genes with either a lower or upper bound randomly or replacing genes with another random value within the user-specified limits.

Termination

Termination of a GA can for instance be set to occur after a specific amount of time or number of generations is reached. In addition, the termination of a genetic algorithm is often guided by the level of convergence in the evolution process. This can be determined by manual inspection of the evolution progress, or by a mathematical method.

Fitness function

GAs use an objective fitness function to guide evolution towards an optimal solution. Such a fitness function determines the fitness of a given candidate solution, which is then used to select those candidates that are most likely to aid prosperous evolution. A fitness function has to be well designed on a number of levels. For instance, the fitness function has to correlate with the goal of the optimization problem, and must ideally also be computed quickly, to allow for many iterations in order to produce a usable result.

Mass spectrometry data

Mass spectrometry is a technique that is used for measuring the mass-to-charge ratio of ions. At the moment there are various proprietary data formats in existence for handling data produced by mass spectrometers, which makes it difficult to directly manipulate the data. To solve this problem, several open, public, data formats have recently been developed. These are mostly based on the eXtensible Markup Language (XML) format. One of these formats is mzXML, [19] which is currently in use by the proteomics community. After the various formats where developed, a new

challenge had to be overcome. Which is reading and writing these open formats. For this the ProteoWizard software suite has been designed. [20]

1.2. Goals

The goal that is set for this project is examining the hypothesis that genetic algorithms can be used as a form of artificial intelligence to find high affinity compounds for a protein target, which comprises implementing and optimizing of a GA. The secondary goal is to speed up the analysis of mass spectra by implementing an automated system.

2. Materials and methods

Building a genetic algorithm for finding high quality compounds in a large amount of possible combinations of building blocks, or reactants, is a task that allows for a number of different approaches. For this task a new Java software package, Compound evolver, [21] was written. The implementation of this software package is divisible into multiple parts. In the sections 2.2- 2.4 the methodology of creating the software package is explained, as well as data and software that has been used. Subsequently, the methods for parameter tuning are described in section 2.5. The methodology of creating the automatic mass spectrometry analysis software is discussed in 2.7.

2.1. Input data

Building a genetic algorithm for finding high quality compounds in a large amount of possible combinations of building blocks, or reactants, is divisible into multiple parts. A large part of the procedure however depends heavily on the type of input data which is supplied. input data, which is discussed in this section.

Fragment libraries

The variables, or building blocks, in this search problem were determined to be the reactants primarily. These are the fragments that are to be joined into a molecule, creating a candidate solution. This approach is greatly complemented by especially multi-component reactions. This is because their properties allows MCRs to be generally applicable to a wide range of starting materials, allowing for the generation of relatively large libraries. In Table 1 an example of a reaction scheme is presented, with an accompanying set of fragments for each reactant in the reaction. Only three different variations are shown for each group of reactants, while in practice the number of variations of different reactants could, in theory, be any limited number, and the number of different reactants in the reaction could also be much higher. The fragment libraries thus describe the possible building blocks of a candidate solution. These building blocks have to be represented in a genotype for the application of the various genetic operators.

To allow fragment libraries to be imported in the genetic algorithm a data format was chosen that can hold a collection of different chemical structures. The chosen format consists structures written according to the Simplified molecular-input line-entry system (SMILES) specification, separated with linebreaks. SMILES are a linguistic construct that denotes a molecular structure as a graph with optional chiral indications. [22]

Table 1

Reaction scheme of a Groebke-Blackburn-Bienaymé multi-component reaction. For each reactant three possible fragments are shown. There are 27 combinations possible for these fragments in this reaction scheme.

2-aminopyridine derivatives	aldehydes	isocyanides	
2-aminopyridine	aldehyde	isocyanide	product

2.2. Creating molecule libraries and the scoring pipeline

The previous section describes that the building blocks for candidate solutions come from libraries of small fragments that can be joined into a molecule. For the fitness function however the potency has to be calculated from a three-dimensional (3D) compound docked in a protein pocket. Therefore, a pipeline was implemented that converts fragments from fragment libraries, into 3D compounds that are ready for docking. In this section is described how an existing protocol for performing the conversion was used as a guideline for the setup of the implemented pipeline.

The automation of the drug affinity scoring protocol currently in operation is a task that allows a number of different approaches. Various options were considered for combining these into a GA.

As repeatedly performing a docking step would require much computational power, it was decided that a web server written in Java, that can handle the computationally demanding processes, with accompanying graphical user interface, would be a good solution. In addition, this would provide a user friendly experience. For the automation of the mentioned protocol, a couple of adaptations had to be made. The existing protocol in use relied on multiple command-line or GUI programs. The primary limitation that forces certain adaptations is that a tool or program requires an application programming interface or at minimum a command line version, because simulating clicks and keyboard input for a program with strictly a graphical user interface is not the most appealing in terms of performance for instance. Although parts of the protocol were changed, the input of the workflow remains the same, which means the program requires besides a set of building blocks for the reactants in a reaction, a reaction scheme, a reference fragment or anchor to guide compounds to the right location, and finally the target protein. All programs that were considered for implementation are listed in Table 2 divided by the pipeline steps and distinguished by the three main properties that were taken into consideration.

Creation of two-dimensional libraries of compound structures

For the genetic algorithm a generation exists of candidates that can be scored, and used for creating new generations. The candidates in this genetic algorithm is set to be a compound, which could be any combination of the input reactants. The score of the candidates however is entirely dependent on its three-dimensional structure. The genetic algorithm has to be able to process the combinations of reactants to obtain the three-dimensional structures. The first step in this process is combining the reactants to form single structures. The generation of libraries is in the existing protocol done through ChemAxon's [23] Reactor software or a Java command-line program that makes use of the same software via an application programming interface (API).

The Reactor program searches superstructures of the reactants specified in the reaction scheme within the list of reactants. Subsequently it tries to connect the reactants according to the part user-specified, part automatic reactant to product atom mapping, to produce chemically feasible products. The atom mapping specifies where atoms in the product originate from. For joining the selected building blocks in the genetic algorithm this software was also chosen.

Generation of three-dimensional conformers

The two-dimensional structures that arise after combining the reactants have to be converted into three-dimensional structures. Because a single structure can often adopt a plethora of multiple conformations, and thus not one conformer can properly represent the compound, multiple conformers have to be generated. The conversion from two-dimensional compounds to three-dimensional compounds was in the manual procedure done with Moloc, [24] which can create a plethora of conformers. This however can also be done with a Java API ChemAxon [23] provides, that has proven to be capable of providing useful 3D structures. [25] Because of its more accessible character the conformers are generated by the ChemAxon Conformer Plugin, which uses a Dreiding force field. Conformer generation has been setup to run with a user-specified maximum amount of conformers. It was also considered to calculate an amount of conformers from the amount of rotatable bonds, but as other factors also have to be considered for such a calculation, like the restrictiveness of the protein, this solution was not implemented. Instead, the user's experience and knowledge about the specific experiment together is most likely able to provide a more realistic number for the necessary amount of three-dimensional conformers.

Placement of conformers in receptor

The conformers that are produced should be placed in the receptor. This is done by aligning it to a reference fragment. Other approaches exist that aim to find the correct location itself, but this is proven to be performing less ideally, quality wise and in computation time. Besides this, the goal in this project was to tackle optimization in a receptor with known pharmacophoric features. In the manual protocol the Open Babel [26] tool Obfit is used to do this step. This is a command-line program, which requires besides the compound itself, the reference fragment and an in the SMARTS defined substructure of the reference fragment to which the compound should be fixed. Because the ChemAxon Alignment class can also produce convincing results without a perfect match between structures, and because of its accessible API, this software is incorporated into the GA next to the Obfit program, to be the primary alignment method.

Energy minimization

Before acquiring a score for a compound it first has to be optimized for minimal energy. There are multiple tools currently in use for energy minimization. Moloc [24] is one of the tools that can be used and AutoDock Vina [27], and Smina [28] were also available for energy minimization. As the result of the docking tool can greatly affect the performance of the genetic algorithm, a solution was required that makes it possible to interface between at least Moloc and Smina. The option was implemented that lets the user select the preferred energy minimization program. Both of the docking programs require a protein in a format other than the regular Protein Data Bank (pdb) file format. For Moloc this file is called an environment entry, for which a conversion tool is available in the Moloc software suite. In the pipeline, functionality is implemented that runs the conversion tool. Meanwhile Smina requires a PDBQT file that can be converted in the command line with the 'prepare_receptor4.py' tool delivered with AutoDockTools.

Table 2

An overview of various methods and tools that are considered for implementation in the scoring pipeline. The methods are distinguished by three properties. ChemAxon provides libraries with Java classes accessible by an API providing excellent programmatic accessibility. Open Babel provides access to its OBAAlign C++ class in Java and should also provide good programmatic accessibility, while the remaining programs only provide a command line program giving them poor to medium programmatic accessibility.

Pipeline step	Tool/Method	Performance	Speed	Programmatic Accessibility
3D library creation				
Conformer generation	ChemAxon Reactor			++
	ChemAxon Conformer Plugin	/		++
	Moloc: Mcnf	+		-
Fixation of compounds to anchor	Obfit	-		-
	Open Babel OBAAlign	-		+
	ChemAxon Alignment	+		++
Energy minimization/scoring	Moloc: Mol3d		-	-
	AutoDock Vina		-	--
	Smina		++	-
++ excellent, + good, / medium, - poor, -- very poor				

Lipinski rules

Prior to inserting candidates in the population, the candidate is checked for invalid amounts of hydrogen bond donors and acceptors, molecular mass and partition coefficient. The hydrogen bond donors and acceptors are calculated by use of ChemAxon's HBDAPPlugin class which was set to exclude sulphur-hydrogen, and halogen-hydrogen bonds, to only count the oxygen-hydrogen and nitrogen-hydrogen bonds. The octanol-water partition coefficient is calculated by ChemAxon's logPPPlugin class.

Anchor deviation

After minimization is performed the inspected results seemed to occasionally suffer from aggressive minimization: the anchor-matching-substructure in the compound has drifted away a significant amount from the position of the anchor. As this was not desirable, functionality was implemented for calculating the root-mean-square deviation (RMSD) of atomic positions between the anchor-matching-substructure in the compound and the anchor itself. This value is used to implement a filter.

Exclusive shape

Minimization would also occasionally fail to move all atoms that were clashing with the receptor. This is undesirable as well which means that functionality was required that would remove candidate conformers that suffer from clashing atoms. This functionality was implemented according to Algorithm 1, which is an adaptation of the exclusive shape functionality from the online virtual screening software called Pharmit. [29] The effects of both the anchor deviation and exclusive shape filtering steps were determined to give an impression about the quantity of candidates affected.

Algorithm 1

The exclusive shape algorithm presented here describes a 3D grid that represents the shape of a solvent excluded surface.

Exclusive shape	
1	<i>resolution</i> = 0.5, <i>tolerance</i> = 0, <i>probe atom</i> = H ₂ O (Van der Waals radius = 1.4)
2	<i>grid</i> = new three-dimensional array, with size: receptor-size / <i>resolution</i>
3	for <i>atom</i> in <i>receptor</i> do :
4	mark a sphere with the Van der Waals radius of the <i>atom</i> + <i>probe atom</i> at the location of the exposed point as occupied.
5	for <i>exposed points</i> in <i>grid</i> do :
6	mark a sphere with the Van der Waals radius of the <i>probe atom</i> at the location of the exposed point as not occupied.
7	remove or add an outer layer in the <i>grid</i> depending on <i>tolerance</i>

Fitness score

The fitness function for the GA is primarily based on the potency of the compounds, which is determined by the docking programs that carry out energy minimization. The docking programs were run, and are implemented to run, with default parameters mostly. Exceptions are the minimize option which is used with Smina, and the worry parameter that is set to 0.01 in Moloc's Mol3d tool. The search space for Smina is determined by calculating the size and centre of a box around the ligand in question, with additional margin added.

