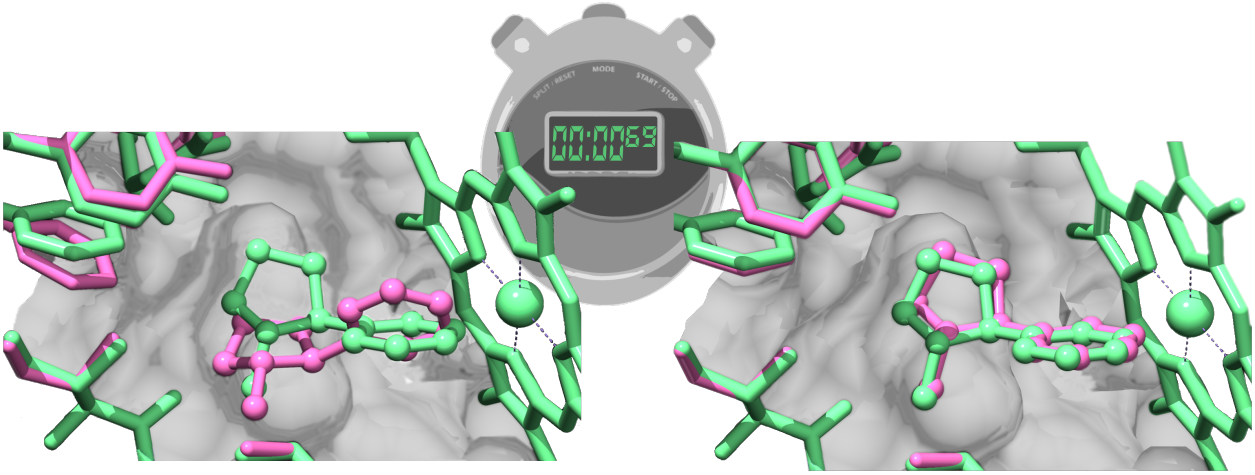
Accelerating Force Field-Based Optimizations of Protein-Ligand Complexes



Dissertation

with the aim of achieving a doctoral degree (Dr. rer. nat.) at the
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics, Universität Hamburg

submitted by

Lennart Erik Heinzerling

2014 in Hamburg

Day of oral defense:

The following evaluators recommend the admission of the dissertation:

Prof. Dr. Matthias Rarey

Prof. Dr. Stephan Olbrich

Abstract

Protein-ligand complexes play a key role in networks of biochemical signals and reactions. These, in turn, are the building blocks that define the behavior of biological mechanisms. The growing understanding of the process of protein-ligand complex formation permits to control these mechanisms. To induce beneficial changes to them, rational approaches, such as drug design and biotechnology, seek to discover targeted modifications to ligands and proteins. These modifications alter or inhibit the biological functionality of the macromolecules.

In these research processes, classical experimental methods have increasingly been complemented with computational approaches. These are usually more cost effective, less time consuming, grant access to larger molecular libraries, and offer detailed visual insights into biochemical processes. Fast computational methods that predict the interplay of proteins and ligands allow for large-scale screening experiments to discover novel pharmaceutical substances. Further, these methods can advance the productivity of researchers by enhancing the usability of their software tools.

However, these fast methods are typically guided by coarse and approximative knowledge-based or empirical scoring functions that quickly evaluate generated complexes. Consequently, forces that determine the formation of protein-ligand complexes are only partly captured. To compensate for this shortcoming, generated structures are frequently refined. A widely used approach for this is potential energy minimization, which relies on molecular mechanics force fields as objective functions. If applied appropriately, this approach drives structures closer to their experimentally measured conformations. However, performing force field-based minimizations is by no means a free ride, as they are notoriously time consuming. Thus, they impair the beneficial runtime behavior of otherwise fast computational methods.

This work addresses this drawback and presents Trooper, a versatile force field-based energy minimization method, and its accelerated version GPUperTrooper. Trooper is applied at the end of a pipeline of computational molecular design tools and is targeted at minimizing small ligands, amino acid side chains in protein binding sites, or both at the same time. For a thorough validation in these application scenarios, Trooper is applied to large data sets. This way, it is demonstrated that Trooper performs on par with commercially available and widely used state-of-the-art energy minimization methods. Furthermore, the range of Trooper's applicability for amino acid side chain optimizations is explored. At the same time, indications for a general limit to these types of optimizations are gathered.

GPUperTrooper is designed for single instruction, multiple data architectures and implemented for graphics processors manufactured by NVIDIA. It accelerates Trooper's protein-ligand complex optimizations. This fast optimizer produces results with a quality that is on par with Trooper. At the same time, GPUperTrooper speeds computations up by two or three orders of magnitude when compared to commercially available tools. Additionally, GPUperTrooper's computations typically save approximately 90% electric energy. This renders GPUperTrooper suitable for time- and cost-efficient application in large- and small-scale scenarios.

Zusammenfassung

Protein-Ligand-Komplexen kommt eine Schlüsselrolle in biochemischen Signal- und Reaktionsnetzwerken zu, welche wiederum das Verhalten ganzer biologischer Mechanismen bestimmen. Die zunehmende Erforschung der Bildung von Protein-Ligand-Komplexen stellt somit eine entscheidende Grundlage für die Beeinflussung dieser Prozesse dar. Um diese vorteilhaft zu modifizieren werden mit rationalen Ansätzen, beispielsweise der Biotechnologie und dem modernen Wirkstoffentwurf, gezielte Modifikationen an Liganden und Proteinen erforscht. Diese Modifikationen verändern oder hemmen die Funktionalität der Makromoleküle.

In diesen Forschungsvorhaben werden klassische, experimentelle Methoden zunehmend von rechnergestützten Verfahren ergänzt. Diese sind für gewöhnlich kosten- und zeiteffektiver, bieten Zugang zu größeren, virtuellen Molekülmengen und ermöglichen die Visualisierung biochemischer Prozesse. Besonders schnelle, rechnergestützte Methoden zur Vorhersage der Struktur von Protein-Ligand-Komplexen ermöglichen die Analyse großer Datensätze, beispielsweise mit dem Ziel des Auffindens neuer pharmazeutischer Wirkstoffe. Des Weiteren ermöglichen diese Methoden Produktivitätssteigerungen in der Forschung, da sie die Gebrauchstauglichkeit der dort eingesetzten Software-Werkzeuge verbessern.

Allerdings basieren diese schnellen Methoden mehrheitlich auf approximativen und heuristischen, empirischen oder wissensbasierten Bewertungsfunktionen. Diese ziehen Kräfte, welche die Bildung von Protein-Ligand-Komplexen bestimmen nur teilweise in Betracht. Aus diesem Grund werden von schnellen Methoden generierte Molekülstrukturen häufig mit nachgelagerten Verfahren verfeinert. Weit verbreitet ist dabei die Minimierung der potentiellen Energie der Strukturen. Dieser Ansatz nutzt Molekülmechanik-Kraftfelder als Zielfunktionen, was ihn sehr zeitintensiv macht. Global betrachtet verschlechtert sich somit das vorteilhafte Laufzeitverhalten der ansonsten schnellen rechnergestützten Generierungsmethoden.

