Gradient Based Optimization in Ligand-Receptor Docking

Dissertation
zur Erlangung des Grades eines
Doktors der Ingenieurwissenschaften
(Dr. Ing.)
der Naturwissenschaftlich-Technischen Fakultät I
– Mathematik und Informatik –
der Universität des Saarlandes

vorgelegt von
Dipl.-Bioinform. Jan Fuhrmann

Saarbrücken
2009

Datum des Kolloquiums:	28.10.2009
Dekan:	Professor Dr. Joachim Weickert, Saarbrücken
Mitglieder des Prüfungsausschusses:	
	Professor Dr. Thorsten Herfet, Saarbrücken (Vorsitzender)
	Dr. Dirk Neumann, Saarbrücken
	Professor Dr. Dr. h.c. Wolfgang J. Paul, Saarbrücken
	Dr. Andreas Hildebrandt, Saarbrücken

Abstract In this work, we compared six global search heuristics and two scoring functions in the field of ligand-receptor docking. A new way for the gradient based minimization of a ligand whose position in space is defined by translation, orientation and a set of torsional flexible angles was implemented and thoroughly tested. The default local search method of a Lamarckian genetic algorithm was replaced by our novel gradient based approach and the new hybrid was compared to non-gradient global search heuristics. Finally, we present our docking program BALLDock, in which we incorporated our findings.

Zusammenfassung In der vorliegenden Arbeit wurden sechs populationsbasierte Optmierungsheuristiken und zwei Scoring-Funktionen im Hinblick auf ihre Leistungsfähigkeit im Bereich Ligand-Rezeptor Docking miteinander verglichen. Parallel dazu wurde eine neuer Ansatz entwickelt, der die lokale, gradientenbasierte Optimierung partiell flexibler Moleküle, deren Position und Konformation durch Translation, Orientierung und eine Anzahl flexibler Bindungswinkel definiert ist, erlaubt. Danach wurde die gradientenfreie Methode zur lokalen Optimierung eines Lamarck genetischen Algorithmus durch das neuartige gradientbasierte Verfahren ersetzt und dessen Einfluss auf die Ergebnisse der globalen Suchheuristik analysiert. Abschließend wird das Dockingprogramm BALLDock vorgestellt, in das die neu gewonnenen Erkenntnisse einflossen.

Contents

List of abbreviations

DE	differential evolution
DOF	degree of freedom
GA	genetic algorithm
HIN	Hyperchem input file
ICM	internal coordinate mechanics
L-BFGS	limited memory Broyden-Fletcher-Goldfarb-Shanno
LGA	Lamarckian genetic algorithm
MD	molecular dynamics
MDLGA	multi-deme Lamarckian genetic algorithm
MDLGAGR	multi-deme Lamarckian genetic algorithm with gradient based local search
PDB	protein data bank
PLP	piecewise linear potential
PSO	particle swarm optimization
RMSD	root mean square deviation
SLERP	spherical linear extrapolation

1. Introduction

Down to this day it happens that unknown tribes emerge from the deep jungle in south America or some islands in the pacific ocean and even those people, never having had any contact with modern civilization and basically still living in the stone age, try to overcome diseases or palliate pain by some kind of medication. Thus, we can conclude, that medicine is one of the most ancient fields of cultural effort. Still, indicated by the average life expectancy (Fig. 1.1), the capabilities of physicians in the civilized world just 200 years ago do not represent a significant improvement over naturopathy, applied by a tribal medicine man.

In the second half of the nineteenth century, however, scientification of medicine and pharmacy together with advances in other natural sciences led to a better understanding of pathological processes and allowed for a much more effective treatment of diseases. Nevertheless, the quest for new drugs has been a process of trial and error, (e.g. arsphenamine, a therapeutic agent against syphilis discovered by Paul Ehrlich) or even of chance (e.g. penicillin, discovered by Alexander Fleming). Emil Fischer explained the activity of an agent by the ”key-lock principle”,[1] i.e. the drug fits like a key to a target structure. This parable was extended by Daniel Koshland to the induced fit theory[2] by postulating, that the conformation of the target protein changes upon binding of a ligand molecule. Recently, an alternative theory called “conformational selection” was introduced to explain alterations in protein conformations. Of course, it would be desirable to be able to blueprint such a key for a target that has been identified as the cause of a specific disease. In fact, this is a central task in computational chemistry, called rational drug design, which implicates the ligand-receptor docking problem (Fig. 1.2):

> Given a small ligand molecule and a large target receptor, reconstruct the native binding pose of the ligand and calculate the binding free energy.

The binding free energy defines some kind of measure for the quality of the ligand-receptor complex and can be determined experimentally. Unfortunately, this approach is time consuming and expensive, so computational chemistry aims at computing the binding free energy in silico.

To date, there are more than sixty different docking programs available and each one, at least to our knowledge, uses the same basic approach: the space of possible ligand positions and con-

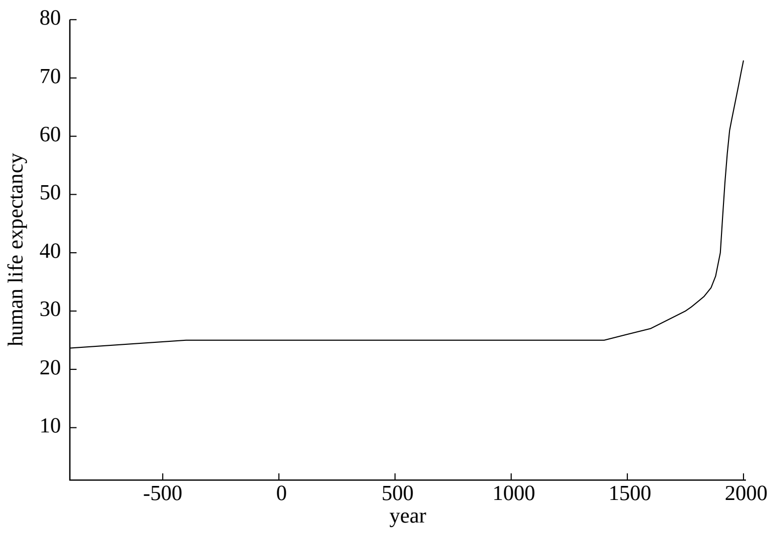


Figure 1.1.: Development of human life expectancy (data extrapolated from multiple sources).

formations is sampled and evaluated by a scoring function. The actual method for the sampling process as well as the scoring function differ from program to program. Results from comparative studies of docking programs are somewhat inconsistent. Nevertheless, there have been two general sampling methods that seem to produce good results on a regular base: fragment-based approaches and population based meta-heuristics. The good performance of the latter is especially surprising as they rely strictly on the one-dimensional result of the scoring function and disregard available information of the potential energy hyper-surface.

A similar task in molecular modeling is the optimization of molecules, based on the energy gradient. Given a structure, ”local optimizers” travel on the potential energy hypersurface (PES) to find the next local minimum, which, hopefully, corresponds to a natural conformation. Here, methods that do not use the energy gradient cannot compete in terms of speed and precision. One example for such a non-gradient method is the Powell minimizer[3] that implicitly gains gradient information by applying bracketing methods to find the minimum of well-defined search directions. This observation raises the question why the energy gradient is regularly ignored in ligand-receptor docking. First, the current programs for molecular optimization work on the pure 3-dimensional representation for each atom, i.e. each atom of the molecule possesses a *x*-, *y*- and *z*-value defining its position in Euclidean space. Contrary to that, most docking programs use an internal representation, which consists of rotations around flexible torsional angles with a 3-dimensional translation and rotation for the whole molecule. This has the advantage of a largely reduced search space. The drawback is the non-trivial computation of the gradient for

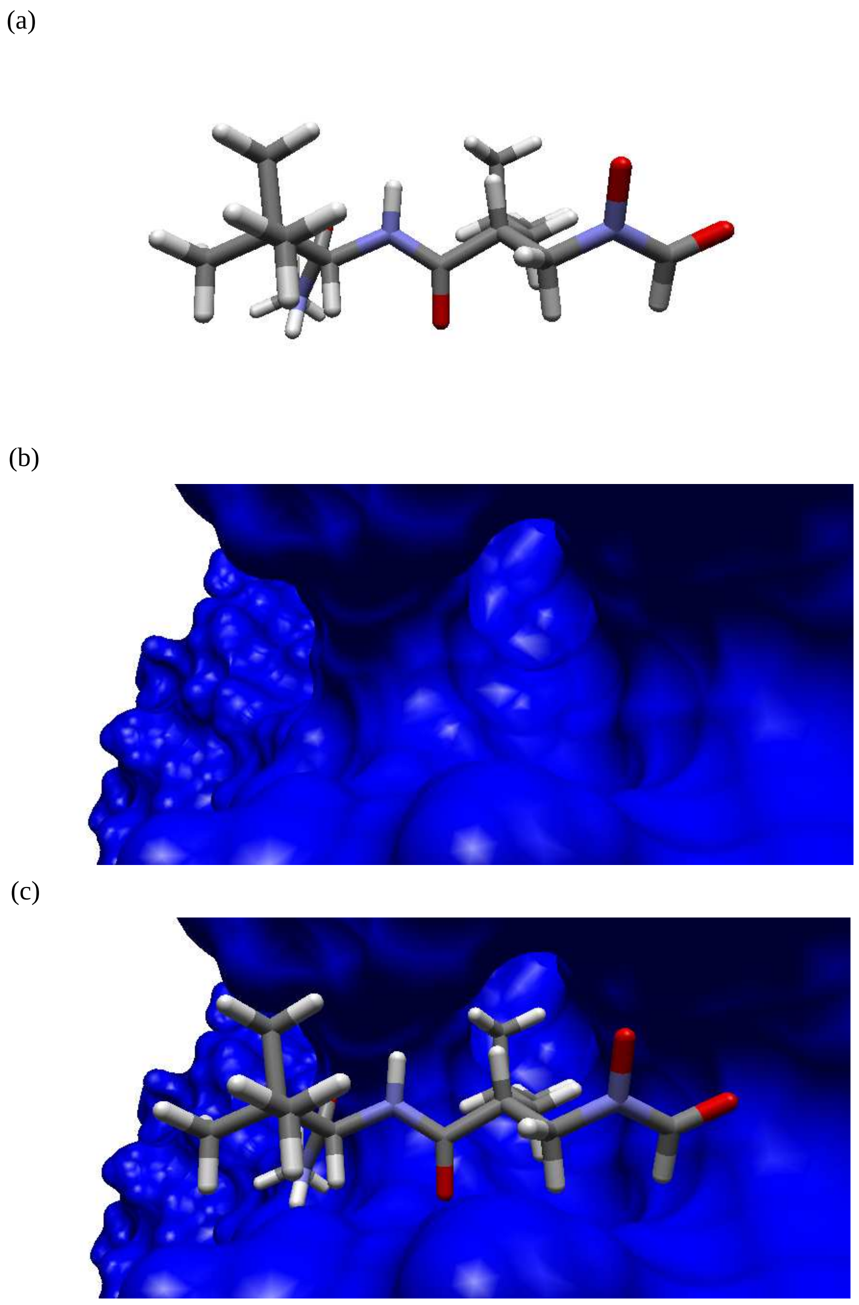


Figure 1.2.: Docking problem illustrated: For a given ligand (a) and receptor (b), we have to find the native binding pose (c).

parameters defining the molecular orientation. There has been an effort to represent not only a single molecule but multiple molecules and their relative positions using virtual atoms and bonds which is called ICM.[4] Additionally, the method to calculate the energy gradient for this representation has been published. However, it must be noted, that the approach to handle the translation and orientation of a rigid body by introducing a set of virtual atoms is highly prone to a gimbal lock like phenomenon:[5] rotational axes may align, leading a loss of one or more degrees of freedom.

2. Objectives

This work aims at answering two main questions: (1) What are the characteristics of population-based meta-heuristics when applied to the ligand-receptor docking problem and (2) Can we improve those meta-heuristics by employing a gradient based local search algorithm.

To study the performance of different population based meta-heuristics, we perform docking experiments with the well established AUTODOCK energy function and piecewise linear potential (PLP) of Gehlhaar. In this process we also analyze the impact of a dedicated local search procedure (Solis & Wets), as proposed by the authors of AUTODOCK. The experiments and their evaluations have to be designed such that they allow for a fair comparison between the various sampling methods and scoring functions.

To answer the second question, we have to develop a method to use gradient based optimization in ligand-receptor docking. This approach requires the computation of the derivatives of the scoring function with respect to the model parameters, which is trivial for translation and flexible torsional angles but problematic for orientational parameters.

In the next step, we replace the local search method of Solis & Wets by our gradient based approach and compare the results to non-gradient search heuristics.

Finally, all optimization methods and the Gehlhaar scoring function are implemented in BALL to provide the docking suite BALLDock.

3. Related Work

3.1. Comparison of docking methods

Most related work was performed by many comparative studies on the accuracy of docking programs and algorithms. In a few cases, only the sampling method was changed while the same scoring function was employed.[6,7] While such studies allow for investigating the influence of the sampling method on the docking results, the number of complexes employed was usually small. In most cases, however, comparisons between different programs using different search heuristics and scoring functions were performed.[8,9] Such studies do not allow to assess the individual influence of the search heuristics or the scoring functions on the docking accuracy, because both are intricately woven with each other in the final program. Hence, such studies impede a fair comparison of the sampling strategies or the scoring functions.[7] In addition, several issues (binding site definition, experience with docking programs etc.) may bias comparisons, too.[10]

3.2. Local optimization in ligand-receptor docking

Local optimization was first applied to ligand-receptor docking in a Lamarckian genetic algorithm by AUTODOCK 3.0, replacing the simulated annealing method of previous versions. Since it does not require any gradient information, AUTODOCK, as well as many docking programs, that are based on AUTODOCK, e.g. PSO@AUTODOCK[11] and SODOCK,[12] use the method of Solis & Wets[13] for local optimization. Interestingly, all three studies unanimously reported a beneficial effect, when local optimization was employed.

In related work with respect to the main focus of this work, gradient based minimization in ligand-receptor docking is mainly confined to structure optimization *after* the actual docking procedure. To our knowledge, there is only a single program, ICM,[4] that utilizes gradient based minimization for ligand receptor docking.[4] Since ICM is a commercial software, the authors obviously do not want to unveil any details of their approach. Additionally, they do not give any

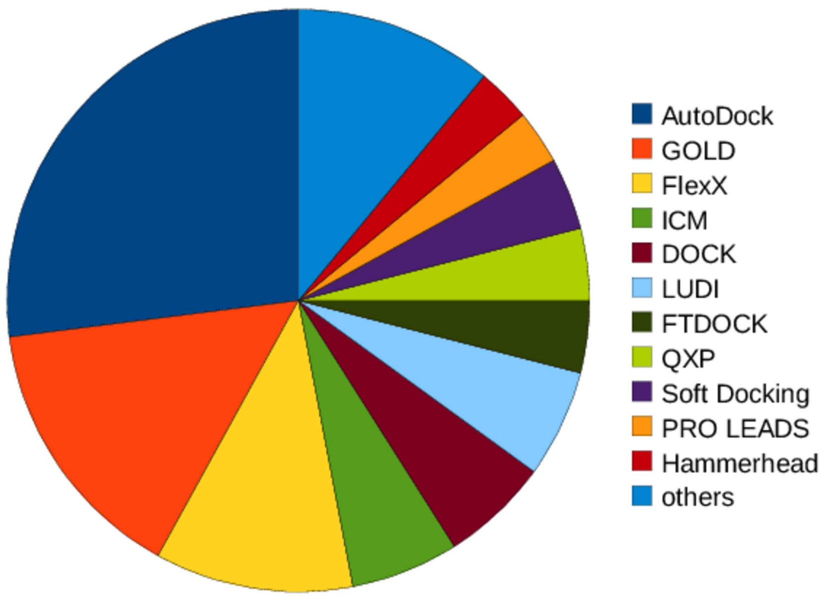


Figure 3.1.: Popularity of different docking programs in terms of number of citations.[14]

information on how gradient based minimization influences the performance of their program. We tried to re-implement the ICM approach of virtual atoms and internal coordinates to handle multiple molecules, but the interactivity of its parameters made it impossible to produce reliable results, suitable for a comparison to our approach, using translational and orientational gradient information, which is unprecedented, at least to our knowledge, in the computational chemistry.

3.3. Ligand-receptor docking

Although a vast number of docking programs was published in the last decades with each one trying to set itself apart from its competitors by employing a different approach to the optimization method of scoring function, there have only been few that achieved a widespread distribution, based on the number of citations in journals (Fig. 3.1).

According to those numbers, FlexX,[15] GOLD[16] and AUTODOCK[17] make up for more than 50% of all published docking applications. While the latter two utilize a genetic and Lamarckian genetic algorithm respectively, similar to the approach used in this work, FlexX uses geometric hashing to position a fragment of the ligand with a subsequent incremental construction. While GOLD and AUTODOCK deliver good results in terms of RMSD to the native binding pose, FlexX, while not much worse in this respect, is highly renowned for its short running time, making it the method of choice for high throughput experiments.

4. Materials and Methods

Nonlinear programming[18] is a key challenge in computational chemistry (e.g. structure optimization), economy (e.g. minimum cost transportation) or engineering (e.g. efficient aerodynamics). In the following we will present two related problems of nonlinear programming, i.e. finding the global and local optimum of a cost function.

4.1. Nonlinear optimization

4.1.1. Global optimization

Finding the minimum of a given function f is a central task in mathematics.

$$\min f : \mathbb{R}^n \mapsto \mathbb{R}$$

means, that we want to find x in $\mathbb{R}$, such that $f(x) \leq f(y)$ for every y in $\mathbb{R}$ (Fig. 4.1). Without loss of generality, we use minimization synonymously for optimization, since every maximization problem can be transformed into a minimization problem by negating the underlying, so called objective function.

Unfortunately, there is no method to this day, that guarantees to find the global minimum for any function in acceptable time and every approach that tries to address this task evolves to some kind of exhaustive search.[19] Of course, exhaustive search is not possible in $\mathbb{R}$ but since in a computational environment, every real number is represented by a finite set of bits, we could try to test every possible state and at the end present the global minimum. The drawback of this method is the enormous number of possible states. Even if we constrain our search to a single dimension with values ranging from 0 to 10, a single precision float employs 24 bits for that range, which yields more than sixteen million (2^{24}) different numbers. Although this number may seem to be large but still manageable, it must be kept in mind that most practical objective functions require much higher dimensionality. In the case of ligand-receptor docking, for a ligand of medium complexity with only four torsional flexible angles yielding ten degrees of freedom,

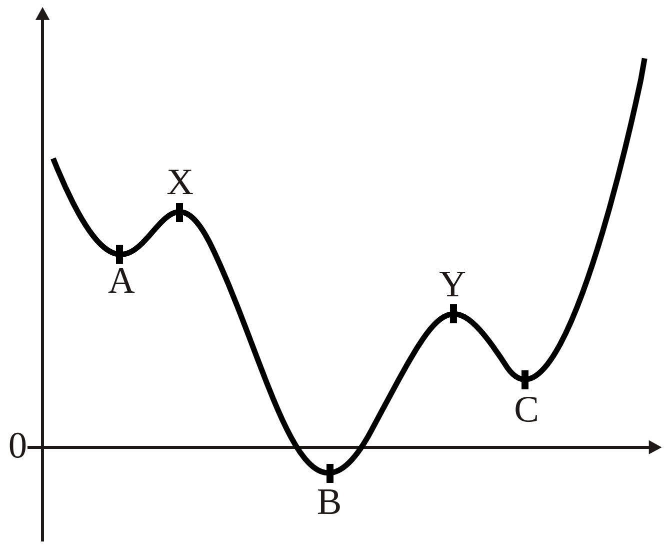


Figure 4.1.: Minima of a one-dimensional real valued function. A and C are local minima while B is the global minimum.

the number of states would be about $2^{24 \cdot 10} \approx 10^{72}$. Even if we employ a very fast scoring function (e.g. 1000 evaluations/s), it would take 10^{61} years to obtain a solution. This time-frame is about ten times larger than the presumed remaining life expectancy of the universe.

If there is no exact algorithm available for a particular problem, or if its running time is impracticable, heuristic search methods are often successfully applied.[20] A heuristic is a kind of recipe or guidance how to work on an optimization problem neither allowing any assumptions of the quality of its solution nor of the running time. However, heuristic search methods often produce good results in short time. In the diverse family of heuristic methods, the subset of meta-heuristics possess a unique feature. In contrast to a heuristic, that is specific for one or a few applications, meta-heuristics are applicable to a virtually infinite number of optimization problems. They are often called black-box algorithms since they don't necessitate a deeper insight in the problem's nature but require only a one-dimensional score to compare the quality of different solutions to the problem.

A meta-heuristic can work on a set of integer variables, real variables, a mix of both or something completely different like graphs or bit-strings. One individual solution can at the same time be interpreted as a binding pose in ligand receptor docking, an instance of an arbitrary non-linear function or define the behavior of traffic lights.