The value returned by the minimization and scoring tools was binding affinity (kcal/mole) for Smina while the returned value by Moloc's Mol3d tool is on an arbitrary scale. While these raw scores are different from each other, neither of them take the molecular mass into account or the lipophilicity. The ligand efficiency (LE) measure takes the amount of heavy – non-hydrogen - atoms into account. The equation is as follows:

$$LE = (\Delta G)/N$$

Where LE is the ligand efficiency, ΔG the binding energy, and N the amount of heavy atoms. The ligand-lipophilicity efficiency (LLE) measure takes the lipophilicity into account in an attempt to estimate druglikeness. The traditional formula for determining the LLE requires knowledge about the half maximal inhibitory concentration (IC_{50}) of the compound obtained either from experimental analysis or from equations that are only applicable to certain interactions. An alternative equation uses the logarithm of the ratio of binding energy (ΔG) and partition coefficient (P):

$$LLE = \log\left(\frac{-\Delta G}{P}\right)$$

Here LLE is the ligand-lipophilicity efficiency. Sadly the binding affinity and the binding energy are not interchangeable, but they do relate with each other, thus this equation is implemented. In addition, the aim of the final fitness score is not to be an absolute value that is as accurate as possible, but rather be an indication of what compound within a large group has more potential to perform well relative to the others.

2.3. Genetic operators and parameters

Genetic representation

As mentioned earlier the genetic representation of a candidate solution is often a sequence of bits. In this case however, the more logical solution to use is a real-valued approach, because simple encoding types are likely to result in invalid molecules after mutation or crossing over. While a bit string encoding type replicates biological evolution rather well, it is not desirable to have unnecessary slow evolution. This means that the data which is passed on to offspring is just a regular set of reactants. Considering the following vectors of reactants **a**, **b** and **c**.

$$\mathbf{a} = [a_1 \ a_2 \ \dots \ a_n]$$

$$\mathbf{b} = [b_1 \ b_2 \ \dots \ b_n]$$

$$\mathbf{c} = [c_1 \ c_2 \ \dots \ c_n]$$

The genetic representation and resulting phenotype of a candidate solution are the following:

$$\mathbf{genotype} = [a_i \ b_i \ c_i]$$

$$phenotype = a_i + b_i + c_i$$

Choosing for this approach does however have an effect on how the crossover and mutation operators can be designed and implemented. These are discussed in the following subsections.

Selection and offspring generation

As described in the Theoretical background, selection picks individual solutions based on their fitness. The procedures truncated selection, fitness proportionate selection and tournament selection, also described in the section Theoretical background, is implemented. A user can thus choose and experiment with these methods. For compatibility with the fitness proportionate selection, the relative probability of a dependent on the normalized fitness score:

$$normfitness_i = (fitness_i - minfitness)/(maxfitness - minfitness)$$

How many parents are selected for a new generation is established as a fraction of the size of the preceding generation. For example, with a selection size of 0.1 and a generation size of 100, 10 parents will be selected and used for creating new offspring. There are various methods for creating new offspring. The Crossover section in **Error! Reference source not found.** describes the traditional method for performing crossover, which is what is applied to binary genetic representations. For this genetic representation however a form of uniform crossover is implemented, which randomly selects for each gene in the genome from what parent the gene is passed to a child. For instance consider the following genotypes **p1** and **p2**:

$$\mathbf{p1} = [a_1 \ b_5 \ c_6]$$

$$\mathbf{p2} = [a_2 \ b_4 \ c_3]$$

$$\mathbf{o} = [(a_1|b_2) \ (b_5|b_4) \ (c_6|c_3)]$$

There are 2^n possibilities for offspring **o** in crossover, where n is the amount of genes in the genotype.

Besides crossover, the elitist strategy and introducing random immigrants are also implemented methods for generating new offspring. To make these methods available for use in a single run of the genetic algorithm a probabilistic procedure is implemented that is based weights as specified by the user. For every child that needs to be created the algorithm picks one of the three methods by a weighted random choice algorithm, presented in Algorithm 2.

Algorithm 2

The weighted random choice algorithm selects a weight based on its relative probability, corresponding to the weight.

Weighted random choice

```

1  sumOfWeights = sum of weights
2  randomValue = random value between 0 and sumOfWeights
3  for  $i = 0$  to length of weights step 1 do
4      randomValue -= weightsi
5      If randomValue < 0 do return  $i$ 
```

Mutation

Because the traditional form of mutation described in Theoretical background is not applicable to our encoding type, a form of uniform mutation is implemented. In this procedure a selected part of the genome is mutated to another random available value at that position. This means that a random reactant will be selected. However, to approximate the traditional approach a distance, or similarity dependent method was also incorporated, which makes sure that reactants very similar to the initial reactant are more likely to be chosen for replacement. This is done by

calculating the Tanimoto dissimilarity scores from the chemical fingerprint for possible replacements of a reactant. The Tanimoto dissimilarity, synonymous to the Jaccard index, is a statistic for similarity or difference between sample sets. [30] [31] Previous work has shown that such a similarity dependent mutation can improve the average fitness score more quickly than the mutation independent of similarity. [3] The user-definable mutation rate represents the probability that a single gene, or reactant, is replaced by another one. This is usually set quite low as too high values will turn the genetic algorithm in a random search, which will not show convergence.

2.4. Multi-reaction evolution

Introducing more intricate reactions which should behave differently based on which atom is present at a specific location (for example the condition that a certain reactant can only be incorporated in the final product by means of esterification) requires a different approach than using a single reaction scheme that can be interpreted by Reactor. The most robust solution was introducing more reaction schemes to be used in evolution, which has a number of advantages. For instance, this solution also could support applying selection pressure on the reaction scheme itself.

Table 3

Comparison of six considered possible implementations for guiding multiple reaction schemes in evolution. The methods can be described as follows: (1) Reaction schemes are drawn in sequence. (2) Reaction schemes are independent and have their own reactants. (3) Reaction schemes use reactants from the same set whether these schemes are independent or not.

Method	Interspecies selection	Interspecies recombination	Separate reaction schemes possible	Reaction determination
1	+	+	-	By reactants
2a	-	-	+	Fixed
2b	+	-	+	Fixed
3a	+	+	+	By reactants
3b	+	+	+	Fixed
3c	+	+	+	By separate gene
+ yes, - no				

Setup of multi-reaction evolution

While multiple reaction schemes in this evolution strategy allows for a great increase in possibilities, the exact setup of this concept can have a large impact. A comparison is made in Table 3

The first method considered that would allow different reaction products to be produced dependent on the reactants that were supplied required an initial reaction scheme (1) that could produce an initial product regardless of the specified reactants. Additional reaction schemes would include the product of the initial reaction with the requirements for this second reaction (for example an oxygen atom where in the previous reaction this could be a plethora of possible atoms). When running these reaction in sequence only those reactants that comply with the rules set by reaction scheme 2 will react. An advantage of this setup is that the same pool of reactants is used which would not require changes to the genetic representation and the crossover methods. A big disadvantage is that this does not allow introducing reaction schemes that are different to begin with and it is thus not very flexible.

A second method is rather different and would use different reaction schemes not correlated with each other and with their own reactants. This gives more flexibility, but it either does not support crossover between candidates with different reaction schemes, which is desired in case of intricate conditional concatenation reaction schemes. In this case the evolution will essentially drive two different populations (2a), with different reaction schemes, or will likely make one of the populations go extinct early on in evolution in favour of another population (2b), dependent on the manner that selection is carried out.

The third method combines the best of the first two methods in that it can be combined with the existing crossover method and that less correlating reaction schemes can be used as well as very similar ones. In this solution the reactants that are used in multiple reaction schemes are specified. The resulting genotype of candidates contain a value for every reactant that is present in the pool of reaction schemes, which makes sure that the chromosomes of different candidates is always the same length. In addition all reactions can be applied to every candidate, allowing for an intuitive crossing over implementation.

The reaction scheme applied to a specific candidate can be determined dynamically, by trying the different reaction schemes available, and picking one that works (3a). This however requires the user to define their reaction in a very strict manner. For instance by using SMARTS in conjunction with logical operators. The reaction scheme can also be fixed for a candidate (3b), which means that reaction schemes are used in equal amounts in the initial generation, and that random Immigrants will be assigned a scheme at random as well as offspring produced by crossover. Adding another gene to the genotype is also a possibility (3c), but this is not logical in the two main applications.

Despite this method being very flexible and powerful, performing crossover between candidates with two very different reaction schemes is not always desired because this is likely to introduce alleles from candidates that do not use the corresponding gene in candidates that do use the gene. In addition, the same set of reactants might serve different purposes within different reaction schemes. To solve this a setting is introduced that will guide crossover between different species. This can be set to always perform crossover, which is logical in case the reaction scheme chosen is dynamic. The second option will find and only perform crossover on those genes that are used by both candidates selected for crossover. The third and last option restricts this specific case of crossover.

2.5. Web application

A web application was built around the genetic algorithm for user friendly access to the system. The back-end implementation of this web application was done with the Java programming language by using the Java Development Kit (JDK) version 1.8.0_181. The Java Platform, Enterprise Edition was used for its servlets and JavaServer Pages (JSP) technologies. The ChemAxon JChem-main API version 17.29.0 was used to get access to the various classes for molecules, importing structures, Reactor and calculator plugins. [23] For the front-end of the web application Hypertext Markup Language (HTML) was used for creating a document structure, Cascading Style Sheets were used for styling and JavaScript to introduce interactivity. The front-end was designed around the AngularJS framework (version 1.6.7) that facilitates exceptional form validation and a single-page application setup. The front-end framework Bootstrap 4.1.3 was also used for consistent and flexible interface. More JavaScript libraries were used like JQuery (version 1.12.1) and Chart.js (version 2.7.2) for visualization of the genetic algorithm.

2.6. Genetic algorithm parameter tuning

Some time ago the evolutionary computation practice was that parameter values were chosen based on ad hoc choices. [32] [3] For the described genetic algorithm a parameter tuning procedure has been set up to semi-automatically explore the parameter space and find well performing combinations. Parameter tuning of a genetic algorithm is essentially a meta optimization problem. Multiple approaches are known for parameter tuning of a genetic algorithm. These include algorithms like a meta-GA, hill climbing algorithm, grid search and random search. [33] Because the objective regarding parameter tuning is not only focused on finding the optimal solution, but also exploration of the parameter space and study the influence of changes in parameters, a grid search approach was chosen. For the genetic algorithm implemented, the parameters listed in Table 4 were expected to affect the behaviour in case of a single reaction scheme. This table makes a clear distinction between ‘quantitative’ (an order can be observed in the parameter values) and ‘qualitative’ parameters (no observed order). The advantage of quantitative parameters is that heuristic search methods can be used for optimization. For qualitative parameters, the only solution is sampling every option.

Table 4

The main parameters that affect the behaviour of the genetic algorithm. A quantitative parameter type refers to the fact that parameter values are at least partially ordered, which can thus be used in regression, opposed to categorical. The qualitative, or categorical parameter values restrict the use of regression or gradient based optimization methods for these parameters, because there often is not an order in the values.

Parameter	Type
population size	quantitative
crossover rate	quantitative
random immigrant rate	quantitative
elitist rate	quantitative
mutation rate	quantitative
selection rate	quantitative
selection method	qualitative
mutation method	qualitative

The parameters that guide the multi-reaction runs also alter the behaviour, but the choice between these options should largely be guided by the characteristics of the specific experiment.

Conventions and standards

The population size in genetic algorithms needs to be large enough to initialize with a rich set of solutions, which suggests that there is a relationship between the population size, the number of dimensions in the problem, and the optimization results. Earlier work recommends a population size of 10 times the number of dimensions, [34] while another genetic algorithm with 3 varying chemical structures uses a population size of 300. [3] It is reported that a mutation rate should be set so that in every candidate a single mutation is introduced. [35] Out of the implemented selection schemes the tournament selection is suggested to perform the best, [36] which trial and error also suggested.