Die folgende Forschungsarbeit befasst sich mit der Beseitigung dieses Nachteils. Dazu werden die auf Molekülmechanik-Kraftfeldern basierende Minimierungsmethode Trooper und ihre beschleunigte Version GPUperTrooper eingeführt. Trooper wird am Ende einer Kette von rechnergestützten Molekülentwurfs-Software-Werkzeugen eingesetzt und optimiert an dieser Stelle kleine Liganden, Seitenketten von Aminosäuren in Proteinbindungstaschen oder beides in Protein-Ligand-Komplexen. Zu Validierungszwecken wird Trooper auf größeren Datensätzen getestet. Auf diese Weise wird gezeigt, dass von Trooper verbesserte Strukturen ähnlich nah an experimentell bestimmte Strukturen geführt werden wie jene, die von kommerziell verfügbaren, aktuellen Methoden verfeinert werden. Des Weiteren wird der Anwendungsbereich von Trooper für die Optimierung von Aminosäureseitenketten untersucht. Diese Experimente liefern gleichzeitig Indizien für eine generelle Grenze der Anwendbarkeit von Seitenketten-Optimierungen.

GPUperTrooper wurde für moderne SIMD-Architekturen entworfen und wurde auf Grafikprozessoren der Firma NVIDIA implementiert. Dieser Algorithmus beschleunigt Troopers Protein-Ligand-Komplex-Optimierungen und liefert dabei Strukturen von gleicher Qualität. Im Vergleich zu kommerziell verfügbaren Methoden arbeitet GPUperTrooper allerdings um zwei bis drei

Größenordnungen schneller. Zusätzlich spart GPUperTrooper etwa 90% an elektrischer Energie ein. Somit ist GPUperTrooper eine ideale Verfeinerungsmethode für Ergebnisse, die von besonders schnellen, rechnergestützten Verfahren geliefert werden. Dieses gilt sowohl für Anwendungen im Hochdurchsatz- als auch im Desktop-Bereich.

Contents

CHAPTER 1

Introduction

1.1. Motivation

Modern research has deciphered a growing number of biological mechanisms that have been mapped to networks of biochemical signals and reactions. The latter are typically catalyzed in binding sites of proteins. Therein, small molecules, so called *ligands*, are transformed and undergo a change in their biological role. Alternatively, a ligand can block a binding site and, as a result, *inhibit* a protein. Both scenarios require the formation of protein-ligand complexes, a process that has been explored for more than a century. The lock and key principle postulated in 1894 [**1**] is considered the first landmark theory in this field. According to this principle, protein and ligand are rigid entities whose complementary shapes cause their close interaction. About 120 years since this early concept, the idea of an interplay of forces exerted by ligands and proteins prevails. These forces lead to mutual conformational reorganizations, during which proteins and ligands exhibit varying degrees of structural flexibility. The result is a low energy equilibrium state [**2**]. In this state, the ligand is covalently or non-covalently bound in a cavity of the protein. This deeper understanding is one of the decisive foundations for rational molecular design approaches in pharmaceutical and biotechnical research. Two examples of these are *rational drug design* and *protein engineering*.

Rational drug design regards pathological states as a result of networks of biochemical reactions. Among these, key reactions are identified and proteins mediating them become targets of intense research. Its efforts aim to identify chemical substances that inhibit, activate, or modify the functionality of the respective target protein. A preferably exhaustive search constitutes the first principal step in this process. This search is centered around discovering ligands that bind to a given protein target. To accomplish this, millions of small molecules are tested in protein assays. These large-scale experiments are named *high throughput screenings*. The following stage refines these obtained ligands. For this, scientists introduce molecular optimizations based on their experience and empirical rules. Results of these efforts are validated with more elaborate experimental assays. This stage is referred to as *lead optimization*.

Biotechnology seeks to synthesize products like nutrition supplements, pharmaceutical substances, and cleaning agents with microorganisms and isolated enzymes. Protein engineering enhances their throughput and specificity. For this, the effects of exchanging amino acids in protein binding sites are studied and exploited. Experiments are set up to produce large numbers of subtypes of an enzyme. Here, a common approach is the random induction of mutations in

the enzyme. A subsequent experimental selection identifies crucial modifications that enhance productivity. A further refinement stage, comparable to lead optimization, then follows.

Experimental methods employed in the introduced research fields are increasingly complemented with computational approaches. These are usually more cost effective, less time consuming, grant access to larger molecular libraries, and offer detailed visual insights into biochemical processes. The computational counterpart to high throughput screening is *virtual screening*. Furthermore, in the lead optimization stage, molecular modeling tools help modify amino acid side chains and parts of ligands. The effects of these refinement attempts are estimated using the same tools.

A common computational method for virtual screening is *molecular docking*. This approach rapidly places ligands into protein binding sites. This requires a large amount of alternative structural arrangements of the small ligand molecules. These arrangements are called *conformations* and are defined by rotations about interatomic bonds. Conformations are often pre-computed by *conformation generators*. Alternatively, they are incrementally constructed when fitting the ligand to the binding pocket. Docking is also a useful tool in the lead optimization phase. Additionally, altering ligands manually and then applying the refinement methods introduced next is common practice.

An exemplary method for protein engineering is *computational mutagenesis*. It exchanges amino acids in protein binding pockets. In this process, amino acid side chain conformations are generated in accordance with empirically derived rotamer libraries.

The introduced computational methods, namely conformation generation, docking, and computational mutagenesis, are most often guided by fast approximative knowledge-based or empirical scoring functions. These grant a quick evaluation of generated amino acid side chain and ligand conformations and poses. Conversely, forces that determine the formation of protein-ligand complexes are only partly captured. To compensate for this shortcoming, generated structures are frequently refined. A widely used approach for this is potential energy minimization, which relies on molecular mechanics force fields as objective functions.

Molecular mechanics force fields describe the potential energy (in short referred to hereinafter as *energy*) of a system of atoms. For this, functions grounded in classical physics are adapted and specifically parameterized for molecular interactions on an atomic scale. These parameterizations are derived from both quantum mechanical calculations and experimental results.

This work is centered around force field-based energy minimizations that are applied at the end of a pipeline of computational molecular design tools. Therefore, this type of minimizations is hereinafter also referred to as *downstream optimizations* or *post-optimizations*. If applied appropriately, post-optimizations drive structures closer to their experimentally measured conformations. At the same time, they compensate for structural overlaps, which preceding steps may introduce. To support this process, constraints can be imposed on predefined degrees of freedom of a molecular system. This way, existing knowledge on it is incorporated into the optimization procedure. However, applying force field-based optimizations is by no means a free ride,

as they are notoriously time consuming. Thus, they impair the cost-effectiveness of otherwise fast computational methods in large-scale application scenarios, such as virtual screening. When integrated into modeling tools that are applied in lead optimization, post-optimizations affect the interactivity of design processes. This entails repercussions on team productivity and user satisfaction, as pointed out in Section 3.1.