4.1.2. Local optimization

While there is no practical method, that guarantees to approximate the *global optimum* of an arbitrary function, numerical methods are able to find a *local optimum*, a point in search space that is optimal in its neighborhood. For example, we could take the Alps as the potential hypersurface of a real valued 2-dimensional function. Finding the local minimum of an arbitrary coordinate roughly corresponds to following the trace of a sphere, rolling downhill to the deepest point of a valley. In Fig. 4.1, a method for local optimization ought to converge to minimum A, starting from any point left from X, to minimum B, starting from any point right from X and left from Y and to minimum C from any point right from Y. The finding, that the local minimum B is identical to the global minimum suggests, that local optimization methods can be more than helpful in global optimization.

In local minimization, we have to distinguish local from global methods. While the latter guarantee to approach the local minimum, this is not true for local methods, that require the initial position to be sufficiently close to a local optimum. If this is not the case, a local method might as well converge to a maximum or a saddle point.

Newton's method

One of the most efficient approaches for local optimization of real valued functions is Newton's method.[21] It requires the gradient

$$\nabla f(\mathbf{x}) = \left[\frac{\delta f}{\delta x_1}, \frac{\delta f}{\delta x_2}, \dots, \frac{\delta f}{\delta x_n}\right]$$

and the Hessian matrix

$$H = \begin{bmatrix} \frac{\delta^2 f}{\delta x_1^2} & \frac{\delta^2 f}{\delta x_1 \delta x_2} & \cdots & \frac{\delta^2 f}{\delta x_1 \delta x_n} \\ \frac{\delta^2 f}{\delta x_2 \delta x_1} & \frac{\delta^2 f}{\delta x_2^2} & \cdots & \frac{\delta^2 f}{\delta x_2 \delta x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\delta^2 f}{\delta x_n \delta x_1} & \frac{\delta^2 f}{\delta x_n \delta x_2} & \cdots & \frac{\delta^2 f}{\delta x_n^2} \end{bmatrix}.$$

By replacing the current position x_k, using the Newton step, by

$$x_{k+1} = x_k - (H(f(x_k)))^{-1} \nabla f(x_k),$$

Newton's method can find the minimum of a quadratic function in one step, if the start position is sufficiently close to the minimum. Of course, one seldom tries to optimize purely quadratic functions, but even in the general case, Newton's method converges rapidly.

Quasi-Newton approach

The quasi-Newton approach[22] avoids two problems of the original Newton's method: Computing the inverse of the Hessian matrix in each step is often not computationally feasible. Thus, the Hessian matrix is approximated by previous steps using only gradient information. Additionally, Newton's method converges only locally, i.e. to a stationary point, e.g. a maximum. Therefore, the quasi-Newton approach uses a globally convergent method to get sufficiently close to a minimum to apply Newton steps.

The general approach of a quasi-Newton method for the minimization of a real valued function F is given by:

1. Compute $\nabla f(x_k)$ and an approximation to H_k.
2. Remove possible ill-conditionedness of H_k by appropriate perturbation.
3. Solve $H_k s_n^k = \nabla f(x_k)$.
4. Take Newton step or determine x_{k+1} by global strategy.

This means, that x_{k+1} is only directly computed by the Newton step, if x_k is sufficiently close to a local minimum. If this is not the case, a global method, e.g. a line search algorithm, is applied. A line search finds the local minimum of a one-dimensional function g, that is defined by

$$g(\alpha) = x_k + \alpha s_n^k.$$

One of the best quasi-Newton methods is the L-BFGS approach that calculates the approximation B_{k+1} of the Hessian matrix by

$$B_{k+1} = B_k + \frac{y_k, y_k^T}{y_k^T s_k} - \frac{B_k s_k (B_k s_k)^T}{s_k^T B_k s_k}$$

with B_k being the previous approximation to the Hessian matrix and y_k being

$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k).$$

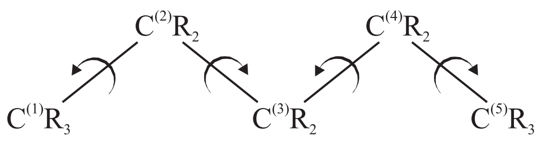


Figure 4.2.: Example for flexible bonds and molecular centroid with R being arbitrary heavy atoms. If bond $C^{(1)}$ - $C^{(2)}$ is rotated, only atoms R connected to $C^{(1)}$ are moved. If bond $C^{(2)}$ - $C^{(3)}$ is rotated, $C^{(1)}$ and atoms R connected to $C^{(1)}$ and $C^{(2)}$ are moved. Due to symmetry, the same holds true for the other two bonds with other indices. This means $C^{(2)}$, $C^{(3)}$ and $C^{(4)}$ are never moved and hence define the molecular centroid.

One additional feature of this method is, that the approximated Hessian matrix is always positive definite, i.e. it is guaranteed, that s_n^k points in a downhill direction.

Solis and Wets optimization method

The local search method of Solis and Wets[13] is a stochastic heuristic for continuous parameter spaces. Its primal purpose is the optimization of functions that do not provide gradient information, e.g. the AUTODOCK scoring function.[17] For our comparison, we closely followed the version of AUTODOCK 3.1 with the only alterations being due to adjustments to the BALL[23] environment. The basic algorithm starts with a random search step and generally follows this direction with random movements as long as the objective function keeps improving. Continued improvements lead to an expansion of the random search steps, whereas continued failing narrows the search. The algorithm iterates until either a maximum number of function evaluations is reached or convergence is established by the random step width falling below a certain threshold value.

4.2. Molecular representation

For structural optimization, we require a molecular representation that can be employed by a search algorithm. The conformation and position of a molecule in space is uniquely defined by the Cartesian coordinates of its atoms. Often the complete molecular flexibility is abandoned for of a reduced set of parameters that is required for representing a molecule. Like many other applications[16,17,24] we use a compact representation of translation, orientation, and a set of flexible bonds that connect rigid compounds. Thus, we need three real values for the translation (t_x, t_y, t_z), one real value for each flexible bond $(\phi_1, .., \phi_n)$, and a unit quaternion composed of four real values (q_1, q_2, q_3, q_4) for the molecule's orientation. A parameter vector $\mathbf{x} = (t_x, t_y, t_z, q_1, q_2, q_3, q_4, \phi_1, .., \phi_n)$ is converted into a molecular conformation by a series of transformations.

In the first step, all flexible bonds are processed. Because in our case a flexible bond is guaranteed not to be part of a ring, it divides the molecule in two substructures. The part containing fewer atoms is rotated while the other one remains stationary (Fig. 4.2). This procedure is applied to all flexible bonds.

In the next step, the whole molecule is rotated. To this end, the origin is defined by the average position of all atoms that were not rotated in the first step thus defining a form of molecular centroid. In other implementations the rotation origin is intuitively placed onto the geometric center of the ligand, but this method complicates the computation of derivatives with respect to orientational parameters.

In the last step, the molecule is moved according to the three translational parameters.

4.3. BALL

All docking methods examined in this work were implemented using the BALL library.[23] BALL is an application framework written in C++ that provides a large number of methods for computational chemistry. It was designed to be an efficient and robust tool for rapid software prototyping.

In a computational environment, the effective handling of large chemical and biological entities requires sophisticated data structures and mathematical objects, as provided by BALL (Fig. 4.3). On top of those, so called foundation classes follow kernel classes, which embody atoms, bonds, molecules, etc. Both classes are used by different components, that implement basic operations, e.g. file input/output and molecular mechanics, while the application layer provides ready-to-operate programs for docking, MD-simulations, etc. The entire code that was written in this work for the modeling of molecules, scoring functions, etc. was generated using the BALL library. Additionally, BALLDock is scheduled to be an integral part of BALL in one of the next releases.

4.4. Astex diverse set

To assess the quality of different search heuristics and scoring functions in ligand-receptor docking, we require a test set. We chose the Astex diverse set,[25] which consists of 85 high resolution protein-ligand structures. All ligands possess drug-like properties with 23 being approved drugs

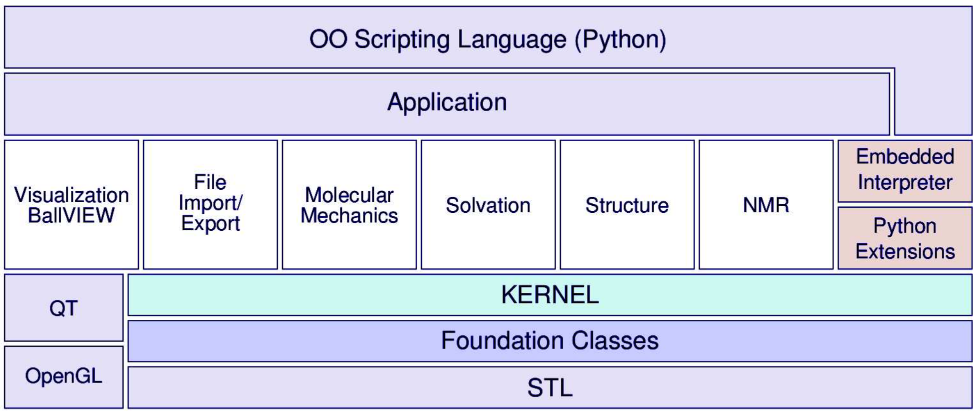


Figure 4.3.: Structure of the BALL library.

and six being in clinical trials. Fig. 4.4 displays the complexity of the ligands in terms of number of flexible torsional angles. The Astex diverse set was used to compare the different search heuristics and the two scoring functions.

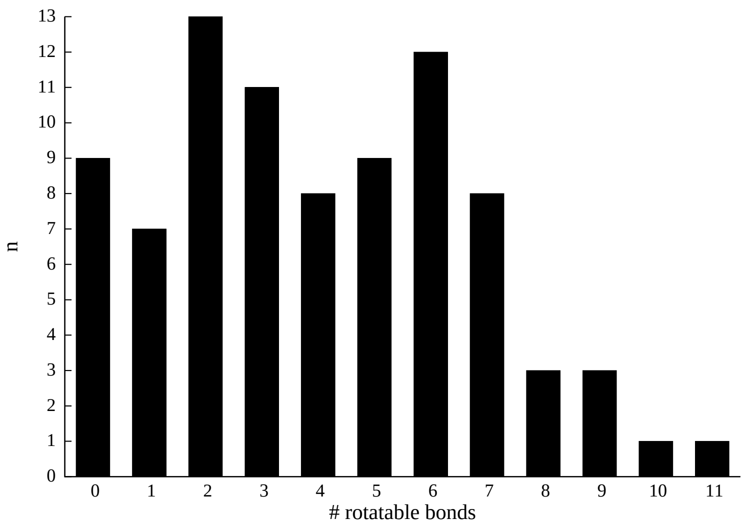


Figure 4.4.: Distribution of rotatable bonds in the Astex diverse set. For some ligands, AUTODOCK demands more bonds to be flexible because of the existence of explicit polar hydrogen atoms.

5. Population Based Meta-Heuristics for Ligand Receptor Docking

All global optimization methods compared in this work belong to the class of population based meta-heuristics. A meta-heuristic is an optimization method, that is not specific for a single problem, but is applicable to a virtually infinite number of tasks. Any arbitrary problem is only required to possess a set of parameters that enables the meta-heuristic to search the space of problem instances. Additionally, it must return a score that provides insight in the quality of a point in search space. In this context, population based algorithms try to gain gradient information by holding a certain number of trial solutions, so called individuals. This gradient information is used to produce new individuals, that have better scores and, hopefully, approach the global optimum.

Here we compare six population based meta-heuristics: four variants of the genetic algorithm,[26] differential evolution,[27] and particle swarm optimization.[28] In the following section we will briefly describe the underlying principles of the individual algorithms and their applications in molecular docking.

5.1. Genetic and Lamarckian genetic algorithm

The **genetic algorithm**[26] (GA) imitates the principles of Darwinian evolutionary theory, particularly natural selection and reproduction. It uses a set of genetic operations to drive a population iteratively toward better solutions. Fig. 5.1 describes the general schedule of a genetic algorithm. Optimization starts with the creation of an initial random population. In the next step each individual is assigned a fitness score that is used to discard the worst, i.e. least fittest members of the pool and to select the best individuals for creating progeny. Individuals that qualify to produce offspring are subject to mating to replenish the pool by producing new individuals, whereas mutation may modify existing individuals. To conserve the current best solutions, elitism is applied, which means that a number of top ranked individuals are protected from mutation. These steps

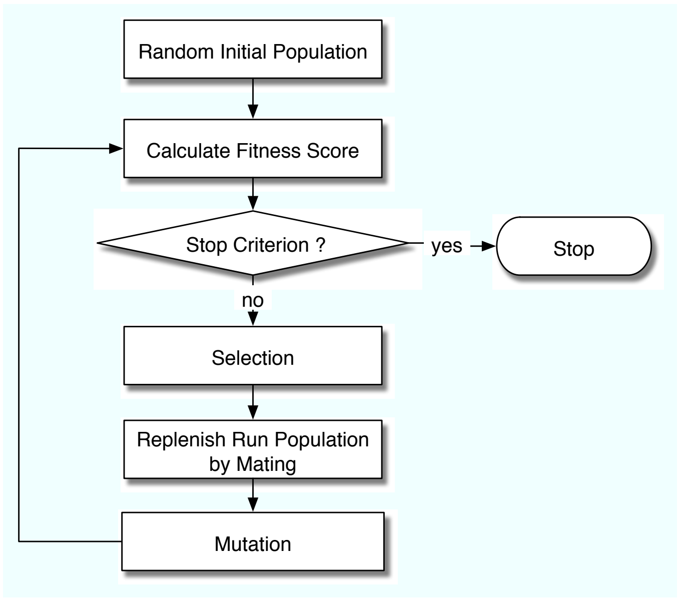


Figure 5.1.: Flowchart of a genetic algorithm

are repeated until a threshold number of iterations is reached or until a convergence criterion has been met. Originally, GA was used to solve combinatorial problems. To this end, a genetic individual has one chromosome, and the process of mating is implemented by a crossover of two chromosomes. However, for real valued functions, applying this approach to a chromosome of real values is not practical. In this case, crossing over leads only to new combinations of the existing real values without introducing new intermediate values. Therefore, for real valued problems there are often special operators.[29]

The standard GA presumably converges too fast to a local optimum, which results in the failing to find the global optimum, especially for higher dimensional search spaces, yet there are two popular modifications to the GA, the distributed genetic algorithm or **multi-deme genetic algorithm**[30] (MDGA) and the **Lamarckian genetic algorithm**[31] (LGA). In MDGA, two or more island populations evolve simultaneously and by allowing a limited migration between these populations, diversity is enforced and convergence is reached more slowly. LGA adds a local search to the GA, to increase the fitness of randomly selected individuals. On the one hand, this increases the diversity of the population and on the other hand raises the chance for finding the global optimum. By merging both modifications, we obtained the **multi-deme Lamarckian genetic algorithm** (MDLGA). Variations of GAs have been employed repeatedly for ligand-receptor docking[32–35] and are employed in the well-known programs AUTODOCK,[17,36] GOLD,[16] and in the recently developed FITTED.[37]

5.2. Differential evolution

The **differential evolution algorithm**[27] (DE) differs from GA mainly by two factors. First, it does not discard a certain proportion of the population, but replaces only existing individuals by better ones. Second, the process of creating new individuals by existing ones, corresponding to the mating process in GA, is more complex (Fig. 5.2). DE selects two individuals and calculates the difference between them. This difference, multiplied by a weighting factor, is then added to a third individual, resulting in the so called trial vector. Finally, DE chooses another individual, the base vector, and performs a crossover operation by randomly blending elements of the base and trial vector. This new vector replaces the base vector only if it features a better score. Application of DE employing the AUTODOCK scoring function to six complexes showed the great potential of this search heuristic in molecular docking.[38] In conjunction with a new scoring function resembling the Gehlhaar scoring function, DE was also used in the GEMDOCK program to dock 100 protein-ligand complexes.[39] Compared to two other commercial programs, GEMDOCK performed slightly better. In a more recent study, docking with DE was performed for a set of 77 complexes using an extended version of the Gehlhaar scoring function.[40] In comparison with commercial docking software, DE was able to identify the correct binding pose with higher accuracy.

5.3. Particle swarm optimization

Like GA and DE, **particle swarm optimization**[28] (PSO) iteratively works on a population of individuals, in this case called particles. In theory, these particles are not replaced by new ones, but, inspired by the behavior of flocking birds, are constantly moving with a velocity v in the parameter space to search the global optimum. For the computation of the new position x_{new} of a particle p, PSO calculates the difference d_1 between the current position x of p and the best position p itself encountered during optimization, as well as the difference d_2 between x and the best solution reported by the neighbors of p. In a first step, a new velocity v_{new} is calculated using two random numbers r_1 and r_2, both in the range between 0 and 1, by

$$v_{new} = v_{old} * wt + c_1 * r_1 * d_1 + c_2 * r_2 * d_2.$$

In this equation, a cognitive weight c_1, a social weight c_2 as well as an inertia weight wt define the impact of the respective contribution to the velocity.

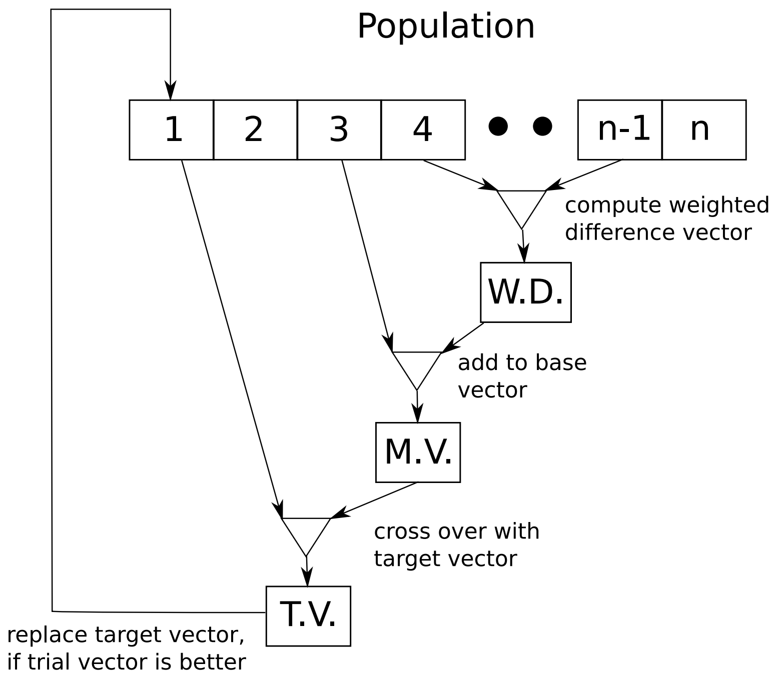

Population
1
2
3
4
n-1
n
compute weighted difference vector
W.D.
add to base vector
M.V.
cross over with target vector
T.V.
replace target vector, if trial vector is better

Figure 5.2.: General scheme of one step in a differential evolution algorithm: First, select four random population members. Adding the weighted difference (W.D.) of two vectors (4 and n-1) to the base vector (3) yields a mutation vector (M.V.). Perform cross over between mutation vector and target vector (1). The resulting trial vector (T.V.) replaces the target vector, if it has a better score.

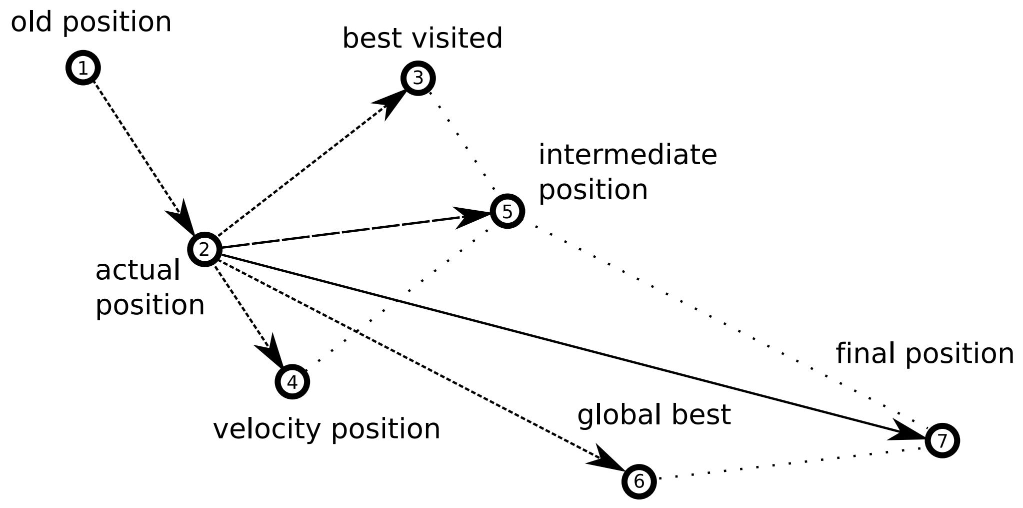


Figure 5.3.: General scheme of one step in particle swarm optimization: The final position (7) of the particle is a combination of three independent directions of motion. The velocity is defined by the particle's previous (1) and actual position (2). The other two being the best position the particle has already visited (5), and the best position the particle is able to see (6).