Parameters to optimize

The behavioural parameters of an optimizer can be varied to run a set of experiments. Since a balance between diversity and convergence can only be achieved with the right combination of parameters, it is important that all combinations of values for parameters are evaluated. In addition, as a genetic algorithm is not deterministic, it is required to do multiple repetitions in

selecting the correct parameters. It is advised to run at least ten to thirty experiments as lower amounts as it can be difficult to show a statistical significant differences between groups. [37] These prerequisites result in a large number of experiments which have to be run. When only a few behavioural parameters are present this is computationally feasible. However, this genetic algorithm arguably has a lot of behavioural parameters. This means that parameters and possible values have to be carefully chosen.

Initial experiments, driven by trial and error, were performed to obtain a general idea where a global optimum might reside. These initial experiments were run with various use cases. For the initial experiments, from the parameters shown in Table 4, the parameters that are chosen to vary in the initial optimization procedure are the population size, mutation method, mutation rate and the selection rate.

Parameter vectors

In the first parameter grid search, the following parameter vectors are assessed, which were based on initial testing of the genetic algorithm, as well as the findings that are reported in other research.

Table 5

The parameter values of which all combinations are evaluated in the grid search experiment. The parameter values are based on standards found in literature and initial tests that were manually inspected.

Parameter	Values
Generation size	150, 200, 250
Mutation method	Similarity dependent, Similarity independent
Mutation rate	0.1, 0.2, 0.3
Crossover rate	0.6, 0.8, 1.0*
Elitism rate	0.0, 0.1, 0.2, 0.3, 0.4*
Random immigrant rate	0.0, 0.1*

* The values of crossover rate, elitism rate and random immigrant rate are combined in a manner such that their sum is always equal to 1.0.

Setup

The parameter tuning grid search was performed with a maximum duration, so runs with a larger population size will run with a similar duration, but with less generations, because providing more computing power to runs with a larger population gives a clear advantage to these runs. All experimental computations were performed using an Ubuntu 16.04.5, 64-bits based machine with two 'Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz' processors, 125 GiB Memory (RAM) and OpenJDK version "1.8.0_181". The maximum duration of each run was determined to be twenty minutes, which allows runs, even with large population sizes, to evolve in more than thirty generations, while not exceeding practical computation time limits for a single experiment. For every optimization run Smina was used to calculate the binding affinities for every compound.

For the first grid search setup the putative menin tumor suppressor protein was used, which is associated with a syndrome known as multiple endocrine neoplasia type 1. The protein is mostly found in the nucleus and inhibits transcriptional activation by proto-oncogene JunD. [38] For the reaction a Groebke-Blackburn-Bienaymé (GBB) multi-component reaction was used, which is shown in Table 1. For this reaction 40 2-aminopyridines, 73 aldehydes and 74 isocyanides were used, creating a theoretical search space of 216.080 possible compounds. Figure 1 shows a section of the protein with the anchor molecule that is used for aligning candidate compounds.

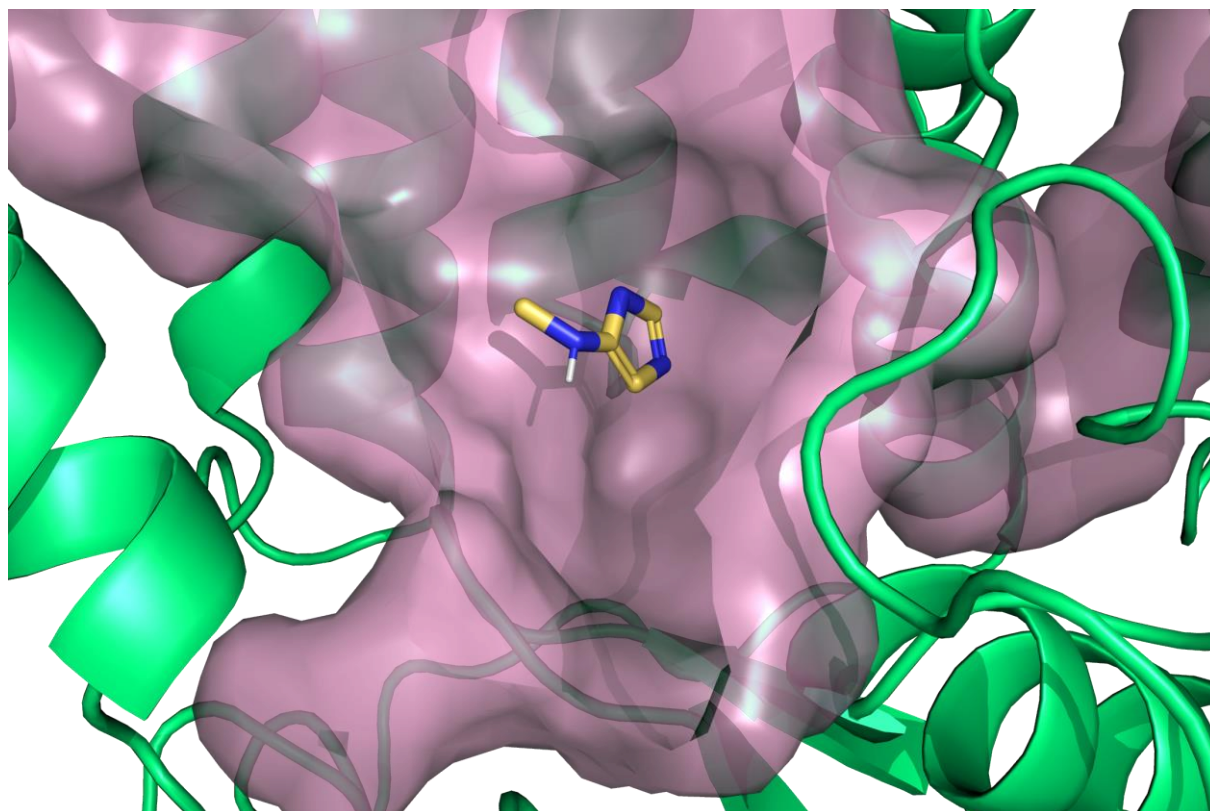


Figure 1
A section of the menin protein with the anchor, or reference fragment that is used in the grid search experiment. Rendered with PyMOL. [39]

Analysis

A generalized linear model (GLM) was used for analysis the grid search experiments. This was chosen because such a model handles continuous variables well in the form of regression. The average of the best-of-run individuals was used as a measure for the performance of the GA.

Other test cases

To get an insight on the effects of the filtering steps to remove clashing conformers for instance. An experiment was performed that uses a more restricting protein. The protein of choice was the programmed death-ligand 1 (PD-L1) protein that is expressed by for example T-cells and B-cells and various types of tumor cells. The protein is a transmembrane protein and interactions with its receptor inhibits T-cell activation and cytokine production. [40] A section of the protein is displayed in Figure 2. An unpublished multi-component reaction is used with this experiments.

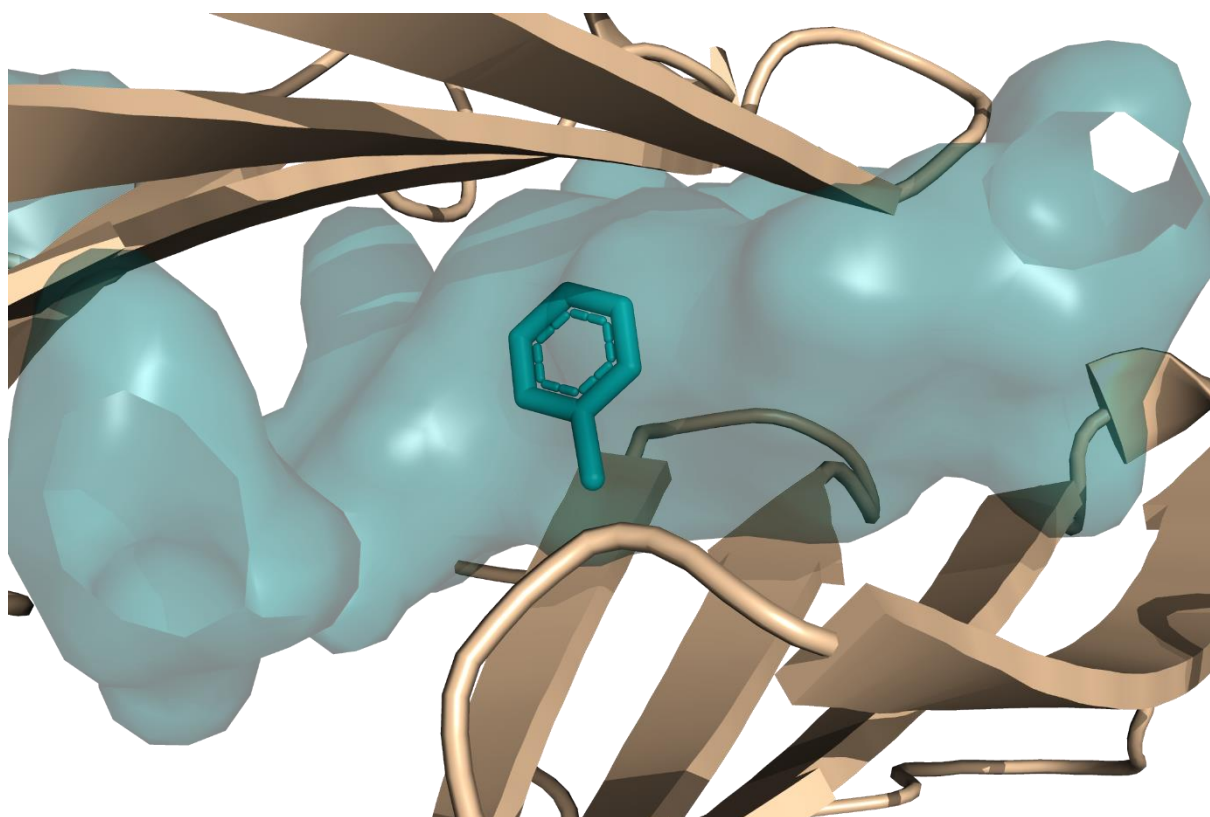


Figure 2

A section of Programmed death-ligand 1 with an anchor, or reference fragment, in one of its cavities. These molecules were used in experiments with the aim to get an insight on what effect removing clashing conformers has. Rendered with PyMOL. [39]

2.7. Automatic mass spectra analysis

Another aspect of AI driven drug discovery is the screening a large amounts of compounds. At this moment the quality of reactions that are carried out automatically by combining combinations of reactants are assessed by performing supercritical fluid chromatography. Combining different reactants is performed rapidly by the use a system that automatically transfers miniature amounts of reactants between a source plate well and a destination plate well. To accelerate the analysis of mass spectra an automatic system for the analysis of the mass spectra was designed and implemented.

Manual inspection of mass spectra

The plates in which reactants are combined are 24 by 16 wells in size and spectrum from each of the wells inside is currently manually inspected to give them a score. The main features that are considered when inspecting a mass spectrum is the height of the peaks that corresponds to the theoretic product compared to the height of the other peaks. If one of these peaks is the highest in the spectrum (the base peak), the well is classified as a well with high amount of product. In addition, if a higher peak exists in the spectra, but has a mass-to-charge ratio below a certain value, the well is also classified as a well with a high amount of product. The well is classified as a well containing very little product if all corresponding peaks are low enough to be almost unnoticeable in between the other peaks. When the peaks corresponding to the theoretic product complies with neither of the two mentioned conditions, the well is classified as a well containing a medium amount of product. The peaks that correspond to the theoretical product are defined as ion adducts, which are ions formed by direct addition of an ion to a molecule. Commonly, the product

is protonated, in which case a corresponding peak should be located at the mass-to-charge ratio that corresponds to the molecular mass of the product in addition to the mass of a hydrogen atom. The retention time range in which the peaks were included in the spectra was chosen by selecting the broad hill in the chromatogram. The chromatogram in this case shows a two-dimensional view of the signal plotted against the time.