Computational power is thus a limiting factor for applying downstream optimizations. Another class of methods, namely *molecular dynamics* (*MD*) simulations, faces the same challenge. For this reason, a large number of MD algorithms have recently been adapted to *graphics processing units* (*GPUs*). *General-purpose computing on graphics processing units* (*GPGPU*) is a widely applied approach for reducing runtimes. Section 2.6 introduces publications that propose different ways of porting MD simulations to the GPU. However, these efforts focus on speeding up dynamics simulations of larger molecular systems. In contrast, this work concentrates on quick downstream optimizations of molecules in protein binding pockets. These systems typically comprise less than 1,500 atoms. Furthermore, the complexity of downstream optimization can be reduced by limiting the set of degrees of freedom.

1.2. Project Overview and Main Contributions

This work presents *TRAmber's Object-oriented OPtimizER* (*Trooper*), a versatile downstream optimization method that uses the molecular mechanics force field *SuperTrAmber* (see Section 3.4.1) as an objective function. Trooper is suitable for all application scenarios introduced in the preceding section. Hence, it is explicitly designed for minimizing small ligands, amino acid side chains in protein binding sites, or both at the same time. Into this process, users can incorporate chemical knowledge by imposing constraints on dihedral angle rotations and ligand movements.

For a thorough validation in ligand, side chain, and protein-ligand complex optimization scenarios, Trooper is applied to large data sets. This way, it is demonstrated that Trooper performs on par with commercially available and widely used state-of-the-art energy minimization methods. Furthermore, the range of Trooper's applicability for amino acid side chain optimizations is explored. Thus, indications for a general limit to these types of optimizations are gathered.

GPUperTrooper, an algorithm designed for *single instruction, multiple data* (*SIMD*) architectures and implemented for GPUs manufactured by NVIDIA, accelerates Trooper's protein-ligand complex optimizations. This fast optimizer produces results with quality that is on par with Trooper. At the same time, GPUperTrooper speeds computations up by two or three orders of magnitude when compared to commercially available tools. Additionally, GPUperTrooper's computations typically save approximately 90% electric energy. This renders GPUperTrooper suitable for time- and cost-efficient large- and small-scale application scenarios.

This work demonstrates that Trooper and GPUperTrooper distinguish themselves from previous work in the following aspects:

- There are only a few downstream optimization methods whose efficacy in conjunction with docking tools has been shown. This work demonstrates that Trooper improves docking poses generated by TrixX (see Section 3.8.3), a state-of-the-art ultra-fast docking tool. For validating this, docking runs are carried out on protein-ligand complexes of the community-wide used Astex Diverse Set [**3**].
- It has been disputed that force field-based downstream optimizations are beneficial when applied prior to a rescoring stage with another objective functions [**4**, **5**]. This research refutes this hypothesis by showing that Trooper improves the rescoring efficacy of HYDE (see Section 3.8.1), a scoring function based on a consistent model for hydrogen bond and dehydration energies.
- As a downstream optimizer for the conformation generator CONFECT (see Section 3.8.5), Trooper incorporates this tool's dihedral angle constraints. In this fashion, Trooper supplements force field potentials with knowledge-based constraints derived from co-crystallized conformations by Schärfer [**6**]. With these, the bias of the SuperTrAmber force field towards gas phase conformations is circumvented. Additionally, a highly specific and transparent association between dihedral angle constraints and torsion patterns is established.
- GPUperTrooper is the only algorithm developed for the SIMD architecture of GPUs that is specifically designed for quick force field-based minimizations of molecules in protein binding sites.
- Compared to widely used commercially available methods, GPUperTrooper optimizes ligands and amino acid side chains in protein binding pockets about 10 times more energy efficient. Hence, GPUperTrooper saves time, energy, and money in large-scale application scenarios.
- In desktop usage scenarios, GPUperTrooper optimizes binding pockets about two orders of magnitude faster than methods operating on a single CPU core. Hence, GPUperTrooper grants access to interactive force field-based minimizations, e.g. for lead optimization tools.
- Coupling GPUperTrooper with the fastest available docking tools like TrixX and PhDOCK [**7**] only negligibly affects their overall runtime behavior.

1.3. Thesis Overview

This work is structured as follows:

The Introduction continues with sections on force field-based energy minimizations and the architecture of NVIDIA graphics processors.

Chapter 2 is a survey of the state-of-the-art of developments in fields relevant to fast force field-based downstream optimizations. It provides an overview on versatile stand-alone optimization tools, as well as conformation generation, docking and protein engineering tools that include downstream optimizations. Furthermore, GPU portations of molecular mechanics force field kernels are discussed.

Chapter 3 introduces methods applied in this work that have been previously developed without contributions of the author of this work. Relevant molecular mechanics force fields, in particular SuperTrAmber, are covered. Furthermore external tools to which Trooper and GPUperTrooper are compared in evaluation experiments are introduced. Finally, statistical methods for hypothesis testing are presented.

Chapter 4 details the major building blocks of Trooper and GPUperTrooper. Among them is a structure for representing molecular data, a framework for implementing molecular mechanics force fields, GPU-based data structures and algorithms that accelerate force computations with the SuperTrAmber force field, and a versatile framework for designing optimization modules based on simulated annealing.

Chapter 5 comprehensively describes the preparation and composition of data sets used in experiments in this work. Furthermore, technical specifications of all benchmarking machines are detailed.

Chapter 6 reports on the application of Trooper and GPUperTrooper. In post-optimizations after docking runs with TrixX, this work's methods drive ligands closer to their respective biologically active conformation. Also, the performance of the rescoring function HYDE is improved. In a next step, the usefulness of these methods for amino acid side chain and flexible protein-ligand optimizations is demonstrated. At the same time, these methods are evaluated by comparative experiments with external methods. Additionally, the range of application of this work's and the external methods is explored. Along with the detailed experiments, GPUperTrooper is thoroughly benchmarked and externally evaluated. Finally, this research shows that Trooper improves results produced by the conformation generator CONFECT while taking its dihedral angle constraints into account.

Chapter 7 comprises a summary of the content of this work and outlines perspectives for future research on downstream optimizations and acceleration approaches targeted at processors with SIMD technology.

The Appendix contains a manual for operating the downstream optimization tool and outlines the structures of developed software libraries. Furthermore, this part comprises acknowledgements, a formal declaration on oath, and the copyright notice for this work.