Then, the particle's new position x_{new} is calculated using the following formula

$$x_{new} = x + v_{new}.$$

Recently, variations of PSO were employed successfully for docking using the AUTODOCK 3.0.5 scoring function.[11,12] In comparison with the LGA implemented in AUTODOCK 3.0.5, they showed very promising results with regard to finding the native binding pose.

5.4. Implementational details

Common to all meta-heuristics presented here is the need to compute the difference of parameters to gain gradient information. This is trivial for real valued parameters used to define translation and torsional angles. To calculate the difference between two rotational angles, we consider values to form a ring. This means, that the distance between $-180°$ and $180°$ is $0°$ rather than $360°$. This enables an unlimited rotation around flexible torsional angles and no torsional angle is preferred. The same approach is used for the translational degrees of freedom, thus producing some kind of periodic boundary condition. Again this is done to prevent optimization methods from favoring ligand positions in the center of the binding pocket.

Dealing with orientations is more complex. In the GA, an offspring's orientation may be calculated from the two parental orientations by two different approaches. In a simple approach, the four values of a unit quaternion are considered to be independent from each other resulting in a linear interpolation between the two parental quaternions. However, the necessary subsequent normalization of the resulting offspring's quaternion may produce unexpected results: The difference of two unit quaternions, describing dissimilar orientations can be defined by a four-dimensional vector. Adding this vector to another unit quaternion may be without effect on the third quaternion due to the normalization. To deal with this problem, it is possible to use the SLERP[41] algorithm, allowing us to compute gradient information without any numerical singularities. Preliminary calculations showed that best results for all meta-heuristics were achieved, when linear interpolation was used for quaternions that are very similar and SLERP was used for quaternions that exhibited less similarity. The similarity of two unit quaternions q_1 and q_2 can be estimated by the scalar product. If q_1 and q_2 are identical, the scalar product yields 1 while in the case that q_1 and q_2 represent maximally different orientations, the scalar product is 0.

5.4.1. Genetic algorithm

When calculating the offspring's values, we differentiate between real valued parameters and quaternion parameters. For real valued parameters, we calculate the difference d between two parameters a and b (without loss of generality, $a < b$) and uniformly randomize the offspring value in the range a and b if d is large, or in the range $a - 0.5 \cdot d$ and $b + 0.5 \cdot d$ if it is small (Fig. 5.4). This discrimination of d was introduced to prevent completely random numbers if a and b are far apart, while allowing a broader search if a and b have similar values.

Mating of unit-quaternion parameters demands an adapted approach, because the four quaternion values are interdependent by the constraint $|q| = 1$. Again, like for simple real valued parameters, we discriminate mating of parameters that are more similar or more different. In the first case, we independently interpolate each of the four quaternion values, followed by a normalization. In the other case, we use SLERP with a uniformly randomized parameter between 0 and 1. Best results were achieved, when the threshold distance for switching from one procedure to the other was 3 Å for translation, 120° for flexible torsional angles, and 0.7 for unit quaternions.

Table 5.1 lists the parameters that were used in this work for the various genetic algorithms.

name	number of populations	initial population	population size	survivors	elitism	mutation rate
GA/LGA	1	100	200	100	1	0.05
MDGA/MDLGA	5	20	40	20	1	0.05

Table 5.1.: Parameters for the genetic algorithms.

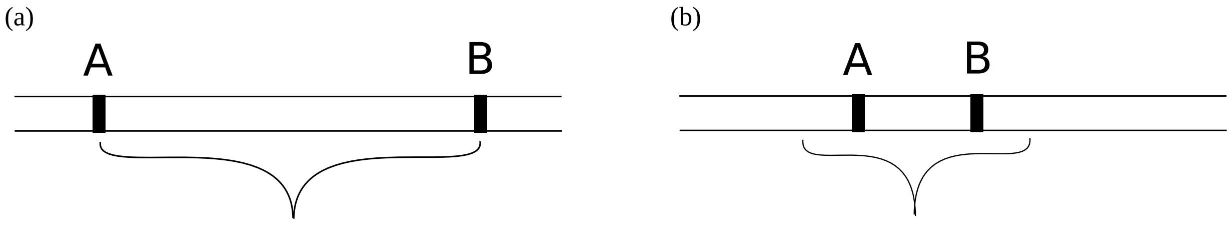


Figure 5.4.: The curly brackets indicate the range for the offspring's value if the parent's values are far apart (a) or close together (b)

5.4.2. Differential evolution

Similar to the GA, we use a special treatment for unit quaternion parameters. If the selected quaternions are similar, we use the same procedure as described above. Since differential evolution necessitates the use of more than two quaternions for calculating an individual's new orientation, employing SLERP is more complex. The computation of the trial vector can be understood as a parallelogram. In the case of unit quaternions, this parallelogram has to be mapped on the surface of a 4-dimensional sphere (Fig. 5.5). If we take the weighted difference between unit quaternion q_1 and q_2, we can slerp (in the following, we will use the word slerp to describe the application of the SLERP algorithm) from q_1 to q_2 with SLERP parameter w to get unit quaternion q_w. Then, we can slerp from q_{base} to q_w with SLERP parameter 0.5 producing q_3. Finally, we slerp from q_1 to q_3 to achieve the desired unit quaternion q_{trial}.

When testing DE, we found that it produced best results, when the weight for the difference calculation was randomized between 0 and 2 for each computation of a trial vector.

Table 5.2 contains the parameters for DE used in this work.

name	initial population	population size	crossover probability
DE	50	50	0.7

Table 5.2.: Parameters for the differential evolution algorithm.

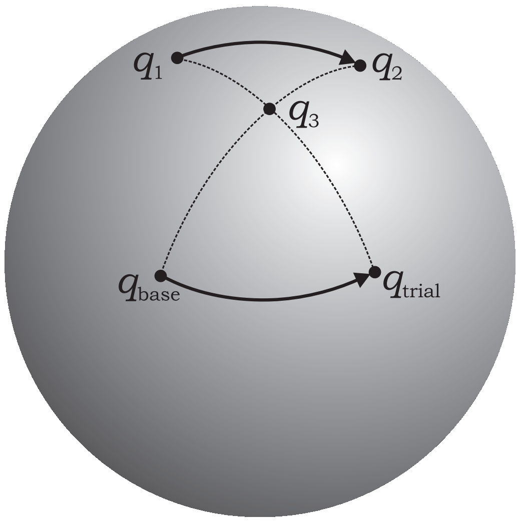


Figure 5.5.: Calculation of the unit quaternion q_{trial} in three dimensions in differential evolution. DE requires to add the difference between q_1 and q_2 to q_{base}, which is achieved by calculating q_3. For simplicity, we assume the weighting factor to be 1 in this example ($q_2 = q_w$).

name	initial population	population size	wt	c_1	c_2
PSO	75	75	0.7	2	2

Table 5.3.: Parameters for the particle swarm algorithm.

5.4.3. Particle swarm optimization

The connectivity of particles in particle swarm optimization is crucial for the convergence behavior of the method. If all particles are able to see the best global solution, the population will converge faster than with limited visibility. In preliminary docking experiments, we found, that a ring geometry produced best results. In our implementation, every individual obtained information from itself and two neighboring individuals (Fig. 5.6).

Both cognitive weight c_1 and social weight c_2 were set to 2 while the inertia weight wt was set to 0.7. Again we can use the same strategy to compute unit quaternion parameters like in DE, but this time, the parallelogram procedure has to be applied twice.

Table 5.3 contains the PSO parameters that we employed in this work.

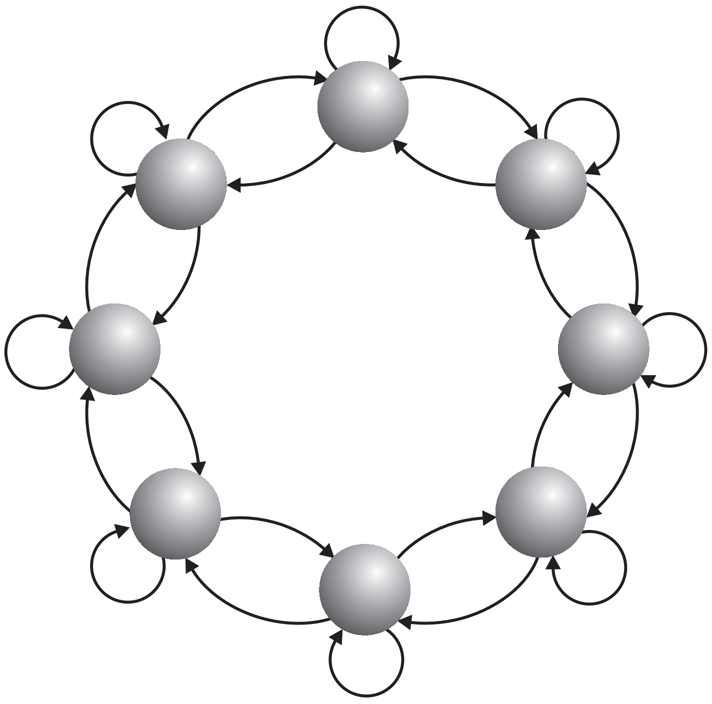


Figure 5.6.: Topology of our particle swarm optimization. Each particle gains information from itself and two neighbouring particles.

6. Scoring Functions for Ligand-Receptor Docking

A scoring function in ligand-receptor docking is expected to meet multiple requirements. In the first place, it should allow to differentiate native binding poses from decoy structures. Secondly, the score should approximate the binding free energy. Furthermore, it ought to be efficiently computable. However, recent publications suggest the usage of different functions for the reconstruction problem and for the final computation of the binding free energy.[42–45] The scoring function for the first problem is evaluated many times during a docking experiment. Therefore, it must be very fast and its global optimum should correspond to the correct binding pose. The computation of the correct binding free energy is performed by a dedicated energy function, that is applied to the predicted binding pose only once. Hence, it can incorporate much more complex terms, that prolongate the computation time.

In this work, we focus on the influence of the scoring function on reconstructing the native binding pose. The actual value of the binding free energy is not of interest, because, as mentioned before, we feel that this task should be performed by a dedicated energy function.

The Gehlhaar[46] scoring function, a piecewise linear potential function, was implemented to be an easily applicable function and to produce a less frustrated energy landscape compared to other scoring functions. It does not include electrostatic contributions and thus does not require the computation of point charges for each atom. On the other hand, it must be noted that the Gehlhaar score cannot be used to estimate the binding free energy. Comparisons of the performance of this scoring function with other scoring or energy functions showed that it performs quite well for identifying the correct pose.[47,48] In addition, since the Gehlhaar scoring function produces a rather smooth energy landscape, the performance of search heuristics may increase when employing this piecewise linear potential function or a variation thereof.[39] The Gehlhaar scoring function has been employed in a number of studies[47–49] and has been implemented in several algorithms[50] and docking programs.[39,51]

The recently revised AUTODOCK scoring function[36] is part of the widely used AUTODOCK

atom type	Donor	Acceptor	Both	Nonpolar
Donor	Steric	HB	HB	Steric
Acceptor	HB	Steric	HB	Steric
Both	HB	HB	HB	Steric
Nonpolar	Steric	Steric	Steric	Steric

Table 6.1.: Atom types for non-bonded interactions.

	A	B	C	D	E	F
Steric	3.4 Å	3.6 Å	4.5 Å	5.5 Å	-0.4	20.0
HB	2.3 Å	2.6 Å	3.1 Å	3.4 Å	-2.0	20.0

Table 6.2.: Parameter set for non-bonded steric and hydrogen-bonding potentials.

docking suite.[17] It employs 6-12 potentials for dispersion-repulsion interactions and a screened Coulomb potential for electrostatic interactions. Additionally, it features a pairwise term for hydrophobic interactions and an explicit term for directional hydrogen bonding between ligand and receptor. However, this directionality is only taken into account in the calculation of the energy grid. For the computation of the ligand's internal energy, the hydrogen bonding term is simplified for computational efficiency by neglecting the geometric contributions.

6.1. Gehlhaar scoring function

In this work we chose the Gehlhaar function[46] mainly for the following reasons: ease of implementation, sufficient correlation of the function values to the RMSD, less frustrated energy landscape compared to other scoring functions and finally as a test case for an inherently not continuously differentiable function. It must be noted that the Gehlhaar score cannot be used to estimate the binding free energy.

The formula for the score E is composed of one bonded term for the torsional potential E_{tor} and one non-bonded term E_{pair} for van der Waals interaction

$$E = E_{tor} + E_{pair}.$$

For the computation of E_{pair}, the Gehlhaar scoring function distinguishes only four atom types: non-polar, hydrogen-bond-donor, hydrogen-bond-acceptor, and both-acceptor-and-donor. The interaction between any of these atom types results in two types of non-bonded interaction, namely steric and hydrogen bond contributions (Table 6.1).

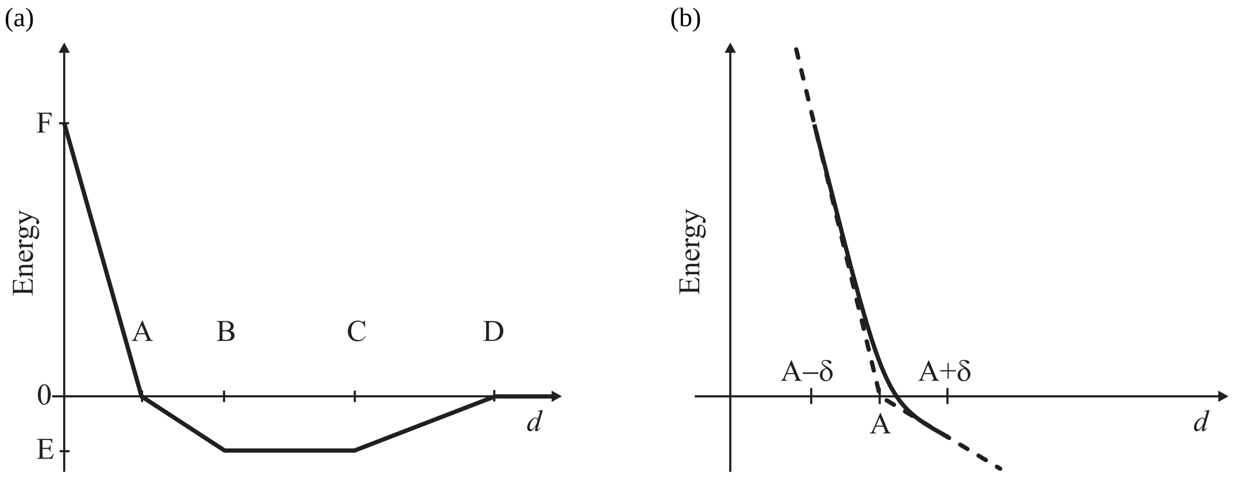


Figure 6.1.: (a) shows the original piecewise linear pairwise potential function used for non-bonded interactions. (b) illustrates the modifications (solid line) applied to the original function (dashed line) in order to produce a continuously differentiable function.

Both interaction types are calculated by an interval piecewise linear function f of the pairwise atom distance d_{ij} of atoms i and j, with each type having different function parameters (Table 6.2, Fig. 6.1)

$$E_{pair} = \sum_{i \neq j} f(d_{ij}).$$

This function is obviously not continuously differentiable so we added a quadratic transition function in an interval of 0.02 Å length at each junction of the original linear segments (Fig. 6.1). These functions are uniquely defined by their interpolation conditions.

The term for the torsional energy E_{tor} is similar to that of other scoring functions, but restricted to $sp^3 - sp^3$ and $sp^2 - sp^3$ bonds:

$$E_{tor} = A \cdot (1 + \cos(n \cdot \phi - \phi_0))$$

with $A = 3.0$, $n = 3$, $\phi_0 = \pi$ for $sp^3 - sp^3$ bonds, and $A = 1.5$, $n = 6$, $\phi_0 = 0$ for $sp^2 - sp^3$ bonds. The original Gehlhaar function provides a separate energy term for the internal non-bonded interaction of the ligand by assigning a penalty of 10^4 if two ligand atoms that do not share a bond come closer than 2.35 Å. This kind of energy calculation is entirely unsuited for the computation of a gradient for it is highly non-continuous. To circumvent this problem, we use the same term for internal ligand-ligand interactions as for ligand-receptor interactions.

6.2. AUTODOCK scoring function

The scoring function of AUTODOCK is an empirical approximation to the binding free energy that is calculated by five individual terms: two Lennard-Jones potentials for van der Waals and hydrogen bonding energies with the latter featuring a term $E(t)$ to include the bonding geometry, a Coulomb potential with a distance dependent dielectricity constant to account for a damping due to the solvent, a pairwise term for hydrophobic interactions, and finally a term for entropic effects.

$$\begin{aligned} E &= W_{vdw}\sum_{i,j}\left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}\right) + W_{hbond}\sum_{i,j}E(t)\left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}}\right) \\ &\quad + W_{elec}\sum_{i,j}\left(\frac{q_i q_j}{\varepsilon(r_{ij})r_{ij}} + W_{sol}\sum_{i,j}(S_i V_j + S_j V_i)e^{\frac{-r_{ij}^2}{2\sigma^2}}\right. \\ &\quad + W_{tor}N_{tor} \end{aligned}$$

All weights W are calibrated with experimental binding free energies. For a faster calculation of the interaction between ligand and receptor, AUTODOCK pre-calculates an energy grid, while during the docking, the scoring function is used to calculate the ligand's internal energy. Since W_{tor} is a constant value for a ligand, it is not included in the calculation of the score during a docking experiment. Additionally, $E(t)$ is only taken into account for the grid computation, while it is neglected in the computation of the ligands internal energy.

7. Comparison of Meta-Heuristics and Scoring Functions

To test the six meta-heuristics, we performed 300 docking runs for each ligand of the Astex data set with each optimization method and scoring functions. During a docking run, the ligand was allowed to move inside a translation box with an edge length of 10 Å centered on the ligand in the correct binding pose. To eliminate any possible preference for a certain orientation, we randomized the ligand's orientation before every single docking run. For each docking run we recorded the final score, the best score after each scoring function evaluation, the total number of scoring function evaluations, the RMSD to the native binding pose, and the RMSD to the target binding pose. The latter is the ligand's position with the best score, that is the presumed global optimum for this scoring function, and does not necessarily coincide with the native binding pose. If the RMSD between target and native binding pose was greater than 2 Å, the scoring function failed in finding this ligand's native pose. To separate the evaluation of the optimization methods from the scoring functions, we compared the search heuristics only in terms of RMSD to the target binding pose. In the following chapters, we define a hit to be a ligand position with an RMSD smaller than 2 Å to the target binding pose, if not specified otherwise. Finally, to assess the reliability of the results, we defined a saturation measure. If *n* is the number of hits, then we define saturation to be the number of hits in the *n* top ranked results divided by *n*. For example, if a meta-heuristic produced ten hits in one docking experiment (300 docking runs) and out of the ten top-ranked results, three were hits, the method achieved a saturation of 0.3.

To compare the running time, we applied the same stopping criterion for all methods. The algorithms stopped if the best score did not improve by a certain amount for a given number of function evaluations. This number depends linearly on the number of flexible torsional angles in the range of 3000 and 5000, while the threshold was 1 for Gehlhaar and 0.1 for AUTODOCK, respectively.

			Gehlhaar		AUTODOCK			
	name	∅ hits (%)	∅ mean score	∅ best score	∅ mean score	∅ best score	∅ function eval.	∅ saturation
0-3 rot. bonds	GA	15.88	-72.58	-98.40	-6.33	-9.04	**6352**	0.74
	MDGA	21.50	-77.47	-99.50	-6.63	-9.14	7073	0.76
	LGA	19.99	-77.71	-99.66	-6.68	-9.12	7694	0.76
	MDLGA	40.28	**-89.11**	**-99.83**	-7.58	**-9.25**	12850	0.86
	PSO	**48.89**	-88.98	-99.56	**-7.81**	-9.16	11988	0.85
	DE	35.69	-85.80	-99.81	-7.48	-9.23	13428	**0.91**
4-7 rot. bonds	GA	3.65	-80.97	-112.61	-6.25	-9.60	**8926**	0.58
	MDGA	5.72	-85.04	-116.82	-6.43	-10.07	9418	0.56
	LGA	4.89	-84.83	-115.59	-6.56	-9.86	10728	0.59
	MDLGA	12.61	**-98.12**	-121.99	-7.66	-10.67	17646	0.62
	PSO	**16.18**	-95.41	-118.57	-7.69	-10.43	17473	0.57
	DE	9.29	-96.27	**-122.76**	**-7.71**	**-10.77**	20263	**0.74**
8-11 rot. bonds	GA	0.18	-85.56	-122.95	-6.18	-10.33	**11663**	0.67
	MDGA	0.62	-89.16	-130.83	-5.93	-9.95	11948	0.19
	LGA	0.25	-88.99	-124.56	-6.34	-10.66	13987	0.67
	MDLGA	**3.25**	**-106.85**	-147.72	**-7.47**	-11.39	22436	0.53
	PSO	2.29	-100.15	-142.3	-7.07	-11.06	22459	0.33
	DE	1.47	-103.57	**-148.67**	-7.081	**-11.635**	26274	**0.70**

Table 7.1.: Comparison of the six meta-heuristics in terms of hit probability, running time, average mean and best energy as well as saturation. The results are partitioned for small, medium and large number of flexible torsional angles. Both number of functions evaluations and saturation are averaged for both scoring functions. For the calculation of the saturation, we included only ligands for which all search methods produced at least one hit to the global optimum.