The mass spectrometry data format that was chosen is .mzXML because working with this format turned out to be relatively straightforward in Python 3.6, [41] the programming language that was used. This was chosen for its large amount of useful libraries developed by the community for both chemical and data analysis.

```
A1-SAMPLE.mzXML
<scan num="2"
      centroided="0"
      msLevel="0"
      peaksCount="80"
      polarity="+"
      retentionTime="PT0.099999S"
      basePeakMz="0.0"
      basePeakIntensity="0.0"
      totIonCurrent="-5798.0"
      msInstrumentID="1">
  <peaks compressionType="zlib"
        compressedLen="363"
        precision="64"
        byteOrder="network"
        contentType="m/z-int">eJxdkM1NA</peaks>
</scan>
```

Figure 3
A simplified mzXML 'scan' element that describes a spectrum at a specific time point in the mass spectrometry experiment.

Implementation

The first step in processing the mzXML formatted files is parsing it into a queryable data structure. This functionality was implemented by use of the python XML parser module. Not all scan elements were considered to be informative. These uninformative scans, labelled with an `msLevel` of 0 are removed in order to decrease the time required for the program to run. The resulting scan elements contains among other properties, the retention time, which indicates at what point in the run the scan originates from. In addition, all the main results of the scan are located in the peaks element. The main content of the peaks element is encoded and compressed, which is converted to a regular collection of pairs which consist of a mass-to-charge ratio with corresponding intensity.

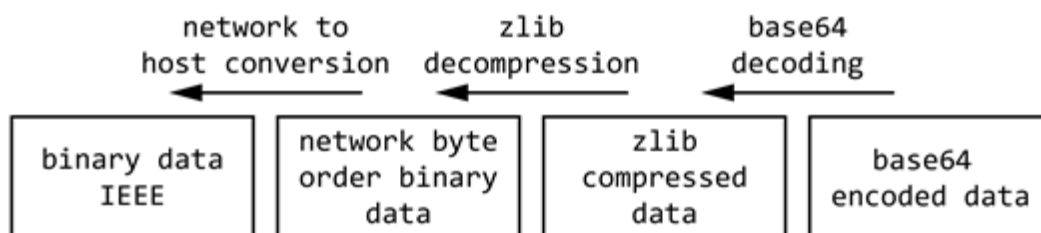


Figure 4

The data flow from the peaks data in mzXML files to a collection of mass-to-charge ratios and corresponding intensities. The Zlib compression is not necessarily an included step in the reverse direction.

Because the MassLynx software package [42] presents these mass spectra's mass-to-charge ratios only as integer values, the subsequent steps in processing the peak values were implemented to also produce integer values. It was implemented that the unrounded values are all collected in their corresponding bin, which are intervals of size 1 around an integer value, summing the intensity values and rounding the mass-to-charge ratios. The more sophisticated way to do this is by using kernel density estimation. This is used to create variable bin sizes, by creating kernels from each data point which are then combined to create a density estimate. The implemented, more simple method for combining different scan elements, was also recently applied in a publication by Jaman et al. [43] Besides this the resulting spectra are also very similar to the spectra processed by the waters MassLynx software package. Outliers within the peak values are removed by the program if they expand further than 5000 times the inter quantile range, which was chosen by trial and error.

Scoring methods

The mass spectra form the basis of the automatic procedure. The next step is actually determining a score for a given mass spectra. The program first compares the peaks of all possible adduct masses for the specific compound, out of which the highest peak is chosen. After that the score is calculated and reported. The reported score is currently calculated by using the traditional approach, which is using the base peak as a reference for the maximum intensity. More sophisticated approaches like using the sum of intensities for another normalization step or applying an outlier scoring method that considers the distribution of peaks were explored, but they did not show an improvement.

2.8. Performance evaluation

The performance of the automatic mass spectrometry analysis procedure was evaluated by calculating the mean squared error (MSE) and the coefficient of determination, also called the R-squared measure. The MSE provides a general idea of the magnitude of the error in the predictions, while the R-squared metric indicates how much of the variation in the dependent variable, the observed scores, is explained by the automatic model. The observed values for spectra were obtained by manually inspecting mass spectra. This is done according to the procedure explained in 2.7, subsection Manual inspection of mass spectra. In addition, the results of both manual and automatic examination were compared to find possible errors in manual inspection, which were corrected only if this error could be confirmed by another manual examination of the spectrum.

3. Results

The newly written compound evolver software, comprised of the genetic algorithm implementation, a web application for accessing the GA, and a command line grid search tool, was created to find high affinity compounds within large chemical search spaces. In the following sections, the main features and functionality of the genetic algorithm, the web application are presented, as well as the results of the parameter tuning procedure. In addition, the usage, output and performance of the automatic analysis of mass spectra is also presented.

3.1. Genetic algorithm

The genetic algorithm makes use of a user defined search space in the form of one or multiple reaction schemes and multiple sets of reactants, or building blocks. Figure 5 shows the pattern that is carried out in the genetic algorithm.

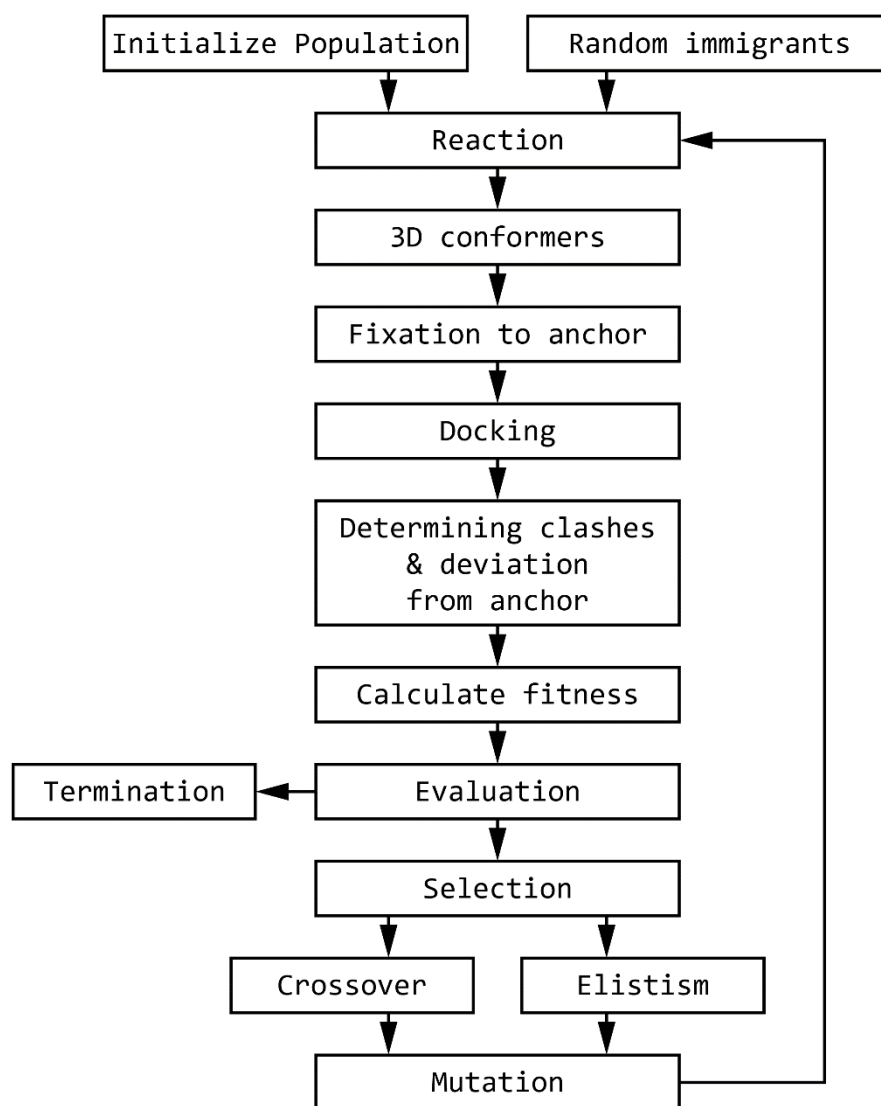
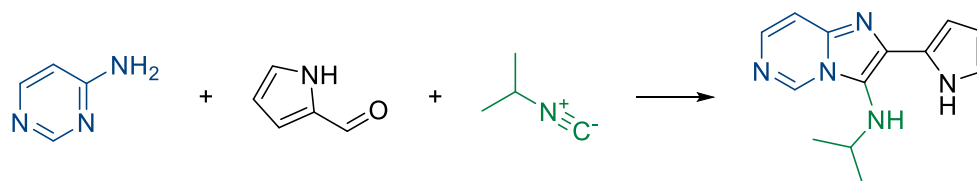


Figure 5
A simplified flowchart of the implemented genetic algorithm pattern.

Intermediate pipeline results

The genetic algorithm makes use of various pipeline steps to obtain a binding affinity score from a non-3D molecule. In this section the various results of the pipeline steps are described and visualized. The visualized intermediate results were generated by the genetic algorithm with the GBB multi-component reaction shown in Table 1, with the menin protein as the target, with corresponding anchor (Figure 1).

The first step in the pipeline is creating a single molecule from the fragment molecules that are represented in the genotype of a candidate. Scheme 1 shows a reaction that was performed in a genetic algorithm run.



Scheme 1

Reaction scheme of a Groebke-Blackburn-Bienaymé multi-component reaction creating N-isopropyl-2-(1H-pyrrol-2-yl)imidazo[1,2-c]pyrimidin-3-amine.

The second step in the pipeline is generating multiple 3D conformers from the molecule. This is done by ChemAxon's Conformer plugin. shows fifteen conformers that are generated in this step, which is the default amount.

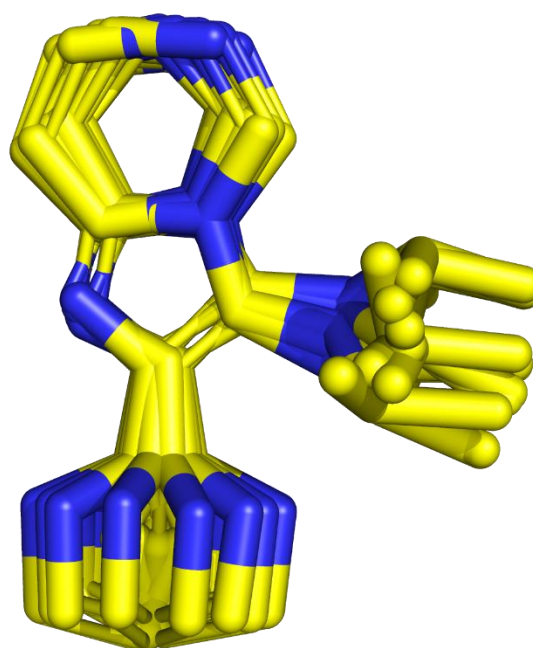


Figure 6

Fifteen conformers generated in the GA pipeline by ChemAxon's Conformer Plugin. Rendered with PyMOL. [39]