1.4. Force Field-Based Molecular Energy Minimizations

Molecular mechanics force fields are trained on data resulting from both experiments and quantum chemical calculations. Based on this data, force field functions determine the potential

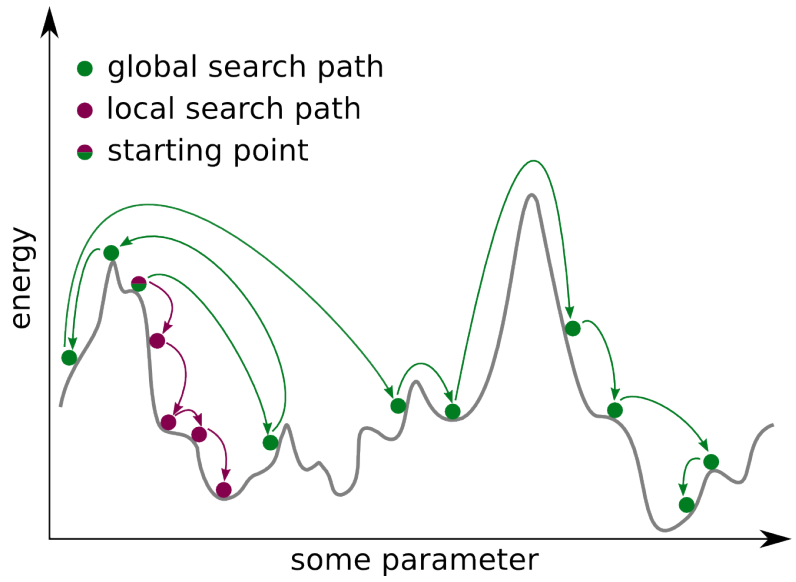


FIGURE 1.1. Typical search paths produced by local and global energy minimization algorithms.

energy of molecular systems with the coordinates of its atoms as parameters. Hence, the potential energy varies with the atom coordinates, which yields an energy hypersurface. The minimum points of this surface are assumed to correspond to stable and thus, biologically favorable, states of the molecular system. While a minimum point in general is called a *local minimum*, the minimum with the lowest energy is termed the *global minimum*. Energy minimization algorithms, referred to as *optimizations*, find states of a molecular system that correspond to minimum energy points. In this work, optimization methods are categorized as either *local* or *global*.

This work defines *local minimizations* as greedy local searches. As illustrated in Figure 1.1, they are guided by the slope of energy functions. With a small number of steps, they detect the local minimum that is nearest in a downhill sense from the starting point of the optimization. In this process, energy barriers with positive slopes cannot be crossed. Thus, locating more than one local minimum or the global minimum requires different starting points.

Most often, local optimization methods are *gradient-based*. This reflects that the first, and in some cases, the second derivative of the energy function is computed for all atom coordinates. The magnitude of changes to these varies with the slope of the energy function. Conjugate gradient [**8**] and steepest descent [**9**] are commonly applied gradient-based methods. There are also local minimization approaches that operate solely on the energy function. The most popular among them is the downhill simplex method [**10**].

The term *global minimization* refers to randomized local searches. Figure 1.1 shows two of their key traits. First, they perform randomized steps without taking the slope of the energy function into account. Second, this class of algorithms can overcome energy barriers. Thus, global minimizations are typically more suitable rough energy functions and potentially detect

TABLE 1.1. Excerpt from technical specifications of NVIDIA GPUs of different generations

generation	shared memory	memory banks	cores per SM	max. SMs per GPU	ref.
Tesla	16 kByte	16	8	30	[**12**]
Fermi	48 kByte	32	32-48	16	[**12**, **13**]
Kepler	48 kByte	32	192	15	[**12**, **14**]

the global minimum of an objective function. Conversely, they are more time-consuming than local minimizations and may yield a result far away from the global minimum. Their outcome furthermore depends on the starting point, in which they resemble local minimizations.

Self-developed optimizations used in this work use *simulated annealing (SA)*, a global minimization method comprehensively introduced in Section 3.3.1. Third-party methods often rely on *genetic algorithms* whose search heuristic mimics the process of natural selection.

1.5. GPU Architecture and Terminology

State-of-the art graphics processors are widely and cheaply available. Desktop systems host them as part of graphics cards that cost less than 500 euros. Furthermore, specially designed accelerator cards for workstations and supercomputers are equipped with cutting-edge GPUs. Their general-purpose computing functionality can be conveniently accessed via well-documented application interfaces. Most accelerated algorithms described in this work are implemented with CUDA [**11**] and thus are targeted at NVIDIA GPUs. The following introduces the CUDA programming paradigm and the NVIDIA GPU architecture, as well as terminology linked to it.

1.5.1. Multiprocessors. A NVIDIA GPU is composed of multiple SIMD processing units (see Figure 1.2.a). These are named *streaming multiprocessors* (*SM*). Synonymously, the term *multiprocessor* is employed. Depending on the GPU generation (see Table 1.1), they are composed of 8 to 192 smaller processing units [**12**] that are named *CUDA cores* or, in short, *cores* (see Figure 1.2.b). CUDA cores process *threads*. These are assigned to them by SMs which, in turn, process multi-dimensional arrays of threads that are termed *thread blocks*. They are decomposed by the SM into *warps*. These are arrays of 32 threads [**12**], which start program execution at the same address.

In accordance with the SIMD model, all threads of a warp execute the same sequence of instructions in lockstep, yet on differing data. Only one execution path can be followed by a warp at a time. Thus, divergence of execution paths within warps is detrimental for performance.

In contrast, divergence of execution paths between different warps does not entail performance drawbacks. Yet the exchange of data in a defined state between warps as well as SMs—called *synchronization*—constitutes a costly operation. On older GPU generations it is even infeasible.

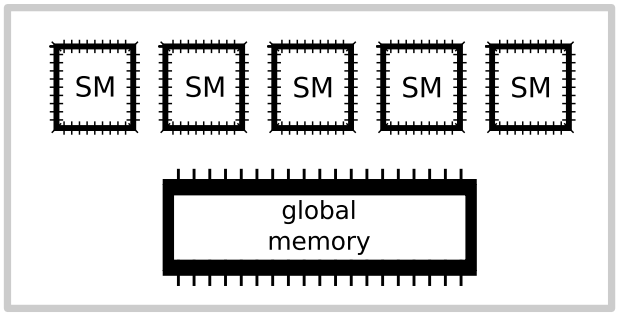


Figure 1.2.A: Sketch of general layout of NVIDIA GPUs. Several SIMD multiprocessors share access to global memory.

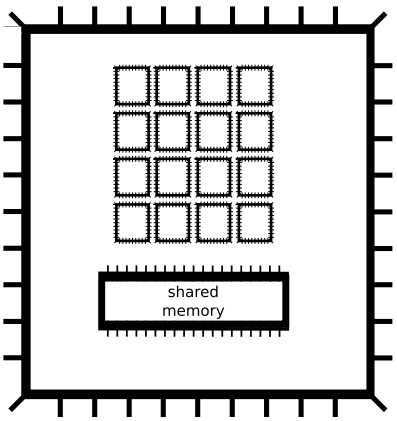


Figure 1.2.B: Sketch of general layout of NVIDIA GPU multiprocessors. Several single cores perform computations and have access to on-chip shared memory.

FIGURE 1.2. Layout of NVIDIA graphics processors at two scales.

1.5.2. Memory hierarchy. NVIDIA GPUs have several layers of memory. Most important to this work are *shared* and *global memory*. Shared memory is located on a SM and is used by its CUDA cores (see Figure 1.2.b). It permits quick access and is limited in size as there are 16 to 48 kByte available per SM [**13**, **14**]. It can be regarded as an equivalent to L1 cache on x86 CPUs. However, the programming paradigms for CPUs and GPUs differ as it is completely up to the programmer to control shared memory usage.

Shared memory is arranged in equally sized modules, named *banks*. They can be accessed simultaneously, but memory requests for the same bank are serialized. This is detrimental for data transfer rates. Among the most important measures for attaining high bandwidth is thus avoiding *memory bank conflicts*.