7.1. Results meta-heuristics

All in all, we performed 306,000 single docking runs. The results are summarized in Table 7.1, subdivided for simple (0-4 flexible torsional angles), medium (4-7 torsional angles) and complex (more than 7 torsional angles) ligands. As expected, the number of function evaluations, required by all meta-heuristics, rises with increasing complexity of the ligand. Nonetheless, the ratio between two different algorithms remains more or less constant with GA requiring the lowest and DE the highest number. It is evident, that GA, MDGA and LGA cannot keep up with the other methods in terms of hit probability and average mean score, regardless of the ligand's complexity. While the ratio between the best and the worst method considering the mean score remains almost constant for all three levels of complexity, the ratio of the hit probability drops off sharply from 0.32 to 0.05. The average best score for simple ligands is almost identical for all methods while GA, MDGA and LGA have a worse performance as ligands get more complex.

For ligands of simple and medium complexity, PSO has the best chance to find the global optimum, but for ligands of high complexity it is surpassed by the MDLGA. DE on the other hand has always the best saturation and for ligands of medium and high complexity also the best average best score. This means, that results close to global optimum are ranked higher with a greater probability than for any other method.

Fig. 7.1 displays the relative performance of all meta-heuristics as a function of the number of flexible torsional angles. For each set of ligands with a given number of torsional flexible angles, we summed up the number of hits. Dividing by the number of docking experiments produced the hit probability for each set. To take the running time into account, we divided the hit probability by the number of function evaluations. Except for ligands of high complexity, the results are very similar for all meta-heuristics, which means, the higher hit probability of some methods in Table 7.1 is simply due to a higher number of function evaluations. At first, the performance of all search heuristics decreases more or less exponentially with the number of rotatable bonds. However, for ligands with more than 7 rotatable bonds, the performance drops drastically. In accordance with the results in Table 7.1, DE seems to be less effective than the other search strategies for small ligands, but its relative performance improves with increasing ligand flexibility.

7.2. Results scoring functions

As mentioned above, a scoring function should allow the reconstruction of the native binding pose and the calculation of the binding free energy. Recent publications[42–45] proposed the application of a fast, simple scoring function for the reconstruction of the binding pose, while a more sophisticated function is used for the calculation of the actual binding free energy. In this work, we focus on the reconstruction problem using the Astex diverse set as a test case. The comparison of the Gehlhaar and AUTODOCK scoring functions is mainly based on the RMSD between the top scored outputs of the docking experiments and the native binding poses. To test the selectivity, we calculated the ratio r between the score h of the best ranked hit to the native binding pose and the score m of the best ranked miss:

$$r = \begin{cases} \frac{(h-m)}{h} & \text{if } h > m \\ -\frac{(m-h)}{m} & \text{if } h < m \end{cases} . \tag{7.1}$$

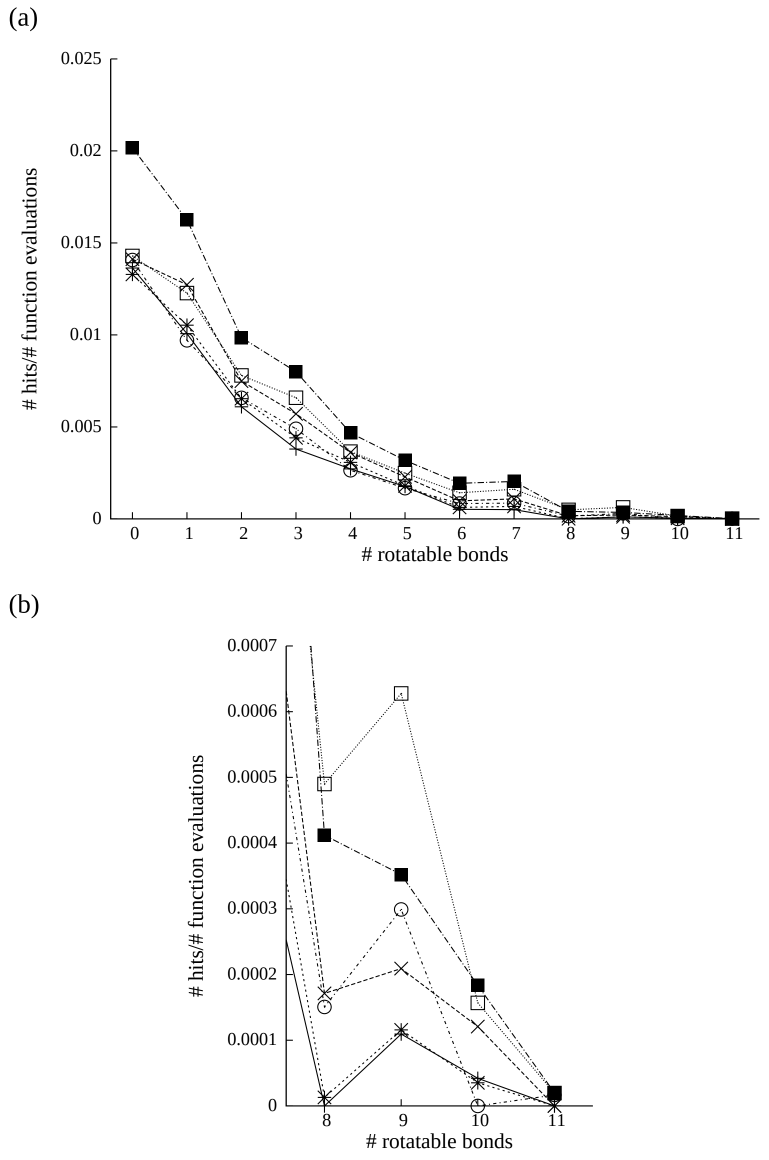


Figure 7.1.: Comparison of the six meta-heuristics in respect of hit probability normalized by the mean number of scoring function evaluations: Genetic algorithm $+$, multi-deme genetic algorithm $\times$, Lamarckian genetic algorithm $*$, multi-deme Lamarckian genetic algorithm $\square$, particle swarm $\blacksquare$, and differential evolution $\circ$.

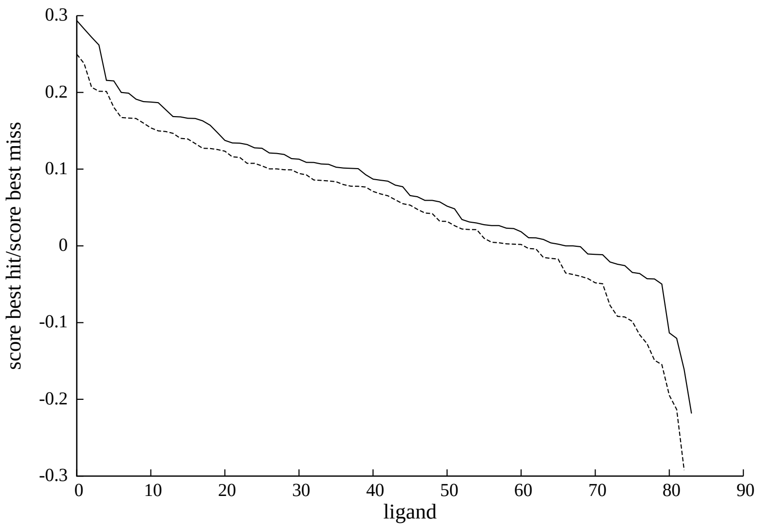


Figure 7.2.: Comparison of the Gehlhaar scoring function (solid line) and the AUTODOCK scoring function. Values were computed using Eq. 7.1.

If *r* is positive, the native binding position was scored better, if *r* is negative, the decoy position was preferred.

For the Astex set, the Gehlhaar scoring function scored 67 native positions (79%) better than any decoy positions, while for the AUTODOCK scoring function this was only the case for 62 ligands (73%). Fig. 7.2 illustrates the relative score calculated using Eq. 7.1. The smaller number of ligands for the AUTODOCK scoring function is caused by only including ligands, for which the native binding pose was hit at least once. However, it must be noted that all results were ordered for each scoring function individually. This means, that the same ligand is not necessarily found at the same position in the two curves. Obviously, the Gehlhaar scoring function produced better results by a considerable margin. For one ligand, the AUTODOCK score of the native binding pose was 30% worse than a decoy position.

Furthermore, to analyze the complexity of the search space generated by both scoring functions, we tested how often the search methods were able to find the global minimum, i.e. producing results with an RMSD of at most 2 Å to the target position. Again, the Gehlhaar scoring function proved to be superior also in this respect: on average, it produced 20% more hits. For 24 ligands, however, the AUTODOCK scoring function had the edge.

Finally, we analyzed the correlation between the Gehlhaar and AUTODOCK scoring functions. For this purpose, we collected the best scored hits to the native binding pose for all ligands for each scoring function. To account for the different values of the scoring functions, we normalized

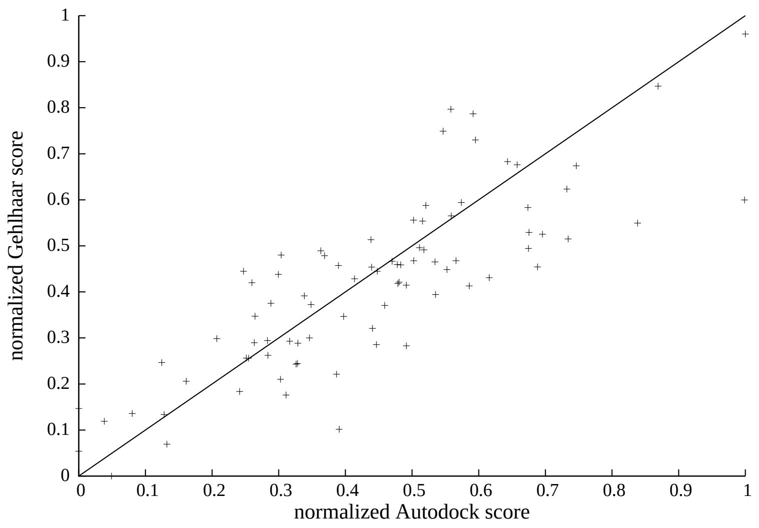


Figure 7.3.: Correlation of normalized Gehlaar and AUTODOCK scores.

the scores of each compound x to a range between one and zero for both scoring functions as follows:[52]

$$S'(x) = \frac{S(x) - S_{\min}}{S_{\max} - S_{\max}} \qquad (7.2)$$

where $S_{\max}$ is the maximum and $S_{\min}$ the minimum value, respectively, of all $S(x)$ computed with the scoring function for which the values are to be normalized.

The correlation between both functions is displayed in Fig. 7.3. In the ideal case, all results would be situated on the line of identity, which is apparently not the case, but for most ligands, there seems to be a good correlation with just a few outliers.

7.3. Discussion

Fig. 7.1 indicates that all meta-heuristics perform similar until ligands and hence the search space get very complex. There is always a trade-off when methods perform better in one respect than others. For example, the most simple GA requires the smallest number of function evaluations, but has also the lowest chance of a hit. When a ligand is highly flexible with many rotatable bonds, the chance of a hit approximates 0 while other, slower converging methods still succeed. To increase the diversity, we employed the Lamarckian GA, the multi-deme GA, and a combination of these two. The number of hits increased when premature convergence of the GA method was inhibited either by using multiple populations (MDGA) or by employing a local search (LGA). Combining both LGA and MDGA in the MDLGA led to an even higher increase in the number of hits and consequently to a higher probability to find the global optimum. Thus, the performance of the GA can be enhanced considerably by slight modifications to the original search method. Other methods to increase the diversity in GAs, e.g. the usage of two different genders[53] or employing a diploid GA where each individual possesses two chromosomes,[54] were not tested in this study.

PSO always performs well when it comes to find near-optimal structures but fails to explore deep valleys in the energy hyper-surface. Therefore, PSO does not produce reliable scores, making it hard to compare the results in terms of their score. Here, an additional local optimization might help to distinguish between true hits and decoy structures. For example, using a time-decreasing inertia weight in PSO allows for global exploration of the landscape in early stages, while the algorithm will primarily perform a local search later on. Both a time-decreasing inertia weight and a local search were implemented for docking with PSO in AUTODOCK and showed quite promising results.[11,12] However, employing local search to enhance the performance may require a longer execution time,[12] which is well in agreement with our results.

The more recently developed differential evolution always had a slightly lower chance to find the global optimum, compared to MDLGA and PSO. Although DE requires more function evaluations, the high values for saturation and average best energy indicate, that in marked difference to PSO, it dedicates more effort to explore a valley in the energy hyper-surface to the lowest point.

All in all, for simple and complex ligands, we would consider the Lamarckian GA with multiple populations to be the best trade-off out of the six tested candidates. It is especially striking that neither the Lamarckian nor the multi-deme modification alone led to significant changes in the algorithm's performance. Nonetheless, the combination of both seems to produce an algorithm, that has a high chance to find the global optimum with reliable scores without demanding an extensive number of function evaluations.

Both scoring functions rank the native (or near-native) conformation better than other positions in most cases and are in the range of good quality docking protocols of 70–80%.[25] However, our results strongly indicate that the additional effort, the AUTODOCK scoring function is employing for calculating the score, is not justified. The Gehlhaar scoring function does not only provide a less frustrated energy surface that facilitates the detection of the global optimum,[55,56] it also scores more native binding poses correctly. This may be due to the fact, that the AUTODOCK scoring function is trained to calculate the binding free energy, disregarding its central task in the reconstruction problem: separating native binding poses from decoy positions.

7.4. Conclusion

For the reconstruction problem, the simple Gehlhaar scoring functions seems to be better suited than the AUTODOCK scoring function. Of course, the Gehlhaar scoring function is not able to rank a native binding pose better than a decoy position in all cases, neither does it allow to estimate the binding free energy. Nonetheless, we suggest to use a simple scoring function, like Gehlhaar, for the reconstruction problem with a search method that does not only deliver the best ranked position but a clustered set of possible binding poses that are afterwards processed by a dedicated energy function.[47,57] These energy functions should be able to detect and quantify small differences between complexes that may change the binding free energy between ligand and receptor dramatically.[58] Such re-scoring functions should also allow for comparing the results for two different ligands to the same protein in terms of affinity.

The refinement of existing population-based search methods and the development of new algorithms has increased the chances of success in docking studies substantially. Although all meta-heuristics employed in this study may be further refined or adopted to the docking problem itself, it seems unlikely that such remedies will lead to marked changes in relative performance. Nevertheless, it is certainly necessary to improve the absolute performance of search algorithms for two reasons: firstly, all search heuristics showed a drop in performance with an increasing number of rotatable bonds. Thus, highly flexible ligands are not easily amenable to docking. Secondly, the docking and sampling performance in non-native docking is reduced considerably in comparison to native-docking.[59]

Out of the tested meta-heuristics, we cannot make a definitive recommendation. We think, that instead of testing a bulk of different heuristics, it is more worthwhile to choose one and adapt the parameters properly. There seems to exist a performance ceiling, caused by the limited information provided by the scoring functions: regardless of the dimensionality of the optimization problem, the analyzed scoring functions return a one-dimensional score.

8. Orientational Gradient

A number of meta-heuristics used for ligand receptor docking, like the Lamarckian Genetic Algorithm,[60] try to improve their results by performing local optimization. These methods can be classified into two distinct categories: approaches that need only function values and methods utilizing the function's derivatives. The first class can be subdivided into deterministic algorithms (e.g., Powell algorithm,[3] Simplex algorithm[61]), and stochastic methods like the algorithm of Solis and Wets.[13] The approaches of the second class benefit from employing derivatives of the objective function[22] and are expected to find better results faster, e.g. "deeper minima" requiring shorter time. Therefore, these approaches are preferable whenever useful derivative information is available.

Nonetheless, the methods of the first class, especially the approach of Solis & Wets,[13] are widely used in docking applications. There are two main reasons for using these methods:

1. In practice, many scoring functions, especially non force field based functions, are continuous but not differentiable. For these functions, non-gradient based techniques of the first class seem favorable.

2. Stochastic methods like the Solis & Wets approach[13] are easily adapted to specific optimization tasks.

On the other hand, the gradient based methods of the second class are restricted to differentiable objective functions. Furthermore, they are sensitive to singularities, like a loss of degree of freedom (DOF), or to non-minimal parametrizations, which is a major issue when it comes to calculating an orientational gradient. In the following chapter, we describe a way to map the space derivatives of the smoothed Gehlhhar scoring function to the parameters of our compact representation that, while not able to eliminate a loss of degree of freedom, is at least able to avoid it.

8.1. Gradient computation

The application of a gradient based optimizer requires the derivatives of the underlying energy or scoring function E with respect to the parameter vector $\mathbf{x}$. The Gehlhaar function[46] consists of a pairwise term E_{pair} and a torsional term E_{tor}. Hence, the gradient $\mathbf{g}$ is given by

$$\mathbf{g} := \frac{\partial E}{\partial \mathbf{x}} = \frac{\partial (E_{pair} + E_{tor})}{\partial \mathbf{x}}.$$

The gradient of E_{tor} can be easily computed and affects only torsional parameters $\phi_1, \ldots, \phi_n$

$$\begin{aligned} \frac{\partial E_{tor}}{\partial \phi} &= \frac{A \cdot (1 + \cos(n \cdot \phi - \phi_0))}{\partial \phi} \\ &= -n \cdot A \cdot \sin(n \cdot \phi - \phi_0). \end{aligned}$$

To calculate the derivatives for the pairwise interactions ∂E_{pair}, we first compute the gradient $\mathbf{g}_i$ for each atom i. This is the sum of all derivatives of pairwise interactions that an atom participates in with $\mathbf{v}_i$ being the position vector of atom i and $\mathbf{v}_j$ being the position vector of the interacting atom j

$$\mathbf{g}_i = \sum_{j \neq i} f'(d_{ij}) \frac{\mathbf{v}_i - \mathbf{v}_j}{\|\mathbf{v}_i - \mathbf{v}_j\|}.$$

Mapping the gradient $\mathbf{g}_i$ of an atom i with position $\mathbf{v}_i$ to an arbitrary parameter r requires the derivative of $\mathbf{v}_i$ with respect to r. $\partial \mathbf{v}_i$ represents the tangential movement of atom i when r varies by an infinitesimal amount and can now be used to calculate the derivative of E_{pair} with respect to r

$$\frac{\partial E_{pair}}{\partial r} = \sum_i \left(\frac{\partial \mathbf{v}_i}{\partial r} \right)^T \mathbf{g}_i. \tag{8.1}$$

In the following section, we will use Eq. 8.1 to calculate the derivatives of E with respect to specific parameters.