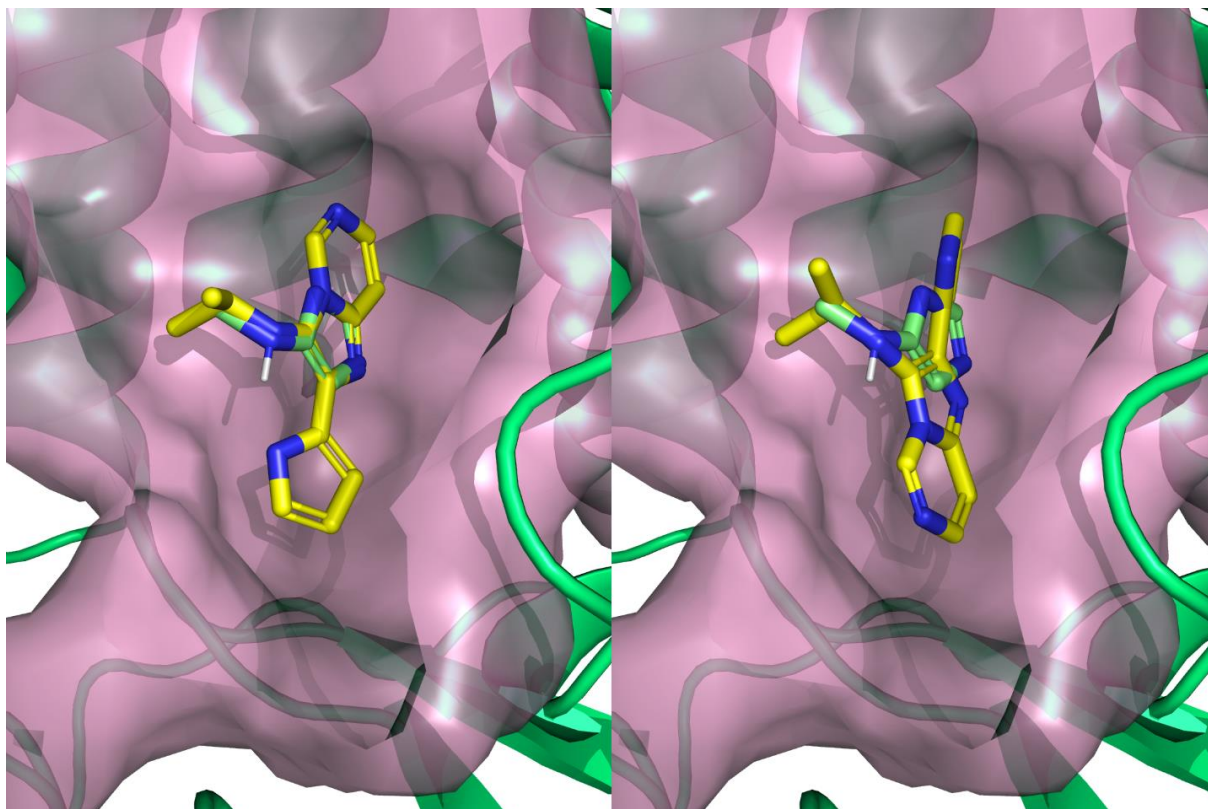


Figure 7

Two conformers that are aligned in the pocket of the menin protein. Both rendered with PyMOL. [39] The protein's surface is purple within the pocket, and the green colour shows the structure of the protein. The yellow molecules are the aligned molecules, and the anchor is shown in light green both the left and right renders.

The next step in the pipeline is placing every conformer within the receptor pocket that has been specified by the user via an anchor molecule. Figure 7 shows two of the generated conformers aligned to the anchor molecule.

Finally, each conformer is minimized by using either Moloc's Mol3d tool, or Smina. These subsequently report a binding affinity score for each conformer individually. Figure 8 shows the best minimized conformer out of fifteen.

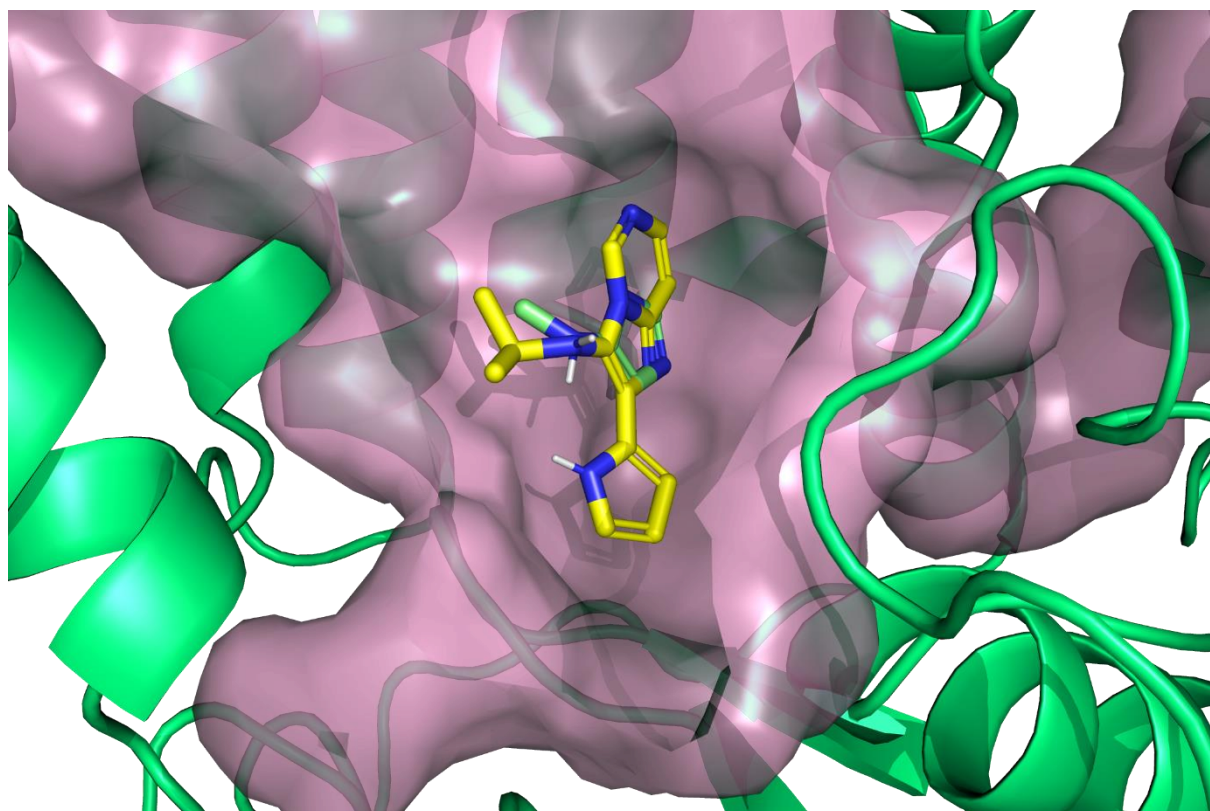


Figure 8

The best conformer out of fifteen conformers when minimized. The protein's surface is purple within the pocket, and the green colour shows the structure of the protein. The yellow molecule is the minimized conformer, and the anchor is shown in light green. Rendered with PyMOL. [39]

3.2. Web application

A web application has been built that gives the end user the ability to use and interact with the genetic algorithm. The graphical user interface of the software package offers a simplistic way to provide parameters to the genetic algorithm. Through the GUI the user is able to specify reactants, reactions, how the reactants map to the reactions, a protein target and an anchor molecule. All genetic operator parameters can be supplied as well as methods for the termination of the GA. Besides this, the fitness measure can be supplied, the amount of conformers to generate, the docking method to use and finally, filter settings which include Lipinski rules, maximum deviation from the anchor and the tolerance for the exclusive shape (the latter both in Ångström). After submitting the form, evolution will be started and the results will be plotted. The user is able to inspect the results by clicking on a generation and subsequently click on a specific compound to download the data. After the download is complete the best conformer can be opened by a molecule visualization program. In Figure 9 are two screenshots presented from the GUI that show a part of the form that is used to provide parameters and a plot that shows the genetic algorithm progress.

Server communication

The server-side structure uses several Servlets to handle Hypertext Transfer Protocol (HTTP) requests. A servlet initiates evolution and establishes a progress connector via a session instance. The progress of the evolution is pushed to the progress connector which can be read by another servlet after a request from the client-side. During, and after the evolution, the files produced by the pipeline process are stored to make them downloadable by the user.

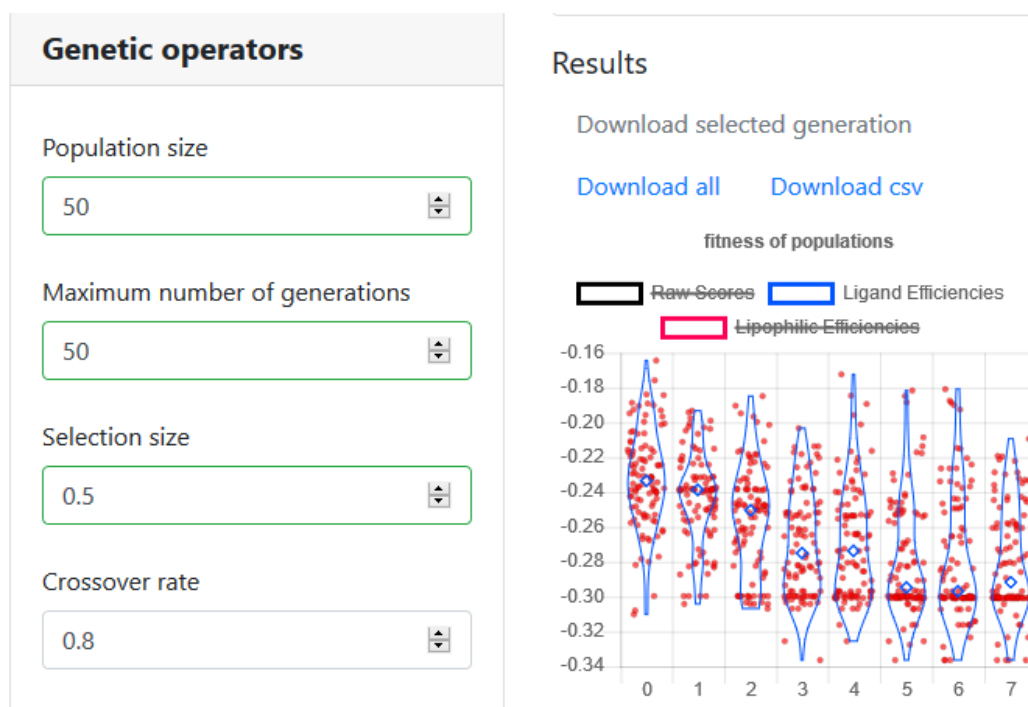


Figure 9

On the left side a part of the GUI is shown which takes parameter values from the user. The right side shows the progress of a genetic algorithm run in a plot. A user is able to see a plot of the raw binding affinity scores, the ligand efficiencies or the lipophilic efficiencies.

3.3. Genetic algorithm parameter tuning

A genetic algorithm is dependent on a series of parameters that can have a large effect on the results. To gain insight on what the parameter space looks like a parameter tuning procedure was carried out. In this section the results of the parameter tuning methods are presented.

Results of initial experiments

Initial trial and error experiments show that a selection size of around 0.5 of the population size performs best. The generation of new offspring seemed best with a crossover rate around 0.8, an elitist rate of around 0.1 and a random immigrants rate of 0.1 as well. The mutation rate seemed best between 0.1 and 0.3.

Grid search results

The main parameter tuning method that was chosen is a grid search algorithm. 90 combinations of parameters were sampled to find an optimal combination. The effects of these first 90 combinations are presented in appendix A, Grid search parameter effects.

After the grid search runs were done, a linear model was built to explain what effect independent variables have on the best-of-run fitness score. Table 6 presents the output summary of the model. The table shows that the estimate for coefficients of mutation method and elitist rate shows a relatively large effect and is significant. The coefficient estimate for population size is according to this also significant, but it shows only a low amount of effect.

The output of the grid search and the model was analysed to confirm that the model can be trusted. Figure 10 shows the amount of candidates that are evaluated per run. What can be seen is that the number of candidates evaluated per run decreases from 18000, to less than 5000 candidates, which has an effect on the best-of-run scores.

Table 6

A summary of the output of the linear model that was made from the grid search experiment. This summary shows the estimate for the coefficients of the variables are located in the second column. The fifth column shows the p-value which shows, together with the last column, at what confidence level, the estimate is significant. It shows that the coefficient estimates for the mutation method, the population size and the elitist rate are significant (< 0.05), although the population size shows a small effect size.