All SMs of a device share access to global memory (see Figure 1.2.a). Vast amounts of it can be allocated. However, frequent data transfer between SMs and global memory should be avoided. Latency coupled to these operations is more than 100 times higher, while bandwidth is considerably lower compared to on-chip shared memory [**12**]. Transfers between GPU device and host main memory are even more disadvantageous, as they have low bandwidth and entail a high overhead [**12**].

1.5.3. Kernels. CUDA allows the user to define *kernels*. These are functions that are executed by threads. The number of kernel executions thus depends on the user-defined number of threads. Each of these threads is assigned a unique ID, which permits controlling them individually.

For executing a kernel, the user triggers (i.e., calls) it. However, frequent calls to GPU kernels should be avoided as they are coupled with a latency of approximately $5\,\mu s$ [**15**]. This is about three orders of magnitude larger compared to x86 architecture.

1.5.4. Algorithmic implications. In summary, architectural details have to be considered when designing algorithms for GPUs. On the kernel level, programmers should utilize shared memory without causing memory bank conflicts. This way, access to global memory is avoided. Furthermore, execution branch divergence and synchronizations have to be circumvented.

Having designed an efficient kernel, it is up to the programmer to integrate it into entire algorithms while sustaining the speedup. This is a major challenge, as data transfer between a host system and a GPU device, as well as the number of kernel calls, have to be kept at a minimum.

CHAPTER 2

Scientific Context

This work is centered around Trooper and its accelerated version GPUperTrooper. Trooper is a versatile and fast optimization method that drives molecules towards experimentally determined bioactive conformations. This method optimizes small molecules as well as binding sites of protein-ligand complexes. Its application scenarios cover downstream optimizations after molecular docking and virtual screening, amino acid side chain optimizations for enzyme engineering, lead optimizations, and small molecule conformation generation. These scenarios have in common that typically a tool with a simple heuristic objective function produces a set of molecule conformations. As demonstrated in the experimental sections, Trooper enhances their average quality. At the same time, its accelerated version GPUperTrooper sustains the speed of simple heuristic approaches. Furthermore, it optionally incorporates chemical knowledge. This way, spatial regions where atoms should be located, as well as favorable angles of rotatable dihedral bonds, can be predefined. The following section discusses available methods that address the same application scope as Trooper.

2.1. Stand-Alone Optimization Tools

This section introduces tools that offer a wide application range. It focuses on those suitable for at least two, or have been employed in at least one, of the scenarios described previously. This research analyzes each tool for its range of applicability, how it handles the selection of flexible protein regions, and whether it offers constraints for optimizations. Furthermore, the research discusses whether the efficacy of the tool for downstream optimizations has been shown. Finally, this work evaluates whether published runtimes suffice for cost- and time-efficient large-scale optimization runs, as well as an interactive workflow. As discussed in Section 3.1, this requires runtimes below $1\,s$ per protein-ligand complex minimization.

2.1.1. AMMOS. Pencheva et al. published AMMOS [**16**], a tool that minimizes small molecules or protein-ligand complexes. This tool grants access to several gradient-based and stochastic optimization procedures as well as molecular mechanics force fields that are part of the AMMP package [**17**]. Pencheva et al. demonstrate the efficacy of AMMOS at improving docked poses. Furthermore, they show that their downstream optimizations enhance enrichment rates in virtual screening in some anecdotal cases.

AMMOS allows for defining a flexible protein region around ligands. However, neither spatial nor dihedral constraints are provided. On a single 3.0 GHz core of an Intel Xeon CPU, AMMOS

on average runs 0.3 *s* for optimizing small molecules, 24 *s* for optimizing ligands in rigid binding pockets, and 44 *s* when treating atoms within a sphere with undefined radius around ligands as flexible. For all optimization runs, the employed cutoffs are undefined. These runtimes are quite promising. However, they neither suffice for performing cost-efficient large-scale optimization runs nor for an interactive workflow in a workstation usage scenario.

2.1.2. MacroModel. MacroModel [**18**] is a molecular modeling program that offers access to a wide range of molecular mechanics force fields. For performing minimizations, several gradient-based methods are provided. Dihedral rotations can be constrained by force constants. This keeps angles in a selected place. Multiple constraints on one torsion angle are not supported. Explicit runtimes for minimization procedures are not specified.

Perola et al. applied MacroModel in an evaluation study [**19**] that assessed the docking tools GOLD [**20**, **21**], ICM [**22**, **23**], and Glide [**24**]. For this study, gradient-based downstream minimization of top-scored ligands resulting from cognate docking runs were performed. For this, the *MMFF94* (see Section 3.4.3) and the OPLS-AA [**25**, **26**] force fields were used as objective functions. These optimizations improved the percentage of correct poses (as defined in Section 3.2) for GOLD from 48% to 62%. For the other tested docking tools, the number of correct poses stayed approximately constant. Perola et al. attribute this to force field-based post-optimization procedures already included in ICM and Glide but not in GOLD. It is noteworthy that other literature sources claim that GOLD comprises downstream optimizations [**27**]. Perola et al. also conclude that the effects of employing an alternative force field are negligible.

All docking tools, plus the post-optimization procedure, were also tested in virtual screening scenarios. For these, Perola et al. report that post-optimizations dramatically improve enrichment rates for systems with tight binding pockets. Finally, runtimes for optimizations are stated, which range from 30 *s* to 60 *s* per pose.

In summary, MacroModel is a versatile tool and offers limited constraints for dihedral angle rotations. Improvements are necessary for incorporating further knowledge on dihedral angle conformations. Apart from that, Perola et al. showed that post-optimizations can improve docking and virtual screening quality. Accelerating MacroModel in this scenario would improve its usability and cost-effectiveness, as optimizations currently take 30 *s* to 60 *s* per pose.

2.1.3. MOE. The *Molecular Operating Environment* (*MOE*) [**28**], which is commercially available, is a fully integrated software package for drug discovery. It contains a module for all-purpose local gradient-based molecular minimizations. MOE offers a broad range of molecular mechanic force fields as objective functions. For defining flexible regions, MOE allows labeling atoms as rigid. Additionally, assigning tether weights to an arbitrary set of atoms constrains their movements and a single dihedral angle restraint interval can be assigned per set of four atoms. Miura et al. [**29**] and Klenner et al. [**30**] employ MOE as a downstream optimization module (see Section 2.2). Yet, they fail to analyze its runtime and efficacy. However, a thorough experimental series on this issue is conducted in this work. As shown later, the runtime of MOE's minimization

module is far above the interactivity threshold (see Section 3.1). Consequently, applying it for large-scale optimization runs entails disproportionate costs.

2.1.4. Rosetta. The Rosetta software suite for modeling macromolecular structures [**31**,**32**] is built with the ROSETTA3 software library [**33**]. It offers several gradient-based optimization procedures and an all-atom knowledge-based energy function as objective function, which was initially developed for protein design [**34**]. However, Rosetta does not include an off-the-shelf minimization tool. Instead, this has to be programmed with the ROSETTA3 library modules. Despite this, the docking tools DARC [**35**] and RosettaLigand [**36**, **37**] described in Section 2.2 employ Rosetta for downstream minimizations.