8.1.1. Translational gradient

Calculating $\partial \mathbf{v}_i$ with respect to a translational parameter t is straightforward because any change in t translates $\mathbf{v}_i$ linearly. Thus, for any translational parameter t, Eq. 8.1 can be reduced to

$$\frac{\partial E_{pair}}{\partial t_x} = (1, 0, 0) \cdot \sum_i \mathbf{g}_i,$$

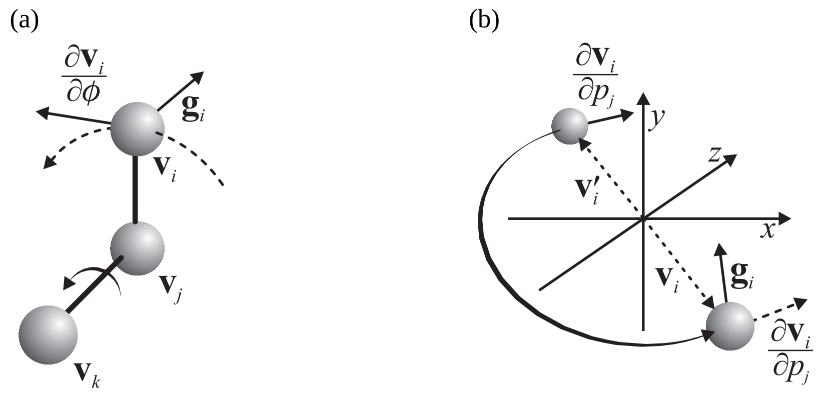

(a)
$\frac{\partial \mathbf{v}_i}{\partial \phi}$
$\mathbf{g}_i$
$\mathbf{v}_i$
$\mathbf{v}_j$
$\mathbf{v}_k$
(b)
$\frac{\partial \mathbf{v}_i}{\partial p_j}$
y
z
x
$\mathbf{v}_i'$
$\mathbf{v}_i$
$\mathbf{g}_i$
$\frac{\partial \mathbf{v}_i}{\partial p_j}$

Figure 8.1.: Mapping of non-bonded gradient to torsional (a) and orientational parameter (b).

$$\frac{\partial E_{pair}}{\partial t_y} = (0,1,0) \cdot \sum_i \mathbf{g}_i,$$
$$\frac{\partial E_{pair}}{\partial t_z} = (0,0,1) \cdot \sum_i \mathbf{g}_i.$$

8.1.2. Torsional gradient

The rapid computation of the torsional gradient has been the subject of numerous scientific studies.[62] If atoms i and j are connected by a flexible bond and atom i is moved by rotating this bond (Fig. 8.1(a)), the derivative of $\mathbf{v}_i$ with respect to a torsional parameter ϕ can be calculated by

$$\frac{\partial \mathbf{v}_i}{\partial \phi} = (\mathbf{v}_k - \mathbf{v}_j) \times (\mathbf{v}_i - \mathbf{v}_j). \tag{8.2}$$

Inserting (8.2) in (8.1) yields

$$\frac{\partial E_{pair}}{\partial \phi} = \sum_i ((\mathbf{v}_k - \mathbf{v}_j) \times (\mathbf{v}_i - \mathbf{v}_j))^T \mathbf{g}_i.$$

8.1.3. Orientational gradient

The most challenging part is the computation of the orientational gradient, because, up to now, there is no minimal representation that does not inherit some kind of singularity, e.g. loss of DOFs. Representing the orientation (three DOF) by an unit quaternion does not include such a singularity, but the independent optimization of its four values is awkward.[63] This is caused by the unit quaternions representing only a subset of the entire four-dimensional quaternion space. To alleviate this problem, we use exponential mapping[64] to map a point $\mathbf{p} = (p_1, p_2, p_3)$ from parameter space $\mathbb{R}^3$ to q in the unit quaternion space S^3:

$$q = \begin{cases} (0,0,0,1) & \text{if } \mathbf{p} = (0,0,0) \\ (\sin(0.5\|\mathbf{p}\|)\frac{\mathbf{p}}{\|\mathbf{p}\|},\ \cos(0.5\|\mathbf{p}\|)) & \text{otherwise} \end{cases}.$$

This enables us to compute the derivative of the corresponding rotation matrix $\mathbf{T}_j = \frac{\partial \mathbf{R}}{\partial p_j}$ for each of the three orientation parameters p_j[64] that can now be used to calculate the gradient $\frac{\partial E_{pair}}{\partial p_j}$. For each evaluation of the objective function, the orientational parameter $\mathbf{p}$ is mapped to an unit quaternion q. q is then converted to a rotation matrix $\mathbf{R}$ that defines the molecular orientation. Let $\mathbf{v}'_i$ be the position of an arbitrary atom i and $\mathbf{v}_i$ the position of the atom after the rotation by $\mathbf{R}$ (Fig. 8.1(b)). Then,

$$\frac{\partial \mathbf{v}_i}{\partial p_j} = \mathbf{T}_j \mathbf{v}'_i.$$

Again inserting in Eq. 8.1 yields

$$\frac{\partial E_{pair}}{\partial p_j} = \sum_i (\mathbf{T}_j \mathbf{v}'_i)^T \mathbf{g}_i, \quad j = 1,2,3.$$

As mentioned before, no method for a minimal parametrization of the orientation is free of singularities. This also holds for exponential mapping, where singularities arise if the length of the orientational parameter vector $\mathbf{p}$ approaches 2π. All parameter vectors $\mathbf{p}$ with $\|\mathbf{p}\| = n \cdot 2\pi, n \in \mathbb{Z}_{>0}$ are mapped to the quaternion $q = (0,0,0,-1)$. For these parameter vectors, all gradients $\frac{\partial \mathbf{v}_i}{\partial p_j}$ point into the same direction, reducing the number of DOFs to one. Fortunately, all possible orientations can be denoted by parameters within a shell of π around the origin in $\mathbb{R}^3$. Thus, we only need to take care that the optimization algorithm stays within this shell.

8.2. Results

To compare our approach to the method of Solis & Wets, we used both methods to optimize the randomly chosen positions and conformations of the prepared ligands (start conformations). Table 8.1 shows the average Gehlhaar-score of 500 minimizations together with the average number of evaluations required to reach a function value at most 1.0 worse than the final score. The number of rotatable bonds corresponds roughly to the complexity of the optimization problem

				our method		Solis & Wets	
PDB ID	Ref.	flexible bonds/ heavy atoms	initial score	score	number of evaluations	score	number of evaluations
1FDS	[65]	0/20	232.5	−50.4	9.8	−12.4	39.4
1FMO	[66]	2/19	295.7	−56.6	21.9	0.5	46.6
2MCP	[67]	3/11	199.2	−30.8	16.8	−11.8	29.1
1DWD	[68]	8/37	714.1	−68.9	34.1	88.5	48.4
1HPV	[69]	9/35	627.3	−75.1	36.0	107.6	69.9
2R04	[70]	10/25	770.7	−19.8	62.5	230.7	51.4
1HTF	[71]	12/41	693.3	−65.9	38.9	117.4	58.2

Table 8.1.: Comparison of our method to Solis & Wets in terms of average initial and final score and average number of function evaluations.

while the average energy before optimization indicates that generally the ligand has multiple van der Waals clashes at the random initial position. The results show that, on average, the score of our method is well below 0 for all ligands. This means that it generally resolves all van der Waals clashes and moves the molecule in a way that it is able to form multiple interactions. Even for more complex ligands, representing more difficult optimization problems, the average score does not deteriorate and seems to be roughly corresponding to the number of heavy atoms. As expected, more complex ligands require more function evaluations to reach the local minimum. In contrast to that, the method of Solis & Wets is able to resolve van der Waals clashes only for simple ligands with both average score and average number of function evaluations being considerably worse compared to our method. As ligands get more complex, the approach of Solis & Wets fails to resolve van der Waals clashes and the scores decline considerably.

There seems to be one outlier, 2R04, for which the results are worse then expected. This is caused by the particular morphology of the binding pocket, which forms a longish tube inside the receptor and is located near the receptor surface. Thus, the likewise elongated ligand can be trapped with one part being situated in the binding pocket and the other outside the receptor while the center is penetrating the protein producing multiple van der Waals clashes (Fig. 8.2).

Fig. 8.3 illustrates the performance difference of both methods and the non-deterministic character of the approach of Solis & Wets for 1DWD. In this case, all minimizations started from the same initial position. Our method always converged to a score of -108.49 (solid line) while the best result out of 100 Solis & Wets minimizations was -69.50 (dashed line). On average, the approach of Solis & Wets reached a value of 76.74 (the dotted line shows a typical minimization). The method of Solis & Wets required 149 function evaluations to produce its best results, a value that was reached by our method with only 17 function evaluations.

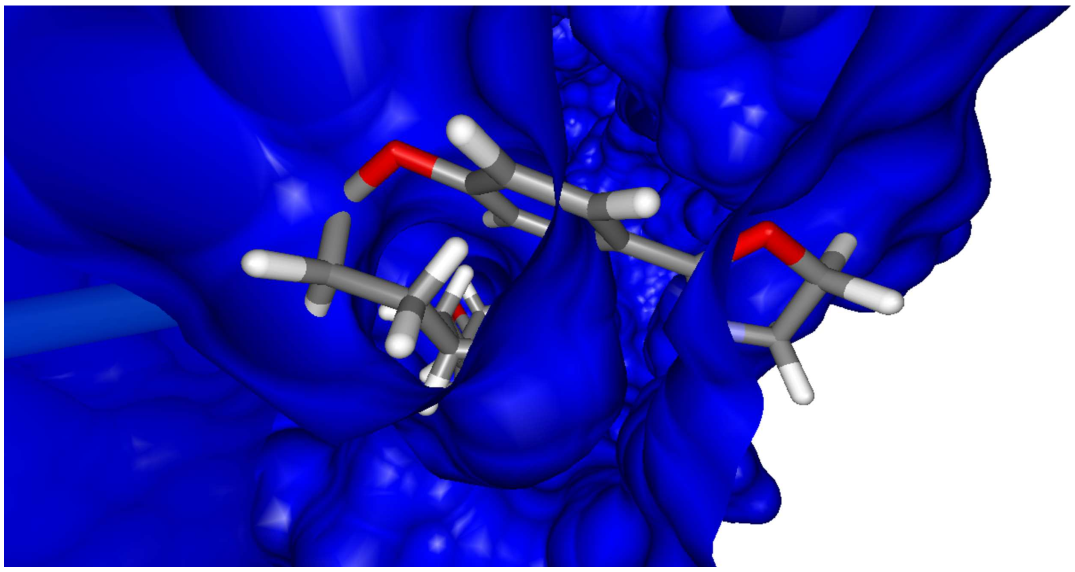


Figure 8.2.: Example for a high energy local minimum. The larger part of the docked ligand 2R04 is situated in the binding pocket on the left while the smaller part penetrates the surface.

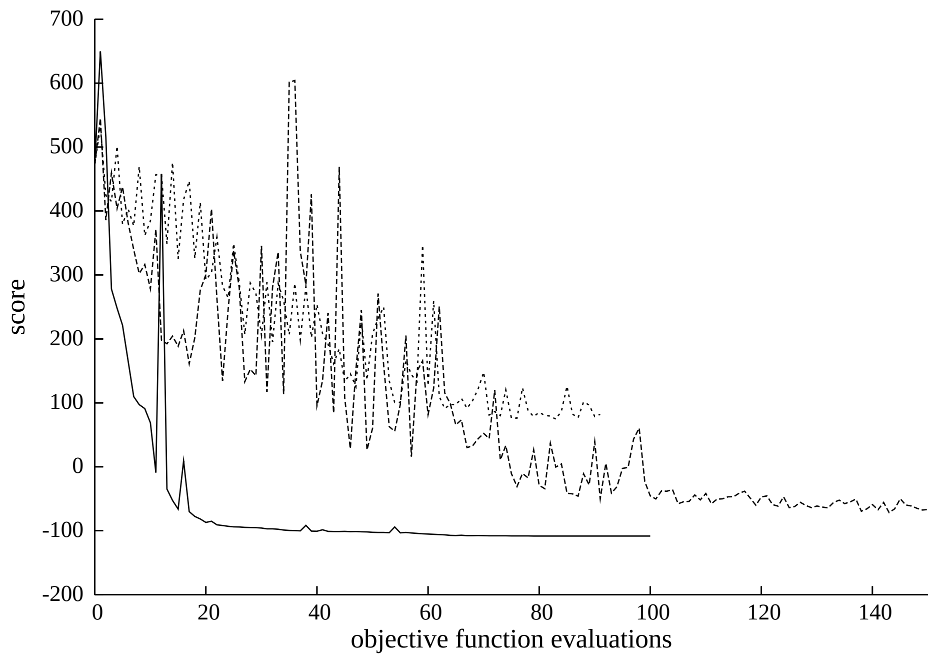


Figure 8.3.: Comparison of one deterministic minimization of our method (solid line) to two different minimizations of Solis & Wets from the same initial position (PDB ID 1DWD). The dashed line is the best result of the approach of Solis & Wets out of 100 minimizations.

8.3. Conclusion

Our results suggest that the effort to make a scoring function differentiable is worthwhile. When it comes to minimization of molecules that are represented by translation, orientation and torsional angles, the approach of Solis & Wets[13] has become a quasi standard procedure. We think that every global optimization method like e.g. the Lamarckian Genetic Algorithm[72] that utilizes the algorithm of Solis & Wets for local optimization will benefit when our method is used instead.

9. Gradient Based Minimization in a Lamarckian Genetic Algorithm

All meta-heuristics analyzed to this point implicitly gained gradient information of the underlying scoring function by evaluating the objective function on stochastically chosen points in the parameter space. The last chapter emphasized the impressive performance of gradient based minimization in ligand-receptor docking. Encouraged by these results, we created a new multi-deme Lamarckian genetic algorithm by replacing the local search procedure of Solis & Wets by our gradient based method (MDLGAGR). In this chapter, we want to analyze, if the improvement in the local search procedure by introducing explicit gradient information, is reflected in the performance of the utilizing global search heuristic. Therefore, we compared the new gradient based Lamarckian genetic algorithm to the standard one with Solis & Wets local search as well as to differential evolution and particle swarm optimization, that do not use any kind of local optimization (Fig. 9.1). For the actual comparison, we used the same approach like in Chapter 7, which permitted us to make use of the available Gehlhaar data from the non-gradient heuristics. To evaluate the influence of local search and population size on the performance of the MDLGA, we tested four different versions of the multi-deme Lamarckian genetic algorithms with gradient based optimization: sPopOne and bPopOne optimize one individual per iteration and population while sPopAll and bPopAll optimize all individuals. Furthermore, sPopOne and sPopAll have a smaller population size (10 individuals), while bPopOne and bPopAll have a larger population size (20 individuals). Each of the four algorithms possesses five interconnected populations and for a single local optimization, the gradient based optimization method is confined to at most 30

Metaheuristic abbreviation	Type of local search	Gradient information
GA, MDGA, PSO, DE	—	implicit
LGA, MDLGA	stochastic	implicit
MDLGAGR	gradient based	implicit + explicit

Table 9.1.: Overview of all tested search heuristics.

evaluations of the objective function.

9.1. Results

We summarized the results for the gradient based search heuristics together with the Gehlhaar results for the non-gradient heuristics from chapter 7 in Table 9.2, subdivided for ligands of low (0-3 flexible torsional angles), medium (4-7 flexible torsional angles) and high (more than 7 torsional angles) complexity. Regardless of the ligand complexity, the gradient based methods have a higher chance to find the global optimum. This difference is even more pronounced for ligands of high complexity. Additionally, the gradient based heuristics also require fewer function evaluations, albeit this difference decreases as ligands get more complex. The highly similar average best scores for ligands of low complexity indicate, that all methods were able to find the global optimum at least once during the 300 docking experiments. More or less, this finding seems to hold for ligands of medium complexity, with the exception of PSO, which, in agreement with our previous results, falls back in this regard. For ligands of high complexity, the gradient based methods, with the exception of bPopOne (indicated by a smaller average best score), deliver consistent results while the performance of the non-gradient search methods deteriorate severely. The declining performance of non-gradient methods compared to their gradient based competitors is also supported by the ratio of the average mean score that drops from 91.7% for simple ligands to 81.8% for complex ligands. Saturation results alone, however, do not allow an unambiguous judgment as the best non-gradient method, DE, stays in a touching distance to the best gradient based approach.

With regard to a comparison within the gradient based methods, those optimizing all individuals have a higher chance to find the global optimum but also require more function evaluations. For the methods, that do optimize just one individual per population and iteration, the one featuring a smaller population seems to produce slightly better results, but this finding is reversed for methods that optimize all individuals.

Fig. 9.1 displays the chance to find the global optimum, normalized by the number of function evaluations for ligands of increasing complexity. Besides the four gradient based methods, we included the best and the worst values from our previous study. Obviously, the gradient based methods are always well above their competitors, and again, the difference gets more marked as the complexity of the ligands increases. Additionally, gradient based heuristics, that optimize all individuals, fare slightly better than those that optimize just one individual per population and iteration. In general, however, the impact of different optimization parameters, at least in this

	name	∅ hits (%)	∅ mean score	∅ best score	∅ function eval.	∅ saturation
0-3 rot. bonds	MDLGA	46.62	-89.11	**-99.83**	14360	0.89
	PSO	52.58	-88.98	-99.56	10955	0.88
	DE	38.55	-85.80	-99.81	12692	0.93
	sPopOne	60.04	-92.64	-99.65	**5657**	0.93
	bPopOne	63.89	-92.71	-99.73	6178	0.89
	sPopAll	78.68	-96.59	-99.63	6720	**0.94**
	bPopAll	**84.77**	**-97.02**	-99.58	7369	**0.94**
4-7 rot. bonds	MDLGA	15.44	-98.36	-122.33	20275	0.55
	PSO	18.67	-95.68	-118.89	16473	0.54
	DE	10.24	-96.45	-123.05	19355	**0.73**
	sPopOne	26.71	-105.17	**-123.59**	**9620**	0.70
	bPopOne	25.06	-101.88	-123.06	10105	0.63
	sPopAll	39.65	**-110.19**	-123.51	11698	0.72
	bPopAll	**47.15**	-109.49	-123.29	12497	0.71
8-11 rot. bonds	MDLGA	3.83	-106.85	-147.72	26048	0.42
	PSO	2.75	-100.15	-142.30	21260	0.46
	DE	1.04	-103.57	-148.67	24837	0.49
	sPopOne	8.16	-123.81	**-156.40**	13533	0.44
	bPopOne	5.12	-112.92	-153.58	**13498**	0.43
	sPopAll	13.54	-130.07	-156.05	15728	0.43
	bPopAll	**21.62**	**-130.57**	**-156.40**	18354	**0.52**

Table 9.2.: Comparison four LGAs with gradient based local search and three population based meta-heuristics in terms of hit probability, running time, average mean and best energy as well as saturation. The results are partitioned for small, medium and large number of flexible torsional angles.

respect, is rather subtle.

9.2. Discussion

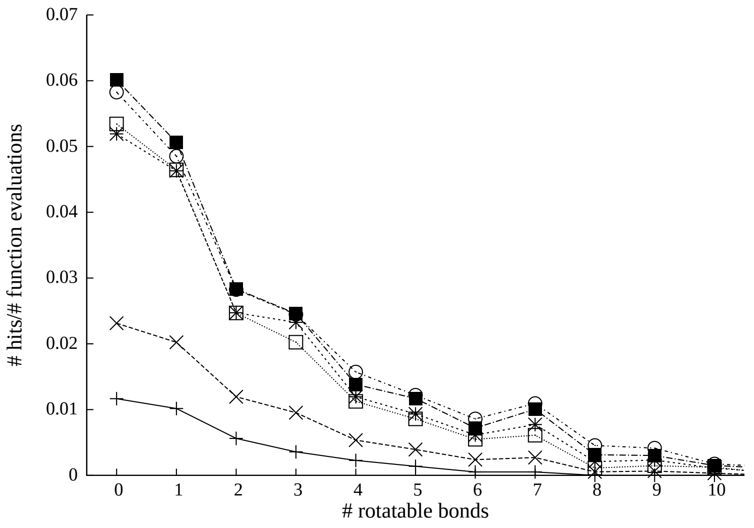
Our results clearly indicate, that population based search heuristics benefit strongly from incorporating a gradient based search method. Regardless of the complexity of the ligand, the gradient based methods deliver better results with fewer iterations. For ligands of high complexity, the performance of non-gradient procedures breaks down almost completely while gradient based methods are still feasible. One simple stochastic calculation reveals the dramatic improvement: if we want to archive a 99% chance to find the global optimum for a ligand of high complexity, we have to perform 118 docking experiments with MDLGA but only 19 with bPopAll. If we take into account, that bPopAll also requires fewer function evaluations, we gained an almost tenfold speedup.

We also tried to answer the question how many individuals should be locally optimized. Both heuristics that improved all individuals performed slightly better than those who improved just one individual per population and iteration. This finding is supported by the fact, that the heuristic that improved all individuals and featured a larger population performed better than the one with a smaller population, especially for ligands of high complexity. We contribute this behavior to a slowed convergence. In contrast, this result was reversed for both methods that just improved one individual. For these methods, the one with the smaller population dedicates a larger deal of its function evaluations to the gradient based local search procedure. All things considered, we conclude, that the gradient based local optimization method profits more efficiently from information provided by function evaluations, compared to non-gradient search heuristics.

9.3. Conclusion

By replacing the local search method of Solis& Wets by a gradient based minimizer evolved a search method that is superior to its non-gradient relative as well as to all population based meta-heuristics we have tested. Furthermore, our results suggest, that as many as possible function evaluations should be performed by the gradient based optimization method, while the actual heuristic is only responsible to deliver new starting positions for local optimizations. Although we cannot make a general statement concerning the impact of gradient based minimization in a heuristic global optimization of a real valued objective function, at least the energy function in ligand-receptor docking seems to be highly suitable for such an algorithm.

(a)



(b)

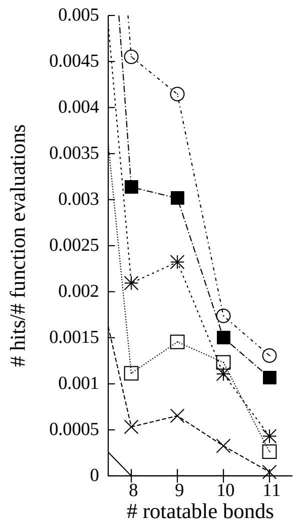


Figure 9.1.: Comparison four LGAs with gradient based local search in respect of hit probability normalized by the mean number of scoring function evaluations: sPopOne $*$, bPopOne $\square$, sPopAll ■, and bPopOne $\circ$. We also included the best ($+$) and the worst ($\times$) results of the gradient-free, population based meta-heuristics.