	Estimate	Standard Error	t-value	p-value	Significance
Intercept	3.36E+02	9.15E+00	36.676	$< 2e-16$	***
Mutation method	-7.52E+00	8.95E-01	-8.404	3.94e-16	***
Mutation rate	5.88E+00	5.48E+00	1.072	0.28428	
Population size	-2.19E-02	1.10E-02	-1.995	0.04659	*
Crossover rate	1.73E+01	9.14E+00	1.889	0.05945	.
Elitist rate	2.69E+01	9.67E+00	2.776	0.00569	**

Significance codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

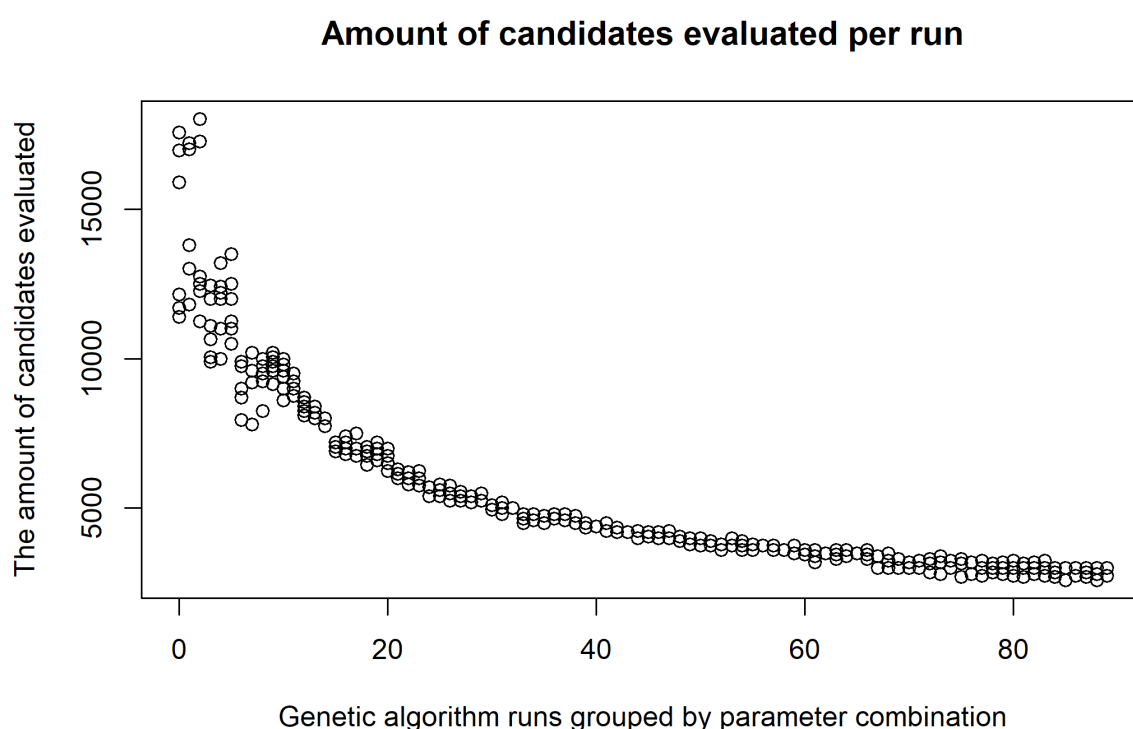


Figure 10

The amount of candidates evaluated per run. Note that the amount decreases as more runs are performed. Initially the amount of candidates that are evaluated is between 10000 and 18000, while more than half of the runs were only able to evaluate less than 5000 candidates, with a minimum of 2600.

A new set of optimization runs was performed in which, instead of a fixed duration per run, a fixed amount of target candidate evaluations was set for each run. Five repetitions were run for every combination of parameters that is tested to stay within the limits for computation time. For the same reason, the number of target candidate evaluations was set to 4000, which accommodates 26 generations for a population size of 150, enough to reach convergence. Subsequently, a new linear model was built. Table 7 presents a summary of this new model. The model shows no significant estimate apart from the intercept.

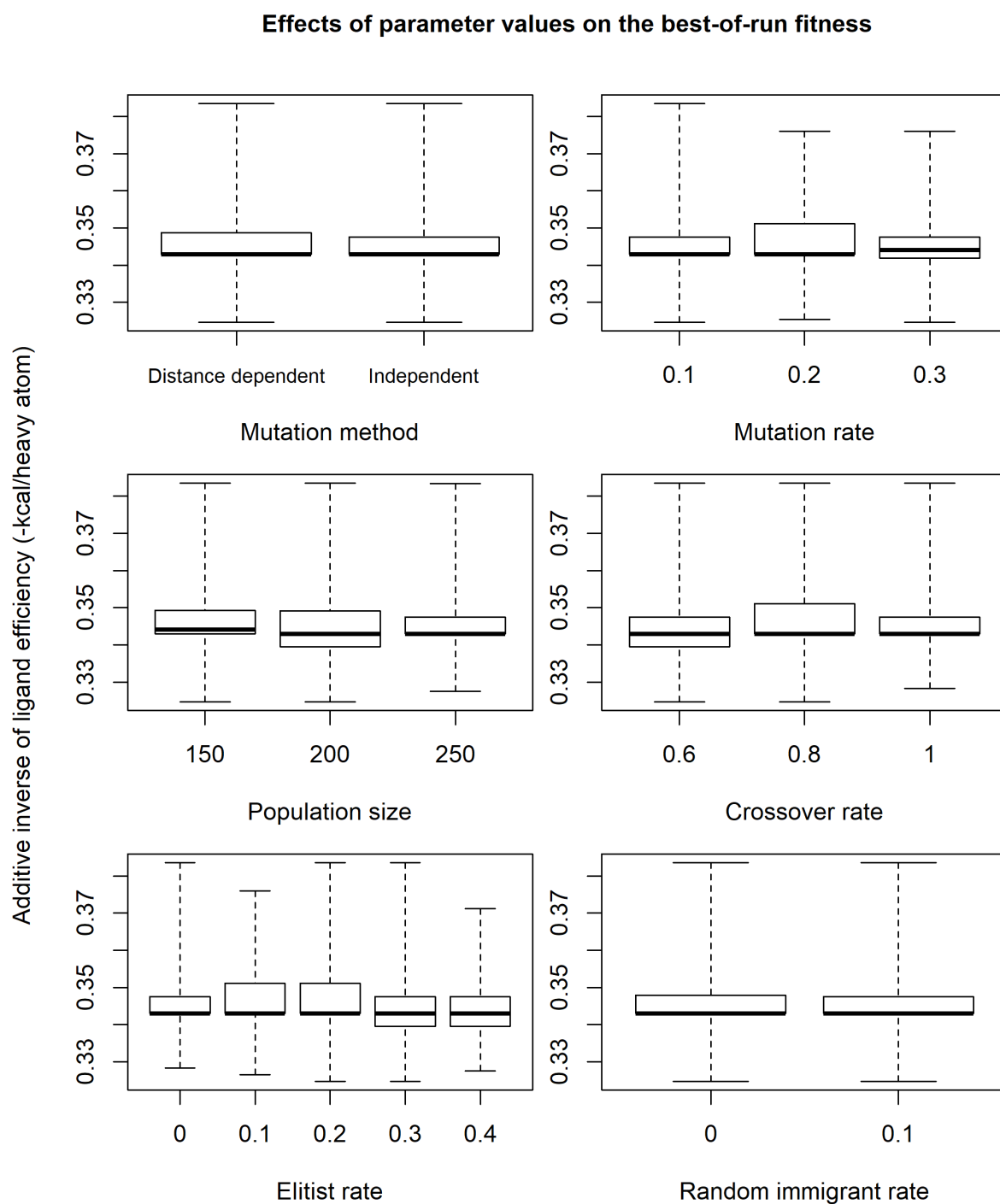


Figure 11
Boxplots that show how the additive inverse of the fitness, ligand efficiency in this case, is affected by different parameter values. All parameters show very little differences between values.

Table 7

A summary of the output of the linear model that was made from the grid search experiment. This summary shows the estimate for the coefficients of the variables are located in the second column. The fifth column shows the p-value which suggests, together with the last column, at what confidence level, the estimate is significant. The table shows low effect sizes and the only estimate that is regarded as significant is the intercept.

	Estimate	Standard Error	t-value	p-value	Significance
Intercept	3,44E+02	3,92E+00	87.913	<2e-16	***
Mutation method	1,29E-01	9,74E-01	0.132	0.895	
Mutation rate	-1,24E+00	5,97E+00	-0.207	0.836	
Population size	-1,08E-02	1,19E-02	-0.908	0.364	
Crossover rate	5,28E+00	3,44E+00	1.533	0.126	
Elitist rate	1,01E+01	1,05E+01	0.956	0.340	

Significance codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

In an attempt to acquire other results from the grid search, combinations of parameters were searched that shows better performance than others. Figure 12 shows the difference in performance between these combinations of parameters. Group 9 especially seems to have a better performance than the other parameter combinations. The parameters of group 9 are the following: a population size of 150, a mutation rate equal to 0.1, a crossover rate of 0.6, an elitism rate of 0.3, a random immigrant rate of 0.1 and the distance independent mutation method.

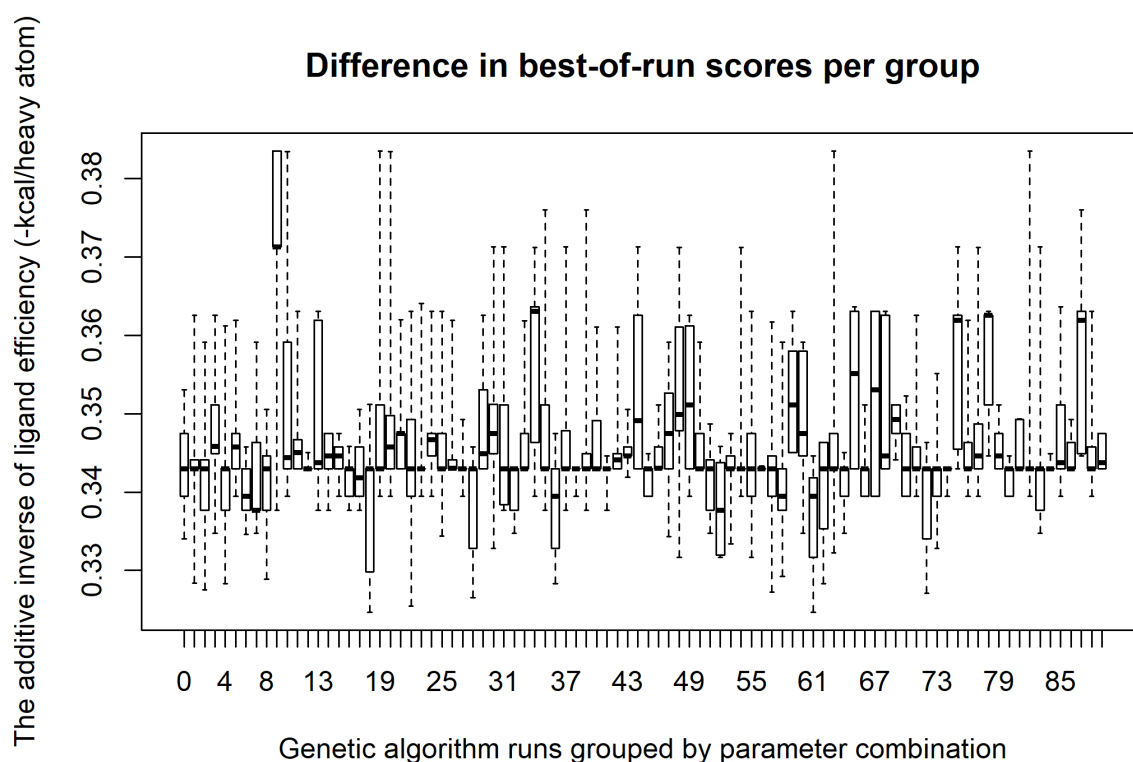


Figure 12

The difference between best-of-run scores grouped by parameter combination. The figure shows that group 9 seems to perform better than the other groups.