2.1.5. SZYBKI. The yet unpublished commercially available tool SZYBKI [**38**] minimizes small molecules, as well as protein binding pockets, without or in the presence of ligands. The MMFF94 and MMFF94s force fields (see Section 3.4.3) serve as objective functions. SZYBKI allows defining flexible regions on atomic basis or by SMARTS patterns [**39**]. It either optimizes entire amino acid residues or their side chains only. Furthermore, harmonic constraints can be imposed on all or a SMARTS pattern-based selection of atoms. However, dihedral rotations cannot be constrained.

According to claims on SZYBKI's homepage [**38**], 200 small molecules are optimized in 60 s. Consequently, an optimization takes 0.5 s on average. It is unclear whether this figure refers to a parallelized run and what kind of machine was used. However, the asymptotic runtime of molecular mechanics force field-based optimization is quadratic. As binding pockets typically contain at least one order of magnitude more atoms than small molecules, the data suggests that SZYBKI's runtime is at least 50 s for this kind of optimization. If that assumption is true, this tool is not suitable for granting an interactive workflow. Furthermore, it renders large-scale optimization campaigns time consuming and costly.

2.1.6. YASARA. *Yet Another Scientific Artificial Reality Application* (*YASARA*) [**40**] is a commercially available molecular-graphics, -modeling and -simulation program. It comprises a simulated annealing-based module for all-purpose molecular minimizations. Algorithmic details on this module have not been published. YASARA allows selecting a flexible region and an objective function from a wide range of molecular mechanics force fields. However, constraining parts of molecules or dihedral rotations is only possible for MDsimulations. Nabuurs et al. [**41**] demonstrate YASARA's efficacy in downstream minimizations (see Section 2.2) and analyze its runtime. This work benchmarks YASARA and shows that it cannot be used for interactive minimizations or time- and cost-efficient large-scale optimization runs.

2.1.7. Summary. This section introduces the all-purpose energy minimization tools MOE, YASARA, and MacroModel, as well as the downstream optimization tools AMMOS and SZYBKI. All permit a precise selection of flexible protein regions. Furthermore, the efficacy in downstream optimizations has been demonstrated for YASARA, MacroModel, and AMMOS. However, all published runtimes for these tools indicate that speedups of two orders of magnitude are necessary

for cost-efficient applications in large-scale drug development scenarios. The same applies to interactive workflow scenarios. Additionally, only MacroModel and MOE partially support fine-grained dihedral angle rotation constraints.

2.2. Optimizations in Docking and Virtual Screening Methods

The following introduces docking tools that include downstream or intermediate force field-based minimizations. In the second type of tools, minimization procedures are clearly separable steps of the overall docking algorithm. This research analyzes each tool to determine which degrees of freedom are included in the minimization procedure. Furthermore, this work investigates this procedure's efficacy and its impacts on the runtime behavior and rescoring schemes of the docking procedure. The latter question is quite interesting as Cole et al. and O'Boyle et al. raised doubts that rescoring, especially with non-smooth functions, is effective when preceded by optimizations with a different objective function [**4**, **5**]. Finally, a summary in Table 2.1 provides an overview that concludes this section.

2.2.1. Docking approaches with downstream optimizations.

ADAM. ADAM [**42**], which was published by Mizutani et al., is a grid-based docking procedure. For removing clashes and modeling induced fit effects, it employs a post-optimization module named BLUTO. This module combines steepest descent and L-BFGS [**43**] to minimize ligands and amino acid side chains in binding pockets. Here, the AMBER94 force field (see Section 3.4.4) serves as an objective function. Mizutani et al. report runtimes between $1\,s$ and $115\,s$ for performing a docking run on an Intel Xeon running at 3.06 GHz. As neither the runtime nor the efficacy of the BLUTO module is specified, comparisons with ADAM are impossible.

CRDOCK. Quite recently, Cabrera et al. reported on CRDOCK [**44**], a follow-up of CDOCK [**45**]. CRDOCK flexibly docks ligands into rigid binding pockets. Several docking protocols are available. By default, a BFGS [**43**] rigid-body minimization post-processes resulting poses. For this, AMBER-like potentials stored in a three-dimensional grid serve as objective functions. Cabrera et al. report a total average runtime of $10\,s$ per conformation for their procedure, measured on a 3.3 GHz Intel Core i5 CPU. However, runtimes for the optimization procedure are not stated and its efficacy is not analyzed.

DARC. Recently, Khar et al. published DARC [**35**], a method for docking small molecules into protein surface pockets. Based on a shape-matching algorithm, DARC places pre-generated ligand conformations into rigid protein pockets. These generated poses are ranked according to a crude and fast scoring scheme. In accordance with this ranking, DARC minimizes all rotatable dihedral angles, in both the protein and the ligand for the top 10% of all generated complexes. At this point, the Rosetta all-atom force field serves as an objective function (see Section 2.1). Finally, DARC reranks minimized complexes based on energetic and structural considerations.

Khar et al. extensively analyze runtimes and speedups achieved through porting the core DARC algorithm to GPU. However, neither the efficacy nor the runtime of performed downstream

optimizations are discussed. Furthermore, the effect of optimizations on the reranking scheme is not analyzed.

Fleksy. Nabuurs et al. compiled a tool chain named Fleksy [**41**] for fully flexible docking. Fleksy employs FlexE [**46**] to perform ensemble docking. Resulting protein-ligand complexes are optimized with YASARA (see Section 3.9.1), which uses the Yamber2 force field [**47**] as an objective function. Subsequently, the optimized complexes are rescored with a consensus function that encompasses FlexX [**48**] and *piecewise linear potential* (*PLP*) [**49**] scores as well as interaction energies of the Yamber2 force field.

Fleksy produces a single docking pose in about $180\,s$. Nabuurs et al. attribute the bigger part of this runtime to the optimization step that "can take up to a few minutes per docking pose" [**41**, p. 6511]. However, an analysis yields that omitting this step deteriorates the RMSD of the best ranking solution in 10 out of 16 cases and the RMSD of the overall best solution is worsened in 9 out of 16 cases. In a further experiment, generated docking poses were subjected to the optimization procedure. For poses with an initial RMSD below $1.5\,\mathring{A}$, Nabuurs et al. report RMSD reductions by $0.31\,\mathring{A}$ to $0.41\,\mathring{A}$.

In summary, the results gathered by Nabuurs et al. indicate that downstream optimizations shift docking poses closer to the respective crystal structure conformation. Also, rescoring functions apparently produce more realistic rankings when applied to optimized poses. In addition, it would be interesting to determine whether optimizations with the rescoring function are more effective than with the Yamber2 force field. Furthermore, the usability and cost-efficiency of Fleksy would benefit from a faster optimization procedure.

FLOG. The *Flexible Ligands Oriented on Grid* (*FLOG*) algorithm by Miller et al. [**50**] is similar to the DOCK procedure (see Section 2.2.2) in all its major steps. FLOG employs a simplex-based [**10**] rigid-body optimization procedure to refine ligand poses as a final step. Miller et al. demonstrate in an anecdotal case study that FLOG's post-optimization procedure enhances the score of ligand poses while doubling the runtime of the docking procedure. Therefore, FLOG constitutes a further docking algorithm that would benefit from a fast downstream optimization procedure.