10. RMSD Score

Some methods, e.g. a Poisson-Boltzmann solver, allow for a more accurate approximation of the binding free energy. Since those computations are very time consuming, a utilization in a scoring function for ligand receptor docking is not feasible. Hence, standard docking procedures are used to obtain binding poses for re-scoring. Especially for ligands with few flexible torsional angles, modern docking programs tend to find the same binding pose in every single docking run. Evolutionary algorithms allow to produce results of higher diversity by choosing a smaller population size and, hence, achieve a faster convergence rate , which in turn increases the chance to get stuck in a local minimum. However, it is desirable to get the best binding poses of a ligand in descending order. Thus, we propose a method to enforce the docking program to produce more diverse results by incorporating a score derived from RMSD of the ligand's current positions to all final binding poses already found.

10.1. Methodology

There are multiple possibilities to derive a function for a given RMSD of *r* to a reference binding pose. Since we use the Gehlhaar scoring function for docking, which does not produce high values for clashed of atoms, we chose the following formula:

$$f(r) = \begin{cases} 20 - 10 \cdot r & \text{if } r < 2 \\ 0 & \text{otherwise} \end{cases} .$$

Hence, if a ligand superimposes a position already visited, the score is at most 20 (corresponding to a single van der Waals clash in the Gehlhaar scoring function) and decreases linearly with increasing RMSD. Since we want to include this score in the gradient based optimization process, we require the space derivatives of the individual atomic contributions. The length of the resulting vector v_i for an atom i with a distance d_i to its reference atom is given by

$$|v_i| = -\frac{10 \cdot d_i}{r \cdot n}$$

while the direction points from the reference atom to atom i.

In doing so, we alter the original scoring function in a way that we fill up sinks in the energy hyper-surface. As a result, those regions are less likely to be visited again by the search method, though not completely impossible.

Depending on the number of individual docking experiments, the number of binding poses that have to be considered for the computation of the RMSD score can lead to a substantial increase of the running time. Hence, it is desirable to only include those binding poses that may lead to a change in the RMSD score. Therefore, we use a hash-grid, provided by BALL, to store the respective binding poses. A hash-grid is a three-dimensional data structure, that allows to store and access objects in containers, indexed by their position in space. The space, covered by the hash-grid, is partitioned in cuboids of equal dimension and each one is associated with a container that stores all of its objects.

We define the edge length of a cuboid to be the break-off distance of our RMSD-scoring function. Then, we store each binding pose in the container, defined by the geometric center of the ligand. The distance of the geometric centers of two binding poses defines a lower bound for the RMSD.

If we have two different positions of the same molecule, defined by the atom's positions $\mathbf{X} = (x_1, x_2, ..., x_n)$ and $\mathbf{Y} = (y_1, y_2, ..., y_n)$, then the RMSD is given by

$$\text{RMSD} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2},$$

while the geometric centers $\bar{x}$ and $\bar{y}$ are given by

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

and

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i.$$

If we take $\bar{y} = 0$ without loss of generality, then the RMSD is given by

$$\begin{aligned}\text{RMSD} &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[(x_i - \bar{x}) - y_i + \bar{x}\right]^2} \\ &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[(x_i - \bar{x} - y_i) + \bar{x}\right]^2}\end{aligned}$$

Substituting $x_i - \bar{x}$ by

$$\tilde{x}_i = x_i - \bar{x}$$

yields

$$\begin{aligned}\text{RMSD} &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[(\tilde{x}_i - y_i) + \bar{x}\right]^2} \\ &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left[(\tilde{x}_i - y_i)^2 + \bar{x}^2 + 2\bar{x}(\tilde{x}_i - y_i)\right]} \\ &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\tilde{x}_i - y_i)^2 + \bar{x}^2 + 2\bar{x}\left(\frac{1}{n}\sum_{i=1}^{n}\tilde{x}_i\right) - 2\bar{x}\left(\frac{1}{n}\sum_{i=1}^{n}y_i\right)}\end{aligned}$$

Since

$$\frac{1}{n}\sum_{i=1}^{n}(\tilde{x}_i - y_i)^2$$

is surely non-negative,

$$\frac{1}{n}\sum_{i=1}^{n}y_i$$

is defined to be nil, and $\tilde{x}_i$ are the transposed atomic positions of **X**, such that its geometric center coincides with the origin, and thus

$$\frac{1}{n}\sum_{i=1}^{n}\tilde{x}_i = 0,$$

we can give a lower bound for the RMSD by

$$\text{RMSD} \geq |\bar{x}|.$$

(Proof by Dr. A. Hildebrandt)

Thus, the computation of the RMSD score can be limited to all binding poses that belong to the cuboid defined by the ligands geometric center and all its 26 neighbors, which leads to a significant reduction in running time. Fig. 10.1 displays the results with and without the RMSD score.

10.2. Application

One ligand that is notorious for being difficult to dock[16] is methylparaben. Although being relatively small with few degrees of freedom, neither GOLD nor the standard BALLDock is capable to find the native binding pose. One possible reason is that the binding pocket is very shallow. However, we observed, that any ligand position closer than 2 Å has a Gehlhaar score of at best -37, while the best scored result (RMSD $>$ 8 Å) has a score of -46.

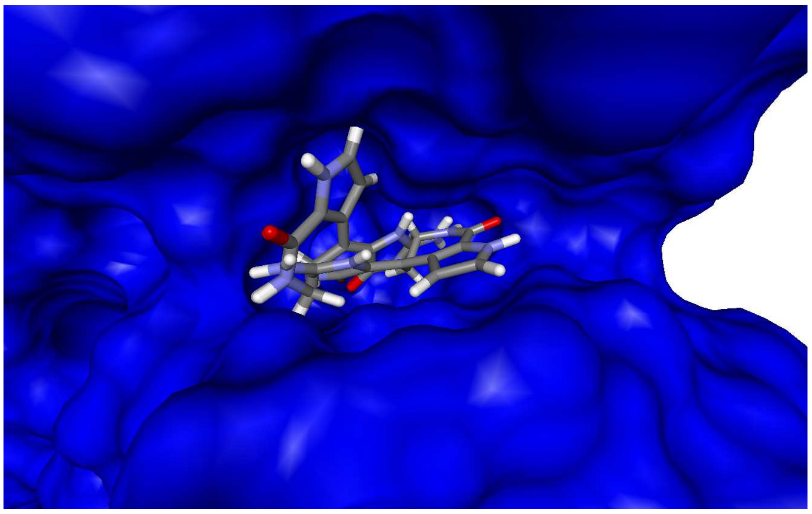
Table 10.1 lists the results of one docking experiment (25 individual docking runs) with and without RMSD score. BALLDock without RMSD score delivers consistent results that cluster around -43 and -46 with one single outlier at -37 (Fig. 10.2). Obviously, it is highly unlikely for BALLDock to find the native bind pose with its score of -37. In contrast, the results of BALLDock with RMSD score slowly decrease and are much more diverse (Fig. 10.3).

Then, we performed ten docking experiments à 25 docking runs. Without RMSD score, the best result had a RMSD of 4.5 Å to the native binding pose, while the average best and average mean RMSD were 6.1 Å and 8.0 Å, respectively. In contrast, with RMSD score, BALLDock was able to hit the native binding pose in every single docking experiment with an overall best result of 1.0 Å, average best RMSD of 1.1 Å, and average mean RMSD of 5.7 Å. Fig. 10.4 displays the binding poses with the smallest RMSD to the native binding pose for each docking experiment.

10.3. Conclusion

Our approach to add a score, derived from the RMSD to already found positions, to the approximated binding free energy leads to a significantly increased diversity of results for ligands with no or few flexible torsional angles. Our results show, that BALLDock employing RMSD score is able to reconstruct the native binding pose in cases where other docking programs fail.

(a)



(b)

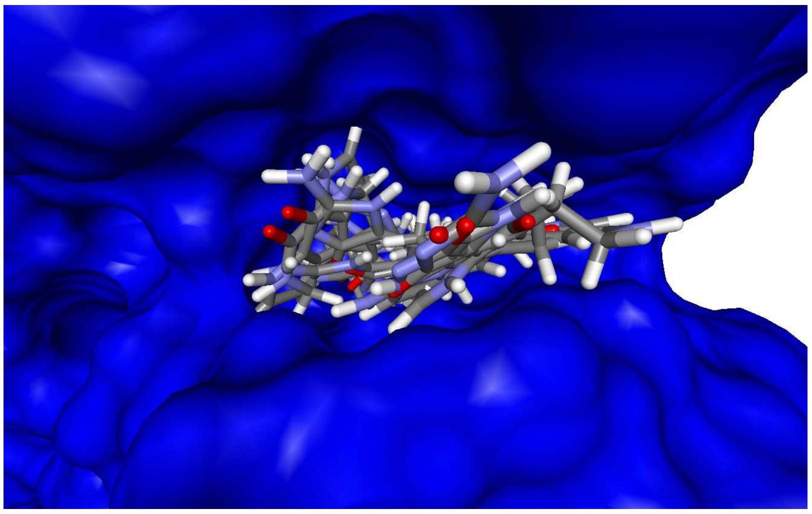


Figure 10.1.: (a) Results of ten docking runs of 1U4D without RMSD score. Nine of them have scores in the range of -73.2 to -73.4 and are almost identically positioned. The only outlier has a score of -70.3. (b) In contrast, the results using RMSD score are much more diverse and range from -62.2 to -73.4.

Gehlhaar score	
with RMSD	without RMSD
-43.21	-43.50
-42.72	-43.56
-40.17	-43.89
-46.78	-42.73
-40.82	-43.74
-37.72	-43.49
-37.49	-46.70
-37.33	-37.49
-36.74	-43.69
-37.15	-43.71
-36.13	-43.38
-36.11	-43.59
-39.42	-46.72
-36.12	-46.76
-36.12	-46.72
-36.02	-43.53
-35.54	-46.74
-35.21	-46.75
-36.71	-46.76
-35.02	-43.73
-35.31	-43.43
-34.91	-43.66
-34.92	-43.50
-34.78	-43.74
-35.20	-46.81

Table 10.1.: Gehlhaar scores of 25 docking runs with and without RMSD score (3MTH).

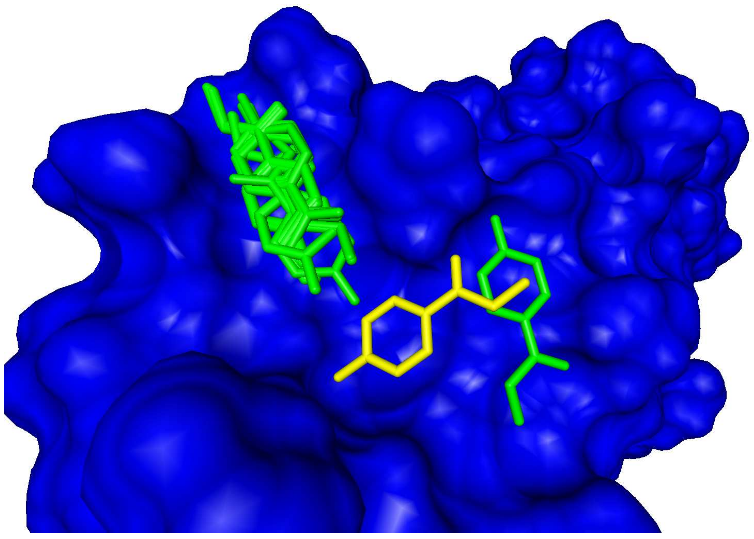


Figure 10.2.: Results of a docking run without RMSD score and the native binding pose (yellow).

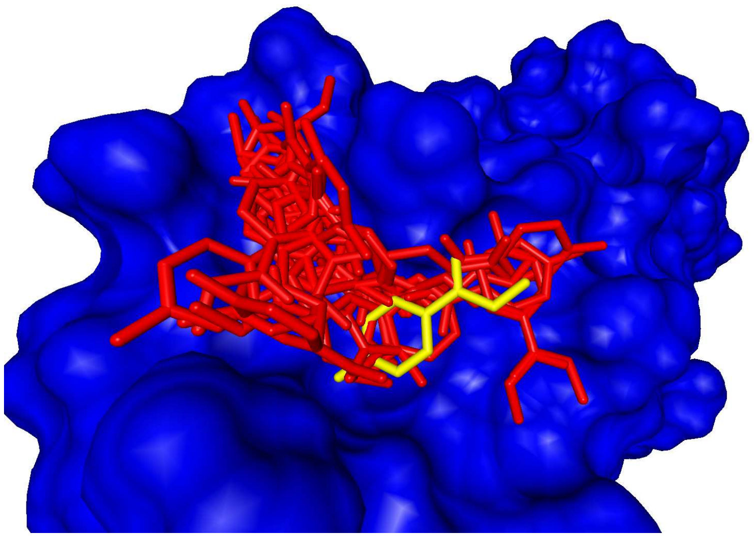


Figure 10.3.: Results of a docking run with RMSD score and the native binding pose (yellow).

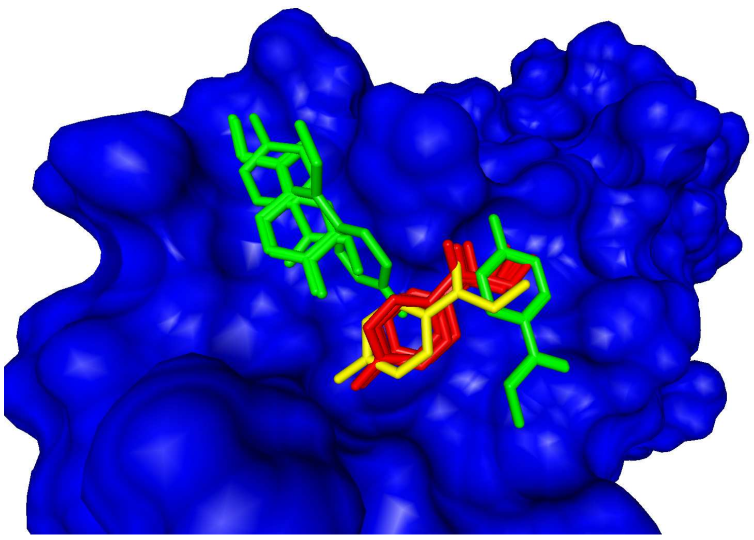


Figure 10.4.: Results with the lowest RMSD to the native binding pose (yellow) using BALL-Dock with (red) and without (green) RMSD score.

11. Problem optimizer interface

The increasing number of available optimization methods (genetic algorithm, differential evolution, particle swarm, Solis & Wets) and optimization tasks (ligand-receptor docking, protein-protein docking, structure optimization, loop prediction, etc.) demands for a common interface in BALL. Therefore, we defined an ensemble of classes, that facilitate the application of different optimization methods to an arbitrary optimization problem.

11.1. Parameter class

Every kind of optimization problem must possess a set of parameters, that represent the search space. To allow for all kinds of parameters, the generic parameter class features only a minimum set of member variables and methods, i.e. a string to define the parameter's name and a static random number generator to allow for a parameter randomization.

Up to now, we implemented a real valued parameter of arbitrary dimensionality and a unit quaternion parameter. Further possibilities are bit-strings or more advanced structures like for example graphs.

RealParameter The class `RealParameter` implements a set of independent, real valued parameter values. Thus, it is derived from `vector<float>` and `GenericParameter` (Fig.

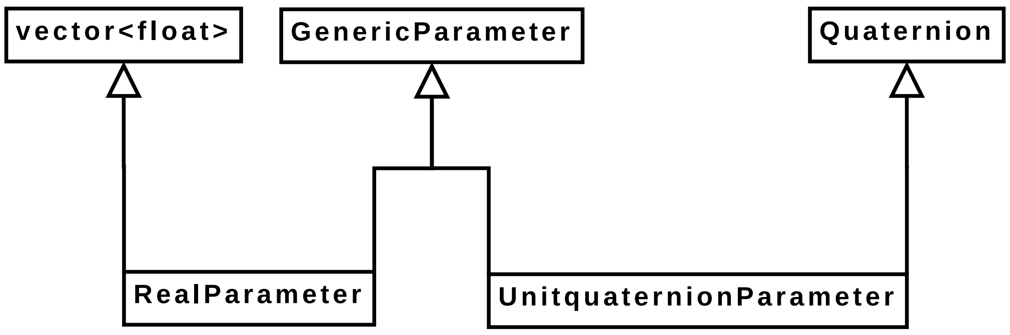


Figure 11.1.: Hierarchy of parameter classes.

11.1). Additionally, it possess a `vector<float>` of the same dimensionality as the number of parameters, named `scaling_`. This vector defines for each parameter the granularity, which is required by some optimization methods, e.g. simulated annealing. Other methods, e.g. our genetic algorithm, demand for upper and lower bounds, which are defined in the vectors `upper_bound_` and `lower_bound_`, again separately for each parameter. Finally, the behavior of a parameter, when the respective bound is violated, has to be defined. Generally, the bounding conditions can be restored by setting the value to the lower bound, if the lower bound has been violated or to the upper bound, in the opposite case. This can be advisable, if the parameter defines e.g. a translation. However, this strategy involves a severe drawback if the parameter defines e.g. a flexible torsional angle. In this case, the artificially introduced barrier at 360° or $+/-180^\circ$ limits the search of molecular conformations. To deal with this problem, `RealParameter` defines a `vector<bool>` that contains a flag for each parameter. If set true, the parameter should be treated like a ring, e.g. if the parameters x violates the upper bound, the x should be replaced by

$$lower\_bound\_ + mod(x - lower\_bound\_, upper\_bound\_ - lower\_bound\_),$$

while the violation of the lower bound should be handled by replacing x by

$$lower\_bound\_ + upper\_bound\_ + mod(x - lower\_bound\_, upper\_bound\_ - lower\_bound\_).$$

If the flag is set to false, the respective parameter is set to the value of the violated bound.

UnitquaternionParameter Representing the orientation of a body in Cartesian space by a unit quaternion is a common technique in molecular modeling. Defining the values of a quaternion q with a standard `RealParameter` is ill fated, since the independent optimization of its four values surely violates the constraint $|q| = 1$. Therefore, we implemented a dedicated unit quaternion parameter, derived from `GenericParameter` and the quaternion class of the BALL library[23] (Fig. 11.1). The optimization method bears the responsibility to conserve the norm 1 constraint, thus guaranteeing that `UnitquaternionParameter` defines a valid orientation.

Since `UnitquaternionParameter` does not require options, e.g. upper and lower bound, the class interface is much more compact, compared to `RealParameter`.

11.2. Problem class

The base class `GenericProblem` defines an interface for an optimization problem. `Generic Problem` possesses the method `connectTo`, that allows to bind a problem class to an optimizer class, using the `registerParameter` method of `Optimizer`. Any specific optimization task derived from `GenericProblem` is expected to meet those requirements: first, it must provide a set of parameters in `parameters_`, and secondly, it must overload the `calculate` method, that is purely virtual in `GenericProblem`. The return value of this function enables a search method to compare the quality of different parameter values.

The canonical way for the application of a local search method is to adopt the problem-optimizer interface. Thus, the algorithm of Solis & Wets was implemented this way, performing its optimization task upon the parameters of `DockProblem`. However, this is not the best choice in every case. In exponential mapping, the gradients are of decreasing quality the more the unit quaternion diverges from the quaternion representing the neutral orientation (0,0,0,1). Since the global optimization method rapidly cuts down on parameter value diversity, most local optimizations won't lead to a radical change of orientation. Therefore, we change the orientation of the ligand before each single gradient based local optimization to ensure that the initial orientation corresponds to the quaternion with the best gradients. Since this approach is rather problem specific, `GenericProblem` possesses a method called `localImprove` that allows to implement a custom-made local search.

Finally, `GenericProblem` provides a method `finalize`, that it called by an optimization method after the actual optimization. It can be used to provide the final results, in the case of docking, the best ligand position is stored in a `HIN`-file.

11.3. Optimizer class

To enable a problem object to connect itself to an optimizer object, `Optimizer` provides the method `registerParameter`. The optimizer has to check, if it is able to optimize the parameter provided by the problem class. If not, there are two possibilities: first it may just abort and report an error, or it only works on the other parameters. A practical example is the application of the local search method of Solis & Wets that is constrained to real valued parameters. A genetic algorithm, on the other hand, can also handle integer values. If e.g. loop conformations are parametrized by integer variables in a genetic algorithm and Solis and Wets is applied to improve existing individuals, the local search can be focused on orientation, translation, and

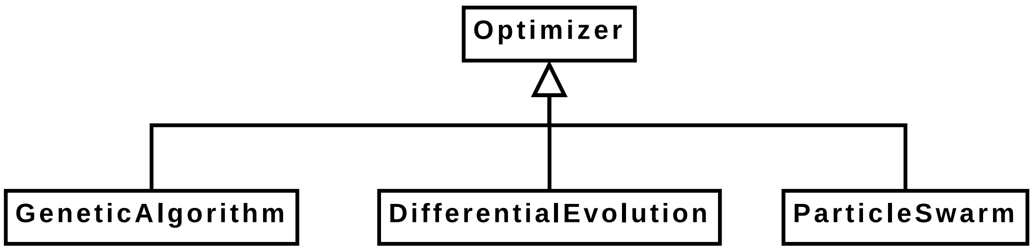


Figure 11.2.: Hierarchy of the optimization classes.

torsional flexible angles disregarding changes in loop conformations.