To test whether or not the differences between these groups also show a significant difference a series of statistical tests are performed. As the aim of the test corresponds with the use case of

analysis of variance (ANOVA), the data is examined to check if it satisfies the assumptions for an ANOVA about normality of the response variable and homogeneity of variances (equality of population variance within each group). By using Bartlett's test, the data proved to not satisfy the assumption about homogeneity of variances (p-value equal to $2.008e-13$). Manual inspection of a quantile-quantile (Q-Q) plot showed that also normality could not be assumed. Therefore, instead of an ANOVA, a Kruskal-Wallis Rank Sum test was performed that tests whether the samples originate from the same distribution. The Kruskal-Wallis Rank sum test returned a p-value of 0.1531 and thus the hypothesis that a group differs from others cannot be accepted.

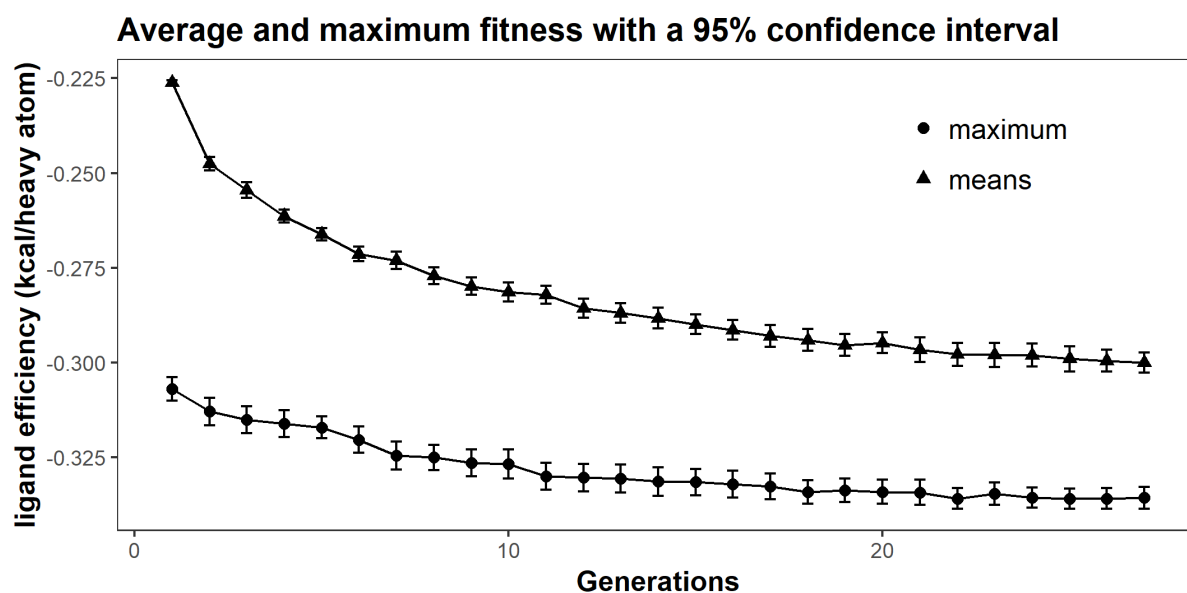
3.4. Filtering effects

To restrain possible solutions from having too far deviated from the anchor position a filtering step was implemented that removes conformers with an RMSD greater than a user-set limit. The effects of this filtering step are presented here. The effects of removing the conformers that clash with the protein are also shown. Two experiments were carried out with one having menin as a protein target, with accompanying anchor, (Figure 1) and programmed death-ligand 1 as a protein target, with accompanying anchor (Figure 2). The reaction used for the menin protein is the one presented in Table 1. The reaction used for the PD-L1 target comes from unpublished work.

In fifty GA runs with a total of 4050 candidate evaluations, an average of 1822 conformers appeared to clash with a tolerance value set to 0 Ångström. The standard deviation in the GA runs was 692, rounded to the nearest integer value. The average amount of candidates that were removed due to a too large deviance from the anchor (>2 Ångström) was around 1192, with a standard deviation of 106 candidates. The GA runs were performed with the parameters from parameter combination 9, which is described in section 3.3. In the second experiment, with PD-L1 as the target, sixteen GA runs are performed with the same set of parameters. In the 4050 candidates, each with 15 conformers, a total of 59005 conformers on average clashed with the protein (97% of the total), with a standard deviation of 393. The average number of candidates that were removed due to a too large deviance from the anchor was around 646, with a standard deviation of 43 candidates.

3.5. Average performance over time

To assess whether the genetic algorithm suffers from getting stuck in local optima, the GA is run multiple times, which results in the algorithm starting at different locations in the search space. the average and maximum fitness over multiple repetitions are resented in Figure 13.



Figure

13

The average and maximum fitness of generations with a 95% confidence interval. The graph shows that compounds improve until convergence is reached. The parameters are the following: a population size of 150, a mutation rate equal to 0.1, a crossover rate of 0.6, an elitism rate of 0.3, a random immigrant rate of 0.1, the distance independent mutation method, tournament selection and a selection size of 0.5. The menin protein was used in combination with a GBB reaction.

3.6. Automatic mass spectra analysis

The second aspect that is discussed in this report is the automatic analysis of mass spectra. A new program was written that reports a score of mass spectra generated with SFC. [44]

Application interfaces

The application that was written offers two manners to interact with the automatic analysis of mass spectra. The first of these is the command line interface (CLI), which delivers a to the point option list for specifying the necessary input and output files as well as the optional mass-to-charge ratio, and retention time ranges. Besides this CLI, a GUI was constructed that incorporates all the functionality that is available via the command line, and adds the option for users to modify the list of adduct masses that are checked in analysis. A screenshot of the GUI is presented in **Error! Reference source not found.**

Mass spectrum analyzer

Select mzXML files

browse

Select smiles file

browse

Enter output file

browse

M/Z range

200 - 600

Time range

0.2 - 0.5

frequent adduct masses

Name	Multiplication factor	Addition of mass
M+H; 1; 1.007276		
M+NH4; 1; 18.033823		
M+Na; 1; 22.989218		
M+CH3OH+H; 1; 33.033489		
M+K; 1; 38.963158		
M+ACN+H; 1; 42.033823		

Add Delete

Run

Figure 14

A screenshot of the automatic mass spectra analysis software. The input fields, from top to bottom, are meant for: Input mzXML files, the smiles file with plate location information, the output file path, the range of mass-to-charge ratios that should be considered, the retention times between which should be used in the analysis, and finally the fields for adding an adduct mass that the software should look for in the spectrum.

In 2.7, Automatic mass spectra analysis, is described that the mzXML file format was chosen for importing the data. For other formats, the user thus has to perform a conversion. The following manual steps are required from the user to be able to use this program. The program and some dependencies first need to be installed. The next step is converting the raw files to mzXML files and finally the program can be started. The user has to fill in some parameters and can click run to get the output.

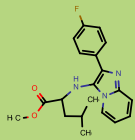
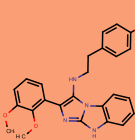
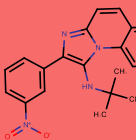
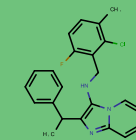
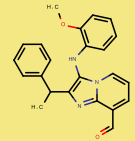
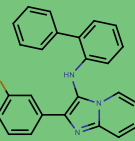
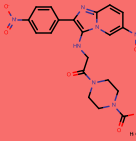
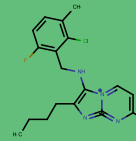
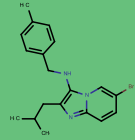
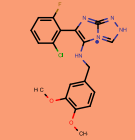
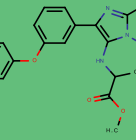
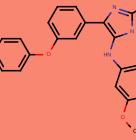
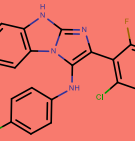
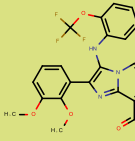
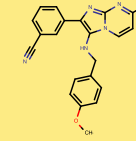
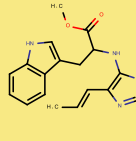
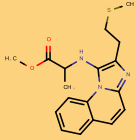

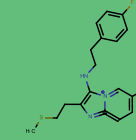
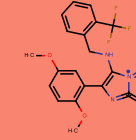
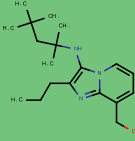
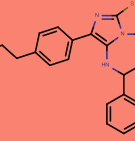
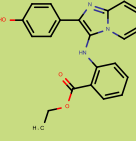
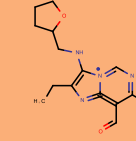
Output

While the application that was made reduces the time that has to be spent by analysing the spectra automatically, reporting the results of the analysis step is also accelerated with this program. After analysis is completed, an excel file is made that contains various different worksheets that present the analysis results. The first of these is a simple heatmap that shows the different samples according to their location on the plate. The second table shows a similar heatmap that contains SMILES in the cells which can be converted to structures with a Microsoft Office Add-in. Because those cells that contain SMILES cannot be formatted automatically according to the score of the compound, a Visual Basic for Applications (VBA) for Office procedure has been made to copy the colours applied to a separate table with the score values by conditional formatting. After this procedure is applied to the excel sheet parts of the table with structures can be copied to Microsoft Word for example. Table 8 shows an example of the results of this procedure. Because the structures in Microsoft Word appear as pictures, editing the table in Word proved to be

challenging. To solve this problem another VBA procedure was developed that changes the text wrapping of the pictures, to the 'tight' method. Besides these heatmaps, a tabular representation of the results are also present in an output file.

Table 8

A part from a heatmap that was made after the automatic analysis of mass spectra. The table was automatically generated with the colours separated in another table in Microsoft Excel. Subsequently a VBA for Office procedure was used to copy the colours to this table. SMILES were converted to structures using the Jchem for Office Add-in.

	A	B	C	D
1				
2				
3				
4				
5				
6				

3.7. Performance evaluation

Evaluation of a model that was developed is an integral part of the process. To evaluate the performance of the automatic mass spectrometry analysis. The mean squared error was calculated as well as the coefficient of determination, or R-squared value. This was done for the dataset that was used for setting up the prediction and processing methods and for the testing dataset.

Table 9

An overview of the automatic mass spectrometry analysis performance. The MSE and R-squared values are calculated for the training dataset, the testing dataset and the two datasets together.

Metric	Data used	Value
Mean squared error	Training data	0.024287
	Testing data	0.03317118
	Overall	0.03095303
R-squared	Training data	0.8215217
	Testing data	0.8148848
	Overall	0.8172361

Table 9 shows that the mean squared error and the R-squared value show only little difference between the training and testing datasets. The time needed to analyse an entire well with compounds is roughly 20 minutes on a computer with an `Intel® Core™ i3-4130 CPU @ 3.40GHz` processor, 8 Gigabytes of memory, and the Microsoft Windows 10 64-bit OS, version 1803. The time that is spend with manual assessment covers multiple days.

4. Conclusion and discussion

For this project, the hypothesis was that genetic algorithms can be used as a form of artificial intelligence to find high affinity compounds for a protein target. To test this hypothesis, a software package was made that links functionality from existing software for docking in a pipeline that acts as a fitness function, a genetic algorithm was implemented that makes use of this fitness function. In addition, a web interface was constructed, the genetic algorithm results are analysed, and parameter tuning was performed. Another aspect of artificial intelligence driven drug design is accelerating the analysis of mass spectrometry data, which also was discussed in this report.

In section 3.13.2 is shown how the genetic algorithm and pipeline are implemented. It shows that the intermediate pipeline steps from reaction, conformer generation, alignment to the anchor and minimization all work as intended.

The functionality of the web application that wraps the GA is listed in section 3.2. The web application is able to perform an adequate selection of actions. However, there is also room for improvement. The current implementation of the web application sets no limit on how many people are able to run the computationally demanding program at the same time, which may not be a major problem for internal use, but might become one if the web application would be accessible for external use.

Section 3.3 presents the parameter tuning results and analysis. The parameter tuning results show that there is a no clear advantage in choosing a specific set of values out for the different parameters that are tested. In the grid search, only five runs were carried out for each combination of parameters, which is considerably lower than the ten to thirty repetitions that are advised. [37] This could contribute to the fact that different parameter combinations did not seem to significantly affect the best-of-run fitness scores.