FRED. With pre-generated conformations, FRED [**51**] produces docking poses by exhaustively searching the space defined by ligand rotations and translations. During this search, proteins are kept rigid and Chemgauss [**51**] and CGO [**51**] serve as objective functions. The top 100 scored poses are subjected to a rigid-body downstream optimization. Like the main procedure, this optimization is based on an exhaustive search, yet with a higher resolution. Effects of the post-optimization are not evaluated. Also, its runtime is not explicitly measured. FRED's total runtime for docking a single compound ranges from $1\,s$ to $50\,s$ on a single core of an Intel Xeon with 2.4 GHz, depending on the resolution of exhaustive search and the employed docking function.

In conclusion, FRED employs coarse downstream optimizations and flexibly docks ligands in less than a minute. It would be interesting to analyze the impact of downstream optimizations on FRED's results and whether FRED could benefit from fast all-atom force field post-optimizations

Glide SP/XP. Friesner et al. published Glide SP [**24**] and XP [**52**] that are both centered around the same flexible docking method. Based on a pipeline of hierarchical filters, Glide exhaustively pre-screens the torsion angle space of ligands. Selected ligand poses are subsequently optimized using the OPLS-AA force field [**25**, **26**] as an objective function. Energetic contributions of the receptor are projected onto a grid and all protein atoms are kept rigid during the optimization. Following it, up to six of the lowest-energy poses are further optimized with a Monte Carlo procedure to detect nearby torsional minima. Finally, a consensus scoring scheme ranks the produced poses.

Friesner et al. state that their pre-screens avoid "computationally expensive energy and gradient evaluations" [**24**, p. 1740]. However, explicit runtimes of the optimization procedure are not given. Instead, docking times for 282 complexes of the *Protein Data Bank* (*PDB*) [**53**] are reported. They range from 6 s to 2034 s with an average value of 134 s on an AMD Athlon MP 1800+ CPU. Assuming that the optimizations consume the bulk of the runtime, a faster procedure would improve Glide's usability and cost effectiveness.

GOLD. GOLD [**20**, **21**] is an optimization procedure based on a genetic algorithm for placing ligands in binding pockets with flexible amino acid side chain. For this, a range of scoring functions are available. Among these are GOLDscore [**27**], a molecular mechanics-like fitness function, and Chemscore [**54**]. The latter can be mapped onto grids for representing receptors. GOLD includes an optional simplex [**10**] downstream optimization [**27**]. It refines hydrogen bond donors on the protein side as well as the position, the orientation, and torsions of the ligand. Yet, neither the efficacy nor the runtime of this procedure have been reported.

Lead Finder. Stroganov et al. developed Lead Finder [**55**], a successor to the yet unpublished docking tool Bœuf. Lead Finder relies on a genetic algorithm to optimize ligands in the presence of a rigid receptor. The best ligand poses emerging from this procedure are post-optimized using a force field-based objective function that constitutes a cross-over of the CHARMM19 [**56**] and OPLS-AA [**25**, **26**] force fields. Neither the efficacy nor the runtime of the post-optimization procedure are analyzed.

NeuroDock. Klenner et al. reported the development of NeuroDock [**30**], a docking tool based on the self-organizing algorithm SOCGEN. NeuroDock places a contracted ligand in a user-defined binding pocket. Subsequently, a stochastic procedure guided by predefined optimal bond lengths unfolds this ligand. Finally, NeuroDock applies MOE's gradient-based optimization procedure (see Section 3.9.2) to energetically minimize generated ligands along with hydrogen atom positions in the protein.

The NeuroDock publication neither specifies the efficacy of the optimization procedure nor any runtimes. Thus, comparisons are infeasible.

PhDOCK. Joseph-McCarthy et al. developed PhDOCK [**7**] that represents binding pockets and ligands as sets of pharmacophore points. For rapid docking, sets of mutually similar ligands are subsumed by three-dimensional-pharmacophores. These are subsequently docked into the pre-processed binding pockets. Then, all ligand conformations associated with the docked pharmacophore are scored. For this, grid-based force field potentials corresponding to those in DOCK (see Section 2.2.2) are employed. Optionally, 15 iterations of a simplex-based [**10**] rigid-body minimization post-process ligand poses.

In a validation study, RMSDs of top scored poses are lowered when downstream minimizations are switched on. Joseph-McCarthy et al. thus propose to include these into PhDOCK's standard protocol. This suggests an extremely fast optimization procedure, as one of PhDOCK's main strengths is speed. It takes less than a second to place a single conformation. The data suggests that PhDOCK's docking performance could benefit from replacing the coarse grid potential-based rigid-body minimization with an all-atom method that additionally optimizes dihedral angles. A study by Wu et al. [**57**] on replacing grid-based potentials with an all-atom approach support this view. Section 2.3.3 provides more details on this matter.

RosettaLigand. Meiler et al. and Davis et al. developed RosettaLigand [**36**, **37**], a docking method that supports amino acid side chain and backbone flexibility. It places ligands randomly in protein binding sites. Subsequently, thus obtained initial protein-ligand complexes are optimized in six rounds with a Monte Carlo-based protocol. During this phase, random perturbations alter ligand positions and conformations. These, along with surrounding amino acid side chains, are subsequently energetically optimized. For this, a gradient-based method and softened potential of the Rosetta force field are used. Docking poses resulting from this minimization loop are finally optimized with a gradient-based procedure and hard potentials.

Davis et al. discuss the effects of introducing backbone flexibility in this final optimization step. However, their analysis is based on a small test set of 20 protein-ligand complexes. Furthermore, the statistical mean is employed as a measure although there are outliers. Thus, results are hard to interpret.

In summary, RosettaLigand's algorithm is a two-stage optimization procedure. Generating one pose with it takes approximately 3,060 s, of which 10 s are dedicated to the final optimization. Runtimes for the intermediate optimization steps are not specified. However, it is reasonable to assume that significant parts of RosettaLigand's runtime are dedicated to these local optimizations, since they follow each ligand perturbation. Speeding them up would enhance RosettaLigand's usability and cost effectiveness.

SurFlex. SurFlex, published by Jain [**58**–**60**], makes use of protein conformations from multiple complexes to flexibly dock ligands into binding pockets. Therefore, fragments representing hydrogen bond donors and acceptors, as well as hydrophobic regions, are placed in binding pockets at ideal positions,according to SurFlex's scoring function. Then, ligands are morphologically aligned to these fragments. All degrees of freedom of resulting docking poses are subjected

to a gradient-based optimization procedure. It employs the DREIDING force field [**61**] as an objective function.

In a small-scale experiment involving four protein targets, Jain explored the effects of downstream optimizations. Adding them improves screening performance in two cases. Occasional degradations of results is also described. In contrast, runtimes of the post-optimization procedure have not been systematically analyzed. However, Jain reports an anecdotal case in which adding pre- and post-optimizations for ligands increases the docking procedure's runtime by 36% or 1.1 s per rotatable bond. Consequently, systematic comparisons to Jain's downstream optimizations are not possible due to the lack of data.