For the actual optimization, initialized by the method `start`, the optimizer applies its strategy to search the parameter space, guided by the score provided by the `calculate` method of the problem class. After the optimization is finished, the optimizer ought to call the `finalize` method of the problem, to specify the results.

For the comparison of population based meta-heuristics, we implemented three different evolutionary algorithms, derived from the generic `Optimizer` class (Fig. 11.2).

11.4. Integration of docking

The class `DockProblem`, derived from `GenericProblem` (Fig. 11.3), implements a model for ligand-receptor docking. It demands one `UnitquaternionParameter` for the orientation of the ligand and one `RealParameter` for translation and flexible torsional angles. Invoking the `assignScore` method moves the ligand according to the parameters and returns the Gehlhaar score of the complex, which is provided by the classes `GehlhaarFF` and `EnergyGrid`. The latter uses a precomputed map of the interaction energy, which is calculated by the class `EnergyGridBuilder` by placing a probe atom on equidistant nodes of a regular three-dimensional grid. The room occupied by this grid should contain the space of reasonable ligand positions. Since there is no representation for internal coordinates in BALL, we implemented `RotateBond`, that allows the user to define a rotation around a flexible torsional angle, and `RotateBonds`, that contains all `RotateBond` objects of a molecule and thereby represents its conformational state.

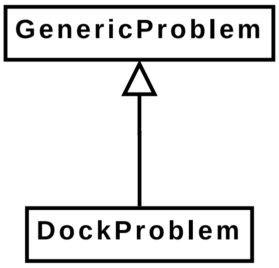


Figure 11.3.: Hierarchy of the problem classes.

11.5. Conclusion

The existing approaches for optimization in BALL are deeply interweaved with molecular force fields, e.g. `AmberFF`. The problem-optimizer interface is a first attempt to separate optimization methods from optimization problems, and thereby permitting an interchangeability of different optimizers and problems. As a proof of concept, we used this framework for the implementation of three population based optimization algorithms, with some simulated annealing methods being under development. At this moment, `DockProblem`, that defines a model for ligand-receptor docking, is the only representation with another one for protein-protein docking being also under development. By adding those newly implemented classes to the BALL library, we allowed for a rapid prototyping of ligand-receptor docking programs.

12. BALLDock Docking Suite

By the realization of `DockProblem` and different optimization classes within BALL, it is possible to build a custom-made docking program in reasonable time. However, since many potential users, e.g. pharmacological researchers, are not familiar with a complex programming language like C++, we developed a ready-to-operate docking program, called BALLDock, together with three auxiliary programs for ligand pre-processing, detection of flexible torsional angles, and energy grid computation. Since `BALLDock` features batch processing, it can be used to dock a whole library of molecules with just one program call. The only task, BALLDock is not able to perform on its own is the addition of hydrogen atoms with a correct bond geometry. Hence, the user is required to add hydrogen atoms to the ligand and receptor.

12.1. ProcessLigand

BALLDock requires all atoms of the ligand to have a unique name. `ProcessLigand` reads in a ligand `HIN`-file or a text file that contains the name of one or multiple ligand `HIN`-files without the `.hin` suffix. In the next step, the uniqueness of all atom names is checked, and if the check fails, all atoms are renamed after their element name together with the rank of the atom in the ligand file. Finally, the molecule is stored in a file, with the original suffix `.hin` being replaced by `-p.hin`.

A call to `ProcessLigand` with a single `HIN`-file may look like:

```
> ./ProcessLigand ligand.hin
```

`ProcessLigand` checks the filename of the first argument and if it is not a `HIN`-file, indicated by the absence of the suffix `.hin`, it takes the argument as a file, that contains the name of all ligands:

```
> ./ProcessLigand ligands.txt
```

12.2. FTAngles

ProcessLigand is followed by `FTAngles`, that tries to find all flexible torsional angles. Again, it is possible to process a single or multiple `HIN`-files. All bonds, that represent a flexible torsional angle, are stored in a file, named after the original file-name of the ligand, followed by the suffix `.rbs`. Thereby, a bond is defined by the names of its atoms. The usage of `FTAngles` is identical to that of `ProcessLigand`.

It must be noted, however, that the method, used by `FTAngles` is not entirely trustworthy. While it is generally reliable for easily decidable bonds, like those in ring structures or double bonds, it may fail for special single bonds, that are not flexible, e.g. bonds involved in mesomerism.

12.3. GridBuilder

The final step before the actual docking experiment is the pre-computation of the energy grid. `Gridbuilder` is started with a configuration file.

```
> ./GridBuilder gb.cfg
```

This configuration file contains all information required by `GridBuilder`:

```
# name of the receptor
receptor_file_name protein.pdb

# name of the grid
grid_name GRID

# edge length of the cube that contains the energy grid
grid_extension 20.
```

Thereby, `receptor_file_name` defines the name of the `PDB`-file, that contains the receptor protein. If `grid_name` is defined, the file that stores the grid data is named after `grid_name`, followed by the suffix `.grid`, otherwise the grid name is derived from the receptor file name. The extension of the grid is defined by a cube whose edge length is given by “grid_extension”. If the locality of the binding pocket is roughly known and the ligand is rather small, an edge length of 20 Å is usually sufficient while otherwise an edge length of 30 Å or more is advised.

12.4. BALLDock

After all torsional flexible angles have been determined and the energy grid has been computed, BALLDock is started for the actual docking process, again with a configuration file:

```
> ./BALLDock dock.cfg
```

Just like for `GridBuilder`, the configuration file defines the program parameters:

```
# number of docking experiments for each ligand
runs 100

# Available: differenial evolution (DE), particle swarm (PSO),
genetic algorithm (GA)
algorithm DE

# without suffix (e.g. GRID for GRID.gr)
grid_name GRID

# file that contains all ligands
ligand_file ligands.txt

# possible translation in each dimension
translation_box 10.

# do local optimization, if possible
local_search 1

# best values for convergence_iterations
# if no local search is used:
# between 1000 (few rotatable bonds) and
# 10000 (many rotatable bonds)
# if local search is used:
# between 100 (few rotatable bonds) and
# 1000 (many rotatable bonds)
convergence_iterations 10000
```

```
convergence_value 1.
max_iterations 50000

# parameters for genetic algorithm
ga_population_size 40
ga_mutation 0.00
ga_immune 1
ga_survivors 20
ga_initial_population 20
ga_population_number 4

# parameters for differential evolution
de_population_size 50
de_mutation 0.7
de_randomize_factor 1
de_factor 2.

# parameters for particle swarm
ps_swarm_size 50
ps_cognitive 2.
ps_social 2.
ps_inertia .7
```

`runs` defines the number of docking experiments performed by BALLDock for each ligand while `algorithm` determines the optimization method. To-date there are three different algorithms available, differential evolution, particle swarm optimization and genetic algorithm with different versions of simulated annealing being under development. Parameters for an individual method are denoted by a prefix (`ga`, `de` and `ps`). The receptor is provided by the pre-computed grid file whose name is given by `grid_name` while the filenames of all ligands are contained in the text file indicated by `ligand_file`. The flexible torsional angles of a ligand are defined by the respective file, produced by `FTAngles`. If this file is empty or absent, `BALLDock` performs a rigid docking, which is valuable, e.g. for docking multiple pre-computed conformations of one molecule. Finally, the translation of the ligand is defined by a cube whose edge length is defined by `translation_box`. All optimization methods feature the same stopping criterion. If, for a number of function evaluations, the best found score doesn't improve by a given threshold value, the algorithm stops. Both parameters are defined by `convergence_iterations` and `convergence_value`, respectively. Since the number of function evaluations performed by the local search procedure is not considered for the stopping criterion, `convergence_iterations` ought to be reduced, if local search is activated. Additionally, it is possible to constrain the running time by defining a maximum number of function evaluations by `max_iterations`.

BALLDock starts with loading the grid data and informs the user about the progress. Subsequently, each ligand is loaded and the defined number of docking runs is performed. Again, the progress is displayed and, after the final ligand was processed, BALLDock builds a table with the best and average scores.

```
--------------------------------------------
| ligand  |  best score  |  average score |
--------------------------------------------
| 1opk    |     -139.5   |       -137.9   |
| 1oq5    |      -94.0   |       - 89.5   |
| 1owe    |     -118.2   |       -115.5   |
| 1oyt    |     -139.8   |       -130.8   |
| 1p2y    |      -63.4   |        -62.6   |
| 1p62    |      -89.1   |        -88.5   |
| 1pmn    |     -126.1   |       -123.2   |
| 1q1g    |     -121.3   |       -120.5   |
| 1q41    |      -84.3   |        -87.1   |
| 1r1h    |     -154.1   |       -125.8   |
| 1r55    |     -120.2   |       -106.8   |
| 1r58    |     -120.5   |       -105.5   |
| 1r9o    |      -81.9   |        -79.7   |
| 1s19    |     -133.7   |       -126.4   |
| 1s3v    |     -135.3   |       -111.9   |
| 1sg0    |     -120.0   |       -118.1   |
| 1sj0    |     -141.6   |       -138.0   |
| 1sq5    |      -76.8   |        -67.3   |
| 1sqn    |      -93.2   |        -92.8   |
--------------------------------------------
```

A small gap between best and average score indicates, that the search method was able to cope with the complexity of the search space. Results for ligands with a larger gap should be examined more carefully, for it is possible, that e.g. a larger population size and thus a delayed convergence might entail higher consistency.

For each ligand, BALLDock stores the best found ligand position of a single docking experiment in a numbered `HIN`-file and writes a log-file with all scores.

```
----------------------------
| 1t40-1.hin    |   -152.3  |
| 1t40-2.hin    |   -136.5  |
| 1t40-3.hin    |   -133.5  |
| 1t40-4.hin    |   -125.4  |
| 1t40-5.hin    |   -113.1  |
| 1t40-6.hin    |   -137.9  |
| 1t40-7.hin    |   -126.4  |
| 1t40-8.hin    |   -110.4  |
| 1t40-9.hin    |   -131.4  |
| 1t40-10.hin   |   -154.3  |
| 1t40-11.hin   |   -132.1  |
| 1t40-12.hin   |   -107.7  |
| 1t40-13.hin   |   -141.4  |
| 1t40-14.hin   |   -158.7  |
| 1t40-15.hin   |   -160.7  |
| 1t40-16.hin   |   -113.2  |
| 1t40-17.hin   |   -135.0  |
| 1t40-18.hin   |   -128.3  |
| 1t40-19.hin   |   -130.2  |
| 1t40-20.hin   |   -106.0  |
----------------------------
```

12.5. Conclusion

Despite the absence of a graphical user interface, BALLDock is an easy-to-use tool to reconstruct the binding mode of a ligand. By utilizing the Gehlhaar scoring functions, BALLDock saves the user from difficult optimization of hydrogen atoms and time consuming calculations of appropriate point charges. The high chance to find the global optimum of the scoring functions together with the low number of function evaluations leads to a significant reduced running time. By using batch processing, testing of a whole library of chemical compounds can be accomplished with just a few working steps. On the down side must be noted, that BALLDock lacks a rotamer library to allow for different ligand conformations of e.g. ring structures as well as a reliable determination of flexible torsional angles. Finally, a further reduction of the running time can be accomplished by an optimization of critical code sections, e.g. cache optimization or utilization of SSE.

13. Prospected features

Not all projected features are completed at this point. Perhaps the most prominent challenge in ligand-receptor docking today is the inclusion of conformational alterations in the receptor. To deal with this problem, we are involved in the implementation of two different approaches. First, we try to describe the movement of the receptor with just one variable by extrapolating residue positions from two extreme positions, as displayed in Fig. 13.1. This is possible, if the backbone performs a limited and isolated movement relative to a rigid domain of the receptor, which is the case, e.g. in human serum albumin. This approach can be extended, by just moving the backbone atoms while the side-chain can be treated as flexible like the ligand.

Furthermore, we use the method of Go and Scheraga[73] to allow for loop movements. Often, e.g. in 17-beta-HSD1, residues belonging to beta-sheets or alpha-helices are relatively rigid, while loop regions of the backbone are highly movable. The mathematical difficulty lies in the fact, that both ends of the flexible loops have to be stationary. If we treat one end to be fixed, the change of a torsional flexible angle illicitly moves the second end. However, it is possible to calculate values for dependent torsional angles to restore the invariant. Fig.13.2 illustrates four different backbone conformations that establish a closed backbone conformation.

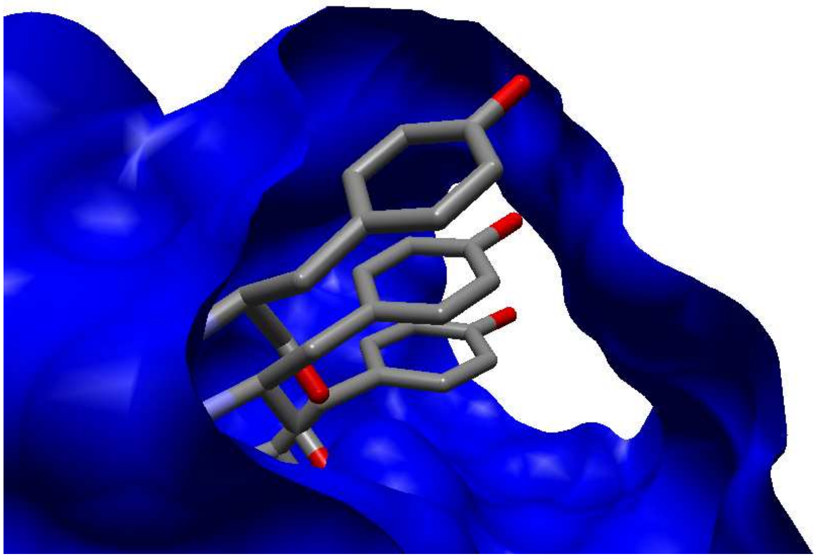


Figure 13.1.: Movement of tyrosine 150 in the warfarin binding pocket of human serum albumin. The three different conformations represent the two extreme and one intermediate positions.

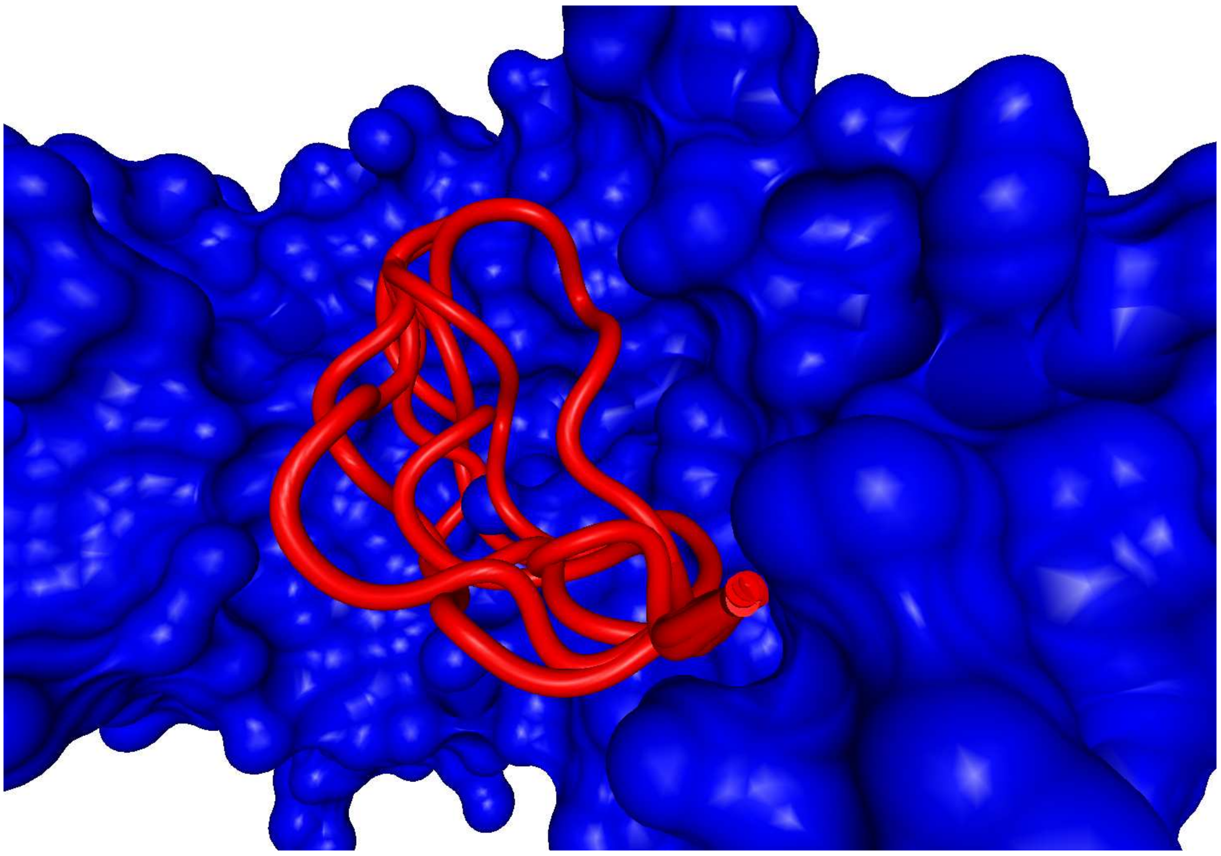


Figure 13.2.: Different possibilities to bridge a gap in the backbone. For two given stationary points, the algorithm of Go and Scheraga calculates values for a set of dependent torsional angles.

14. Conclusion

Despite the existence of many programs for ligand-receptor docking, the problem is considered to be unsolved. There is still a long way to go to provide pharmacological researchers with what they wish for: a tool that accurately predicts the binding free energy for an arbitrary ligand and receptor. One of the limiting factors to-date is the reconstruction of the native binding pose. Even if there is no method to calculate the binding free energy, a deeper understanding of the interaction between functional groups of the ligand and receptor is hugely useful for lead optimization.

In fact, most molecules that fall within Lipinski's rule of five[74] or some other measure for drug-likeliness can be docked by different kinds of docking programs. The predicted binding mode does not necessarily coincide with the native binding pose, but in most cases, a deviation can be attributed to a failing scoring function, rather than a failing optimization method.

Improving the speed at which an optimization methods finds the global optimum of a scoring function is desirable. But does a limited speedup, that is also achievable by the application of better hardware, justify the high scientific effort evidenced by numerous publications? The answer would most probably be "no" if this was the whole story, but looking closely at the limiting factors of ligand-receptor docking tells a different truth. On the part of the energy calculation, the influence of water as a polar solvent to the binding free energy can hardly be estimated without time consuming molecular-dynamics simulations. Furthermore, when it comes to predicting the native binding pose, the assumption, that the receptor conformation remains unaffected by the influence of the ligand is inadmissible in many cases. If the underlying model does not allow for induced fit alterations of the receptor, we can't expect any scoring function to have its global optimum nearby the native binding pose. This being the case, why not just add receptor flexibility to our model? The answer is, that currently available search methods give up in the face of the complexity of the resulting search space.

We do not claim, that the method presented in this work solves the ligand-receptor docking problem. Nevertheless, our results suggest, that BALLDock pushes the limits set by the inability of existing search algorithms to cope with search spaces of higher complexity. In a following step, beyond the focus of this work, we want to analyze, if we can reconstruct the native binding poses of ligand-receptor complexes, that require more or less extensive alterations in the receptor conformations.

15. Summary

15.1. Evolutionary algorithms applied to ligand-receptor docking

In contrast to other publications,[11, 12, 39, 40] we could not find a substantial advantage in performance for a specific global search heuristic. For search spaces of high complexity, all evolutionary algorithms seem to be hampered by a lack of information, given by the one-dimensional score of the objective function.

15.2. Comparison of a simple to a more complex scoring function in ligand-receptor docking

We compared the simple Gehlhaar scoring function to the AUTODOCK scoring function solely on the base of their ability to discriminate native binding poses from decoy positions. In this regard, the AUTODOCK approach to include complex contributions to the binding free energy, like entropic and hydrophobic interactions, does not lead to better results. Quite the contrary, the Gehlhaar function scored the native binding poses better than any decoy position for more ligands

15.3. Gradient based local search in ligand-receptor docking

We presented a novel way to apply gradient based local optimization in ligand-receptor docking. Thereby, singularities that arise from representing a molecule by translation and orientation, are avoided by using exponential mapping. A comparison to the gradient-free method of Solis & Wets proved the general superiority of our approach for local optimization. A Lamarckian genetic algorithm experienced a boost in performance in terms of running time and ability to find the global optimum in search spaces of high complexity when using gradient based local optimization.