The main method used in parameter tuning is a grid search, which is normally is very capable of exploring the parameter space. The applied grid search however suffers from the large amount of experiments needed to assess a relatively small amount of parameters, which forces the use of - possibly - a too small search space. To perform a grid search, a finite set of “reasonable” values for each parameter has to be chosen, which could introduce a bias and thus deceiving results. Even with 450 GA runs - consisting of 90 parameter combinations, each ran 5 times - the minimum recommended number of repetitions are not met.

To limit the amount of parameter combinations that had to be run, the selection method and selection size parameter values were determined prior to the grid search experiment. Initial experiments, together with conventions and standards resulted in the tournament selection being chosen out of three implemented selection methods. However, no statistical methods were used to show a significant difference in performance. In genetic algorithm literature, and metaheuristics in general, many other methods for optimization and genetic operators are described. And since no optimal set of parameters could be obtained, it is likely that improvements can be made to the current setup.

Section 3.4 describes the effect of filtering conformers that clash with the target protein even after energy minimization. It shows that the amount of conformers that are filtered is greater in case the target protein has a more restricting structure, as is expected. The number of compounds that

were too far away from the anchor after minimization actually went down. This could be attributed to the restricting structure as well, because the compound was not able to move greatly inside the cavity of the protein. Considering these findings, it is advised to increase the amount of conformers that should be generated, if evolving compounds for a more restricting target protein is attempted. It should also be noted that the conformer generation plugin from Chemaxon currently is not able to preserve the stereochemistry of asymmetric carbon atoms, which might introduce compounds with a stereochemistry that is less favourable in a solution. In the future this could be solved by filtering conformers based on stereochemistry.

In section 3.5 can be seen that the genetic algorithm is able to converge to an optimal compound, and that multiple repetitions of the same experiment end up at the same optimum, suggesting that the GA does not get stuck in a local optimum. However, as can be seen in Figure 12, higher ligand efficiencies are sometimes encountered than those that are around the convergence point in Figure 13. And genetic algorithms are known to be prone to getting stuck in local optima.

A second aspect that was discussed in this report is the acceleration of mass spectrometry analysis through a new piece of software. The implementation and interface of the automatic mass spectrometry analysis software is described in section 3.6. The analysis duration is compared and the program offers a great increase in speed. In section 3.7 two performance metrics are presented, which show more than, adequate numbers. While this newly written piece of software works as intended, currently a baseline of peaks in a spectrum is not considered in scoring, while a very low noise level for example should return an increased score relative to a spectrum with a level of noise.

In conclusion, using artificial intelligence in drug design is a promising technique to both drive drug affinity optimization, and perform automatic analysis of mass spectrometry results. These findings in combination with recent developments in being able to rapidly synthesize and screen a large amount of compounds, could provide significant advantages for drug discovery in terms of efficiency, diversity and costs.

5. References

- [1] C. Kalinski, M. Umkehrer, L. Weber, J. Kolb, C. Burdack en G. Ross, „On the industrial applications of MCRs: molecular diversity in drug discovery and generic drug synthesis,” *Molecular Diversity*, vol. 14, pp. 513-522, 2010.
- [2] L. Weber and A. Dömling, “Evolutionary multicomponent reaction (MCR) chemistry provides a boon/boost to efficiency and diversity in drug discovery,” b5srl, 2014.
- [3] R. P. Sheridan en S. K. Kearsley, „Using a Genetic Algorithm To Suggest Combinatorial Libraries,” *J. Chem. Inf. Comput. Sci.*, vol. 35, nr. 2, pp. 310-320, 1995.
- [4] F. Dey en A. Caflisch, „Fragment-Based de Novo Ligand Design by Multiobjective Evolutionary Optimization,” *J. Chem. Inf. Model.*, vol. 28, pp. 679-690, 2008.
- [5] “Introduction to Genetic Algorithms,” 5 1996. [Online]. Available: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html. [Accessed 4 9 2018].
- [6] J.-P. Rennard, “Introduction to Genetic Algorithm,” 5 2000. [Online]. Available: <http://www.rennard.org/alife/english/gavintrgb.html>. [Accessed 4 9 2018].
- [7] A. Dömling, „Recent Developments in Isocyanide Based Multicomponent Reactions in Applied Chemistry,” *Chem. Rev.*, vol. 106, nr. 1, pp. 17-89, 2006.
- [8] C. A. Lipinski, F. Lombardo, B. W. Dominy en P. J. Feeney, „Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings,” *Advanced Drug Delivery Reviews*, vol. 46, nr. 1-3, pp. 3-26, 2001.
- [9] I. D. Kuntz, K. Chen, K. A. Sharp en P. A. Kollman, „The maximal affinity of ligands,” *Proc Natl Acad Sci U S A*, vol. 96, nr. 18, p. 9997-10002, 1999.
- [10] M. P. Edwards en D. A. Price, „Role of Physicochemical Properties and Ligand Lipophilicity Efficiency in Addressing Drug Safety Risks,” *Annual Reports in Medicinal Chemistry*, vol. 45, pp. 380-391, 2010.
- [11] J. Bergstra en Y. Bengio, „Random Search for Hyper-Parameter Optimization,” *J. Machine Learning Research*, vol. 13, p. 281-305, 2012.
- [12] M. Mitchell, *An introduction to genetic algorithms*, MIT Press, 1998.
- [13] J. H. Holland, „Genetic Algorithms,” *Scientific American*, vol. 267, pp. 66-72, 1992.
- [14] O. Roeva, S. Fidanova en M. Paprzycki, „Influence of the population size on the genetic algorithm performance in case of cultivation process modelling,” in *Federated Conference on Computer Science and Information Systems*, Kraków, Poland, 2013.
- [15] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, Massachusetts: Addison-Wesley, 1989.

- [16] B. L. Miller en D. E. Goldberg, „Genetic Algorithms, Tournament Selection, and the Effects of Noise,” *Complex Systems*, vol. 9, pp. 193-212, 1995.
- [17] T. Kuthan en J. Lánský, „Genetic Algorithms in Syllable-Based Text Compression,” 2007, p. 21-34.
- [18] Jansen en Wegener, „The Analysis of Evolutionary Algorithms—A Proof That Crossover Really Can Help,” *Algorithmica*, vol. 34, nr. 1, pp. 47-66, 2002.
- [19] P. G. A. Pedrioli, J. K. Eng, R. Hubley, M. Vogelzang, E. W. Deutsch, B. Raught, B. Pratt, E. Nilsson, R. H. Angeletti, R. Apweiler, K. Cheung, C. E. Costello, H. Hermjakob, S. Huang, R. K. J. Jr en E. K, „A common open representation of mass spectrometry data and its application to proteomics research,” *Nature Biotechnology*, vol. 22, p. 1459-1466, 2004.
- [20] D. Kessner, M. Chambers, R. Burke, D. Agus en P. Mallick, „ProteoWizard: open source software for rapid proteomics tools development,” *Bioinformatics*, vol. 24, nr. 21, p. 2534-2536, 2008.
- [21] R. Warmerdam, “ca_warmerdam / compound-evolver,” 2019. [Online]. Available: https://bitbucket.org/ca_warmerdam/compound-evolver/src/master/. [Accessed 14 1 2019].
- [22] D. Weininger, „SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules,” *J. Chem. Inf. Comput. Sci.*, vol. 28, nr. 1, pp. 31-36, 1988.
- [23] ChemAxon, „ChemAxon - Software Solutions and Services for Chemistry & Biology,” [Online]. Available: <https://chemaxon.com/>.
- [24] Gerber Molecular Design, “Moloc, a Molecular Design Software Suite,” [Online]. Available: <http://www.moloc.ch/>. [Accessed 2018 9 24].
- [25] G. Imre, G. Veress, A. Volford en Ö. Farkas, „Molecules from the Minkowski space: An approach to building 3D molecular structures,” *Journal of Molecular Structure: THEOCHEM*, Vols. %1 van %2666-667, pp. 51-59, 2003.
- [26] N. M. O'Boyle, M. Banck, C. A. James, C. Morley, T. Vandermeersch en G. R. Hutchison, „Open Babel: An open chemical toolbox,” *Journal of Cheminformatics*, vol. 3, nr. 33, 2011.
- [27] A. J. O. Oleg Trott, „AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading,” *Journal of Computational Chemistry*, vol. 31, pp. 455-461, 2009.
- [28] D. R. Koes, M. P. Baumgartner en C. J. Camacho, „Lessons Learned in Empirical Scoring with smina from the CSAR 2011 Benchmarking Exercise,” *Journal of Chemical Information and Modeling*, vol. 53, nr. 8, p. 1893-1904, 2013.
- [29] J. Sunseri en D. R. Koes, „Pharmit: interactive exploration of chemical space,” *Nucleic Acids Res.*, vol. 44, nr. Web Server issue, pp. W442-W448, 2016.
- [30] T. Tanimoto, “An elementary mathematical theory of classification and prediction,” International Business Machines Corporation, 1957.

- [31] Chemaxon Ltd., "Similarity search," Chemaxon Ltd., 3 8 2018. [Online]. Available: <https://docs.chemaxon.com/display/docs/Similarity+search>. [Accessed 18 12 2018].
- [32] S. S. A.E. Eiben, „Parameter tuning for configuring and analyzing evolutionary algorithms,” *Swarm and Evolutionary Computation*, vol. 1, nr. 1, pp. 19-31, 2011.
- [33] J. Bergstra, R. Bardenet, Y. Bengio en B. Kégl, „Algorithms for Hyper-Parameter Optimization,” *Neural Information Processing Systems Foundation*, vol. 24, 2011.
- [34] R. Storn, „On the usage of differential evolution for function optimization,” in *Proceedings of North American Fuzzy Information Processing*, Berkeley, CA, USA, 1996.
- [35] I. Harvey, „Artificial Evolution: A Continuing SAGA,” *Evolutionary Robotics*, pp. 94-109, 2001.
- [36] T. Blickle en L. Thiele, „A Comparison of Selection Schemes used in Genetic Algorithms,” Computer Engineering and Communication Networks Lab, Zurich, 1995.
- [37] M. Wineberg and S. Christensen, “An Introduction to Statistics for EC Experimental Analysis,” 2004. [Online]. Available: <http://www.cis.uoguelph.ca/~wineberg/publications/ECStat2004.pdf>. [Accessed 17 1 2019].
- [38] RefSeq, “MEN1 menin 1 [Homo sapiens (human)],” NCBI, October 2008. [Online]. Available: <https://www.ncbi.nlm.nih.gov/gene?Db=gene&Cmd=ShowDetailView&TermToSearch=4221>. [Accessed 9 1 2019].
- [39] „The PyMOL Molecular Graphics System,” Version 2.2.0 Schrödinger, LLC..
- [40] RefSeq, “CD274 molecule [Homo sapiens (human)],” NCBI, September 2015. [Online]. Available: <https://www.ncbi.nlm.nih.gov/gene?Db=gene&Cmd=ShowDetailView&TermToSearch=29126>. [Accessed 11 1 2019].
- [41] Python Software Foundation, „Python Language Reference, version 3.6,” [Online]. Available: <http://www.python.org>.
- [42] Waters, “MassLynx Mass Spectrometry Software,” Waters, [Online]. Available: http://www.waters.com/waters/en_NL/MassLynx-Mass-Spectrometry-Software-/nav.htm?cid=513164&locale=en_NL. [Accessed 9 January 2019].
- [43] Z. Jaman, A. Mufti, S. Sah, L. Avramova en P. D. H. Thompson, „High Throughput Experimentation and Continuous Flow Validation of Suzuki–Miyaura Cross-Coupling Reactions,” *Chemistry – A European Journal*, vol. 24, nr. 38, pp. 9546-9554, 2018.
- [44] R. Warmerdam, “ca_warmerdam / auto-ms-analysis,” 2019. [Online]. Available: https://bitbucket.org/ca_warmerdam/auto-ms-analysis/src/rules/. [Accessed 18 1 2019].

Appendices

A. Grid search parameter effects

Effects of parameter values on the best-of-run fitness