15.4. RMSD score

We developed a method to enforce more diverse results in ligand-receptor docking by employing a score derived from RMSD of the ligand's current positions to already found binding poses. Thus, BALLDock has a higher chance to reconstruct native binding poses even if they do not coincide with global minima of the underlying scoring function.

15.5. BALLDock

Our new gradient based approach was incorporated as set of C++ classes in the BALL library. Finally, we implemented BALLDock as an easy-to-use command-line tool for ligand-receptor docking.

16. Zusammenfassung

16.1. Evolutionäre Algorithmen im Bereich Ligand-Rezeptor Docking

Unser Vergleich mehrerer evolutionärer Algorithmen lieferte keine Beweise für eine entscheidend höhere Leistungsfähigkeit einer Methode. Während dieses Ergbnis im Gegensatz zu anderen Veröffentlichungen[11, 12, 39, 40] steht, bestätigten unsere Untersuchungen, dass alle Algorithmen ab einer bestimmten Komplexität des Suchraums nicht mehr in der Lage sind, das globale Optimum zu finden. Dieses Verhalten führten wir auf einen mangelnden Informationsgehalt einer eindimensionalen Zielfunktion in hochdimensionalen Suchräumen zurück.

16.2. Vergleich einer einfachen mit einer komplexeren Zielfunktion für Ligand-Rezeptor Docking

Auch beim Vergleich zweier Zielfunktionen für Ligand-Rezeptor Docking waren die Unterschiede im Hinblick auf die Fähigkeit, korrekte von falschen Bindepositionen zu unterscheiden marginal. Überraschenderweise lieferte die sehr einfach aufgebaute Gehlhaar Funktion etwas bessere Ergebnisse als die AUTODOCK-Funktion, obwohl letztere komplexe Terme für entropische und hydrophobe Wechselwirkungen beinhaltet.

16.3. Gradientbasierte lokale Suche für Ligand-Rezeptor Docking

Unter Verwendung von Exponential Mapping entwickelten wir einen neuen Ansatz zur gradientbasierten lokalen Optimierung im Bereich Ligand-Rezeptor Docking. Dabei legten wir großen Wert darauf, einen Verlust von Freiheitsgraden zu vermeiden. Im direkten Vergleich zur weit verbreiteten Methode von Solis & Wets, die ohne Gradientinformation auskommt, lieferte unser Ansatz entscheidend bessere Ergebnisse. Die Ersetzung der Methode von Solis & Wets in einem Lamarck genetischen Algorithmus durch unseren Ansatz führte ebenfalls zu deutlich verbesserten Ergebnissen im Hinblick auf die Laufzeit und die Fähigkeit, das globale Optimum in Suchräumen hoher Komplexität zu finden.

16.4. RMSD score

Um eine größere Vielfalt an Ergebnissen zu erreichen, entwickelten wir eine Methode um den Abstand eines Liganden zu bereits in vorherigen Docking-Läufen gefundenen Positionen zu bewerten. Dadurch steigt die Wahrscheinlichkeit, dass BALLDock die korrekte Bindeposition rekonstruiert, selbst wenn diese nicht mit dem globalen Optimum der Zielfunktion übereinstimmt.

16.5. BALLDock

Alle in dieser Arbeit programmierten Suchmethoden, Zielfunktionen und sonstigen Klassen wurden zur BALL Bibliothek hinzugefügt. Dadurch ist es in kurzer Zeit möglich, ein eigenes Dockingprogramm zusammenzustellen. Außerdem entwickelten wir mit BALLDock ein fertiges Kommandozeilenprogramm, welches mittels Optionsdateien auch für Laien leicht zu handhaben ist.

Bibliography

[1] E. Fischer. Einfluss der configuration auf die wirkung der enzyme. *Ber. Dt. Chem. Ges*, 27:2985–2993, 1894.

[2] D.E. Koshland. Application of a theory of enzyme specificity to protein synthesis. *Proc Natl Acad Sci U S A*, 44:98–104, 1958.

[3] M.J.D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Computer Journal*, 7:155–162, 1964.

[4] R. Abagyan, M. Totrov, and D. Kuznetsov. Icm - a new method for protein modeling and design: Applications to docking and structure prediction from the distorted native conformation. *J. Comput. Chem.*, 15:488–506, 1994.

[5] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques–Theory and Practice*. ACM Press, 1992.

[6] D.E. Clark and D.R. Westhead. Evolutionary algorithms in computer-aided molecular design. *J. Comput. Aided Mol. Des.*, 10:337–358, 1996.

[7] M. Vieth, J.D. Hirst, B.N. Dominy, H. Dailer, and C.L. Brooks III. Assessing search strategies for flexible docking. *J. Comput. Chem.*, 19:1623–1631, 1998.

[8] B.D. Bursulaya, M. Totrov, R. Abagyan, and C.L. Brooks III. Comparative study of several algorithms for flexible ligand docking. *J. Comput. Aided Mol. Des.*, 17:755–763, 2003.

[9] E. Kellenberger, J. Rodrigo, P. Muller, and D. Rognan. Comparative evaluation of eight docking tools for docking and virtual screening accuracy. *Proteins*, 57:225–242, 2004.

[10] J.C. Cole, C.W. Murray, J.W.M. Nissink, R.D. Taylor, and R. Taylor. Comparing protein-ligand docking programs is difficult. *Proteins*, 60:325–332, 2005.

[11] V. Namasivayam and R. Günther. PSO@AUTODOCK: A fast flexible molecular docking program based on swarm intelligence. *Chem. Biol. Drug. Des.*, 70:475–484, 2007.

[12] H.-M. Chen, B.-F. Liu, H.-L. Huang, S.-F. Hwang, and S.-Y. Ho. SODOCK: Swarm optimization for highly flexible protein-ligand docking. *J. Comput. Chem.*, 28:612–623, 2007.

[13] F.J. Solis and R.J.-B. Wets. Minimization by random search techniques. *Math. Op. Research*, 2:19–30, 1981.

[14] S.F. Sousa, P.A. Fernandes, and M.J. Ramos. Protein-ligand docking: Current status and future challenges. *Proteins*, 65:15–26, 2006.

[15] M. Rarey, S. Wefing, and T. Lengauer. Placement of medium-sized molecular fragments into active sites of proteins. *J. Comput. Aided Mol. Des.*, 10:41–54, 1996.

[16] G. Jones, P. Willet, R.C. Glen, and A.R. Leach. Development and validation of a genetic algorithm for flexible docking. *J. Mol. Biol.*, 267:727–748, 1997.

[17] G.M. Morris, D.S. Goodsell, R.S. Halliday, R. Huey, W.E. Hart, R.K. Belew, and A.J. Olson. Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function. *J. Comput. Chem.*, 19:1639–1662, 1998.

[18] M. Avriel. *Nonlinear Programming: Analysis and Methods*. Dover Pubn Inc, 2003.

[19] Z. Michalewicz and D.B. Fogel. *How to Solve it: Modern Heuristics*. Springer, 2004.

[20] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[21] T.J. Ypma. Historical development of the newtonraphson method. *SIAM Review*, 37:531–551, 1995.

[22] Jr. J.E. Dennis and R.B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations (Classics in Applied Mathematics, 16)*. Soc. for Industrial & Applied Math., 1996.

[23] O. Kohlbacher and H.-P. Lenhof. BALL - rapid software prototyping in computational molecular biology. *Bioinformatics*, 16:815–824, 2000.

[24] J.-Y. Trosset and H.A. Scheraga. PRODOCK: Software package for protein modeling and docking. *J. Comput. Chem.*, 20:412–427, 1999.

[25] M.J. Hartshorn, M.L. Verdonk, G. Chessari, S.C. Brewerton, W.T. M. Mooij, P.N. Mortenson, and C.W. Murray. Diverse, high-quality test set for the validation of protein-ligand docking performance. *J. Med. Chem.*, 50:726–741, 2007.

[26] J.H. Holland. *Adaption in natural and artificial systems*. MIT Press, Cambridge, MA, 1975.

[27] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, ICSI, 1995.

[28] R. Eberhart and J. Kennedy. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks (Perth Australia)*, pages 1942–1948, Piscataway, NJ (USA), 1995. IEEE Press.

[29] R.L. Haupt and S.E. Haupt. *Practical Genetic Algorithms*. John Wiley Sons, New York, USA, 1998.

[30] E. Cantú-Paz. *Efficient and accurate parallel genetic algorithms*. Kluwer Academic Publishers, Boston, MA, 2001.

[31] W.E. Hart. *Adaptive Global Optimization with Local Search*. PhD thesis, Computer Science and Engineering Department, University of California, San Diego, 19994.

[32] C.M. Oshiro, I.D. Kuntz, and J.S. Dixon. Flexible ligand docking using a genetic algorithm. *J. Comput. Aided Mol. Des.*, 9:113–130, 1995.

[33] R.S. Judson, E.P. Jaeger, and A.M. Treasurywala. A genetic algorithm based method for docking flexible molecules. *J. Mol. Struct. (Theochem)*, 308:191–206, 1994.

[34] P.S. Charifson, J.J. Corkery, M.A. Murcko, and W.P. Walters. Consensus scoring: a method for obtaining improved hit rates from docking databases of three-dimensional structures into proteins. *J. Med. Chem.*, 42:5100–5109, 1999.

[35] J.S. Taylor and R.M. Burnett. DARWIN: a program for docking flexible molecules. *Proteins*, 41:173–191, 2000.

[36] R. Huey, G.M. Morris, A.J. Olson, and D.S. Goodsell. A semiempirical free energy force field with charge-based desolvation j. computational chemistry. *J. Comput. Chem.*, 28:1145–1152, 2007.

[37] C.R. Corbeil, P. Englebienne, and N. Moitessier. Docking ligands into flexible and solvated macromolecules. 1. Development and validation of FITTED 1.0. *J. Chem. Inf. Model.*, 47:435–449, 2007.

[38] R. Thomsen. Flexible ligand docking using differential evolution. In *Proceedings of the 2003 Congress on Evolutionary Computation*, volume 4, pages 2354–22361, Piscataway, NJ (USA), 2003. IEEE Press.

[39] J.-M. Yang and C.-C. Chen. GEMDOCK: A generic evolutionary method for molecular docking. *Proteins*, 55:288–304, 2004.

[40] R. Thomsen and M. H. Christensen. MolDock: a new technique for high-accuracy molecular docking. *J. Med. Chem.*, 49:3315–3321, 2006.

[41] K. Shoemake. Euler angle conversion. In *Graphics gems IV*, pages 222–229. Academic Press Professional, Inc., San Diego, CA, USA, 1994.

[42] A. Kerzmann, J. Fuhrmann, D. Neumann, and O. Kohlbacher. Balldock/slick: a new method for protein-carbohydrate docking. *J. Chem. Inf. Model.*, 48:1616–1625, 2008.

[43] C.R. Guimarares and M. Cardozo. Mm-gb/sa rescoring of docking poses in structure-based lead optimization. *J. Chem. Inf. Model.*, 48:958–970, 2008.

[44] D.C. Thompson, C. Humblet, and D. Joseph-McCarthy. MM-GB/SA Rescoring of docking poses in structure-based lead optimization. *J. Chem. Inf. Model.*, 48:1081–1091, 2008.

[45] J. Lee and C. Seok. A statistical rescoring scheme for protein-ligand docking: Consideration of entropic effect. *Proteins*, 70:1074–1083, 2008.

[46] D.K. Gehlhaar, G.M. Verkhivker, P.A. Rejto, C.J. Sherman, D.R. Fogel, L.J. Fogel, and S.T. Freer. Molecular recognition of the inhibitor AG-1343 by HIV-1 protease: conformationally flexible docking by evolutionary programming. *Chem. Biol.*, 2:317–324, 1995.

[47] M. Stahl and M. Rarey. Detailed analysis of scoring functions for virtual screening. *J. Med. Chem.*, 44:1035–1042, 2001.

[48] R. Wang, Y. Lu, and S. Wang. Comparative evaluation of 11 scoring functions for molecular docking. *J. Med. Chem.*, 46:2287–2303, 2003.

[49] A. Oda, K. Tsuchida, T. Takakura, N. Yamaostu, and S. Hirono. Comparison of consensus scoring strategies for evaluating computational models of protein-ligand complexes. *J. Chem. Inf. Model.*, 46:380–391, 2006.

[50] V. Choi. YUCCA: an efficient algorithm for small-molecule docking. *Chem. Biodivers.*, 2:1517–1524, 2005.

[51] C.M. Venkatachalam, X. Jiang, T. Oldfield, and M. Waldman. LigandFit: a novel method for the shape-directed rapid docking of ligands to protein active sites. *J. Mol. Graph. Model.*, 21:289–307, 2003.

[52] J.-M. Yang, Y.-F. Chen, T.-W. Shen, B.S. Kristal, and D.F. Hsu. Consensus scoring criteria for improving enrichment in virtual screening. *J. Chem. Inf. Model.*, 45:1134–1146, 2005.

[53] T. Drezner and Z. Drezner. Gender-specific genetic algorithms. *INFOR*, 44:117–127, 2006.

[54] D. Mokkedem and A. Khellaf. Pareto-optimal solutions for multicriteria optimization of a chemical engineering process using a diploid genetic algorithm. *Intl. Trans in Op. Res.*, 15:51–65, 2008.

[55] G.M. Verkhivker, D. Bouzida, D.K. Gehlhaar, P.A. Reijto, S. Arthurs, A.B. Colson, S.T. Freer, V. Larson, B.A. Luty, T. Marrone, and P.W. Rose. Deciphering common failures in molecular docking of ligand-protein complexes. *J. Comput. Aided Mol. Des.*, 14:731–751, 2000.

[56] G.M. Verkhivker, D. Bouzida, D.K. Gehlhaar, P.A. Reijto, S.T. Freer, and P.W. Rose. Complexity and simplicity of ligand-macromolecule interactions: the energy landscape perspective. *Curr. Opin. Struct. Biol.*, 12:197–203, 2002.

[57] M. Vieth, J.D. Hirst, A. Kolinski, and C.L. Brooks III. Assessing energy functions for flexible docking. *J. Comput. Chem.*, 19:1612–1622, 1998.

[58] G.L. Warren, C.W. Andrews, A.-M. Capelli, B. Clarke, J. LaLonde, M.H. Lambert, M. Lindvall, N. Nevins, S.F. Semus, S. Senger, G. Tedesco, I.D. Wall, J.M. Woolven, C.E. Peishoff, and M.S. Head. A critical assessment of docking programs and scoring functions. *J. Med. Chem.*, 49:5912–5931, 2006.

[59] M.L. Verdonk, P.N. Mortenson, R.J. Hall, M.J. Hartshorn, and C.W. Murray. Protein-ligand docking against non-native protein conformers. *J. Chem. Inf. Model.*, 48:2214–2225, 2008.

[60] W.E. Hart and R.K. Belew. Optimization with genetic algorithm hybrids that use local search. In *Adaptive Individuals in Evolving Populations: Models and Algorithms*, pages 483–496. Westview Press, 1996.

[61] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[62] H. Abe, W. Braun, T. Noguti, and N. Go. Rapid calculation of first and second derivatives of conformational energy with respect to dihedral angles for proteins. General recurrent equations. *Comput. Chem.*, 8:239–247, 1984.

[63] J. Schmidt and H. Niemann. Using Quaternions for Parametrizing 3–D Rotations in Unconstrained Nonlinear Optimization. In T. Ertl, B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2001*, pages 399–406, Stuttgart, Germany, 2001. AKA/IOS Press, Berlin, Amsterdam.

[64] F.S. Grassia. Practical parameterization of rotations using the exponential map. *J. Graph. Tools*, 3:29–48, 1998.

[65] R. Breton, D. Housset, C. Mazza, and J.C. Fontecilla-Camps. The structure of a complex of human 17beta-hydroxysteroid dehydrogenase with estradiol and NADP+ identifies two principal targets for the design of inhibitors. *Structure*, 4:905–915, 1996.

[66] N. Narayana, S. Cox, S. Shaltiel, S.S. Taylor, and N. Xuong. Crystal structure of a polyhistidine-tagged recombinant catalytic subunit of cAMP-dependent protein kinase complexed with the peptide inhibitor PKI(5-24) and adenosine. *Biochemistry*, 36:4438–4448, 1997.

[67] E.A. Padalan, G.H. Cohen, and D.R. Davies. Refined crystal structure of the Mc/Pc603 Fab-phosphocholine complex at 3.1 angstroms resolution. *To be published*.

[68] D.W. Banner and P. Hadvary. Crystallographic analysis at 3.0-A resolution of the binding to human thrombin of four active site-directed inhibitors. *J. Biol. Chem.*, 266:20085–20093, 1991.

[69] E.E. Kim, C.T. Baker, M.D. Dwver, M.A. Murcko, B.G. Rao, R.D.Tung, and M.A. Navia. Crystal structure of HIV-1 protease in complex with Vx-478, a potent and orally bioavailable inhibitor of the enzyme. *J. Am. Chem. Soc.*, 117:1181–1182, 1995.

[70] I. Badger, I. Minor, M.A. Oliveira, T.I. Smith, and M.G. Rossmann. Structural analysis of antiviral agents that interact with the capsid of human rhinoviruses. *Proteins*, 6:1–19, 1989.

[71] H. Jhoti, O.M. Singh, M.P. Weir, R. Cooke, P. Murray-Rust, and A. Wonacott. X-ray crystallographic studies of a series of penicillin-derived asymmetric inhibitors of HIV-1 protease. *Biochemistry*, 33:8417–8427, 1994.

[72] W.E. Hart. *Adaptive global optimization with local search*. PhD thesis, Computer Science and Engeneering Department, University of California, San Diego, 1994.

[73] N. Go and H.A. Scheraga. Ring closure and local conformational deformations of chain molecules. *Macromolecules*, 3:178–187, 1970.

[74] C.A. Lipinski, F.L., B.W. Dominy, and P.J. Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced Drug Delivery Reviews*, 46:3–26, 2001.

List of Figures

List of Tables

A. Publications

Papers in refereed journals

A. Kerzmann, J. Fuhrmann, D. Neumann, and O. Kohlbacher. Balldock/Slick: a new method for protein-carbohydrate docking. *J. Chem. Inf. Model.*, 48:1616–1625, 2008.

J. Fuhrmann, A. Rurainski, H.P. Lenhof, and D. Neumann. A new method for the gradient-based optimization of molecular complexes. *J. Comput. Chem.*, 30:1371–1378, 2009.

K. Szalai, J. Fuhrmann, T. Pavkov, M. Scheidl, J. Wallmann, K.H. Brämswig, S. Vrtala, O. Scheiner, W. Keller, J.M. Saint-Remy, D. Neumann, I. Pali-Schäll, and E. Jensen-Jarolim Mimotopes identify conformational B-cell epitopes on the two major house dust mite allergens Der p 1 and Der p 2. *Mol. Immunol.*,40:1308–1317,2008.

Submitted papers in refereed journals

J. Fuhrmann, A. Rurainski, and D. Neumann. A fair comparison of population based search heuristics in ligand-receptor docking. *J. Comp. Aid. Mol. Des.*

J. Fuhrmann, A. Rurainski, H.P. Lenhof, and D. Neumann. A new Lamarckian genetic algorithm for flexible ligand-receptor docking. *J. Comput. Chem.*

Presentations at international conferences

K. Szalai, M. Scheidl, J. Fuhrmann, J. Wallmann, S. Vrtala, O. Scheiner, J.M. Saint-Remy, D. Neumann, I. Schöll, and E. Jensen-Jarolim. Generation of B-cell mimotopes of the group 2 mite allergens for epitope localization. Clemens von Pirquet Symposium, December 7.-9., 2006, Vienna/Austria.

K. Szalai, M. Scheidl, J. Fuhrmann, J. Wallmann, R. Brunner, S. Vrtala, O. Scheiner, J.M. Saint-Remy, D. Neumann, I. Schöll, and E. Jensen-Jarolim. Mimotopes localize conformational epitopes on two major house dust mite allergens and are promising candidates for epitope-specific immunotherapy. 2nd International Symposium on Molecular Allergology, April 22.-24., 2007, Rome/Italy.

J. Fuhrmann, A. Rurainski, H.P. Lenhof, and D. Neumann. A new method for the gradient based optimization of partially flexible molecules using derivatives of a fair quaternion parametrization. European Bioperspectives 2008 - Book of Abstracts, p. 322-323, October 7.-9., 2008, Hannover/Germany.

B. Curriculum Vitae

Name	Jan Fuhrmann
Adress	Waldstr. 48
	66583 Spiesen-Elversberg
	Germany
Date of birth	November 30, 1976
Place of birth	St. Ingbert, Germany
Citizenship	German

Education

2005–2009	PhD student
	Center for Bioinformatics
	Saarland University, Saarbrücken, Germany
2001–2005	Diplom
	Junior Research Group for Drug Transport
	Saarland University, Saarbrücken, Germany
1987–1996	Abitur
	Gymnasium am Krebsberg, Neunkirchen, Germany

Teaching Experience

2004 – 2009	Tutor Modeling drug transport with bioinformatics methods
2009	Tutor Parallel programming for bioinformatics