2.2.2. Docking tools with intermediate optimizations.

DOCK. Kuntz et al. first published their pioneering DOCK algorithm in 1982 [**62**]. Follow-up versions [**63**–**67**] are still based on the original concept. It represents binding sites by a set of spheres. Onto these, ligand spheres and atoms are matched. The conformational space of the ligand is then sampled via an incremental construction approach. DOCK employs pre-computed grid-based force field potentials calculated with AMBER3 [**68**,**69**] for scoring. Simplex-based [**10**] rigid-body optimizations improve ligand poses during the search algorithm. However, downstream optimizations are not performed. This is due to a study by Gschwend and Kuntz [**70**] who conclude that intermediate optimizations incorporated into the DOCK procedure perform on par with downstream optimizations. These were tested by Meng et al. [**71**] and were found to drive docking poses significantly closer to experimentally determined bioactive structures. However, in both studies, optimizations increase DOCK's runtime by up to one order of magnitude. Consequently, Yang et al. [**72**] measured a runtime of 498 s per docking run with DOCK.

FITTED. Corbeil et al. reported on FITTED [**73**–**75**], a tool for flexible protein-ligand docking. FITTED first generates initial docking poses by randomly placing ligand conformations in binding pockets. At this point, conjugate gradient-based [**8**] optimizations are performed for improving ligand poses. In these, the AMBER3 [**68**, **69**] and GAFF [**76**] force fields are used as objective functions. Corbeil et al. describe their optimizations as "a time-consuming step in the docking process" [**74**, p. 904]. However, exact runtimes are not reported.

Generated initial poses are refined by FITTED with a global optimization based on a genetic algorithm. In each optimization cycle, a user-defined proportion of poses are additionally optimized with the gradient-based method described previously. Again, no runtimes are reported. Also, the efficacy of the single optimization steps is not analyzed.

In summary, Corbeil et al. do not provide a thorough report on their intermediate optimization procedures. Still, there are strong indications that the usability of FITTED would benefit from faster minimization methods.

Glide/IFD. Sherman et al. presented an *induced-fit docking* (*IFD*) procedure [**77**], which builds upon the Glide docking tool (see Section 2.2.1) and a protein refinement module named Prime [**78**–**80**]. IFD first docks ligands into a rigid receptor structure using Glide with a softened

energy function. After scoring, the top 20 complexes are optimized as follows: in a first step, side chains are sampled. Then, a more thorough procedure minimizes thus generated conformations, including ligand and backbone atoms. For this, an adapted version of the OPLS-AA force field [**25**, **26**, **78**] serves as an objective function. Ligands are subsequently redocked into the generated receptor conformations. For this, Glide's standard protocol is applied. It comprises downstream optimizations for ligands.

An analysis on the efficacy of the single optimization steps is not given. Still, Sherman et al. specify that the total average runtime of their procedure amounts to 18,000 s on a 1.6 GHz AMD64 Opteron CPU. For a single complex, protein sampling alone takes 600 s. IFD could thus benefit from a faster optimization procedure.

ICM. ICM [**22**,**23**] docks ligands into binding pockets while considering flexible amino acid side chains. Therefore, it performs global Monte Carlo minimizations with a force field combined of AMBER3 [**68**, **69**] and ECEPP/3 potentials [**81**–**83**] as an objective function. Gradient-based local optimizations follow each perturbation during this Monte Carlo procedure. However, downstream optimizations following the global optimization are not performed. No reports on the runtime or the efficacy of the intermediate optimization steps have been published.

PCRelax. May and Zacharias presented a docking approach whose protein flexibility model is based on normal mode analysis [**84**, **85**]. Their docking protocol initially generates poses with the PCRelax method [**84**]. These are subsequently refined in several optimization steps. For this, the AMBER99 force field [**86**] is used as an objective function for proteins and the GAFF force field [**76**] for ligands. Only non-bonded interactions and dihedral energies are considered and a cutoff of 10 $\AA$ is imposed. The gradient-based L-BFGS method [**43**] is applied. Zacharias and May measured an average runtime of 11 s for a single docking run, including side chain flexibility in the binding pocket.

Optimization steps dominate the runtime of the fully flexible PCRelax approach. Speeding them up would enhance usability cost effectiveness of this docking tool. To tackle this issue, Leis and Zacharias published a grid-based version of their method [**87**]. However, this study neither reports runtimes nor tests on larger data sets.

2.2.3. Summary. As evident in Table 2.1, force field-based optimizations are widely used as an integral part of docking tools. Despite this popularity, the efficacy of these integrated refinements has only been thoroughly demonstrated for Fleksy, PhDOCK, and DOCK. Furthermore, Perola et al. [**19**] analyzed an external downstream optimization scheme for GOLD, Glide, and ICM that improved docking poses produced by GOLD.

Additionally, the effects of downstream optimizations on redocking schemes, which are typically part of docking pipeline, are virtually unknown. To the author's knowledge, so far only Nabuurs et al. have published a thorough analysis on this topic for their tool Fleksy. This work contributes to rectifying the imbalance between the number of tools applying force field-based optimizations and the number of studies analyzing their effects.

TABLE 2.1. Docking and virtual screening tools with intermediate or downstream force field-based optimizations. The total docking runtime is given along with the proportion of it consumed by the optimization procedure. The fourth and the fifth column tell whether the efficacy of the optimization procedure and its impact on rescoring are published.

tool	runtime per pose (s)	runtime of optimization	efficacy tested	rescoring tested	ref.
ADAM	1-115	-	no	no	[**42**]
CRDOCK	10	-	no	no	[**44**]
DARC	~ 3	-	no	no	[**35**]
DOCK	498	up to 90%	yes	no	[**62**–**67**, **70**, **71**]
FITTED	720	-	no	no	[**73**–**75**]
Fleksy	~ 180	$\sim 60\,s$[b]	yes	yes	[**41**]
FLOG	35.66	$15.82\,s$	yes[d]	no	[**50**]
FRED	1-50	-	no	no	[**51**]
Glide/IFD	1,200	$> 600\,s$	no	no	[**77**]
Glide SP/XP	6-2,034	-	no	no	[**24**, **52**]
GOLD	-	-	no	no	[**20**, **21**]
ICM	-	-	no	no	[**22**, **23**]
Lead Finder	60	-	no	no	[**55**]
NeuroDock	-	-	no	no	[**30**]
PCRelax	11	-	no	no	[**84**, **85**]
PhDOCK	< 1	-	yes	no	[**7**]
RosettaLigand	$\sim 3{,}060$	$\sim 10\,s$[a]	no	no	[**36**, **37**]
SurFlex	24.6	$6.6\,s$[c]	yes[d]	no	[**58**–**60**]

[a] not including intermediate optimization steps
[b] "up to a few minutes" [**41**, p. 6511]
[c] for a ligand with six rotatable bonds
[d] anecdotal evidence

More data is available on runtimes of force field-based refinement procedures. The proportion of computation time spent on these ranges from 27% to 90% for the analyzed docking tools. Clearly, optimization procedures thus constitute major runtime bottlenecks. This explains why ultra-fast docking approaches like PhDOCK refine their results with as little as 15 minimization steps. More thorough optimizations would spoil the overall runtime behavior; thus, force field-based optimization procedures should be significantly accelerated. This way they can be routinely applied without impairing interactivity, usability, and cost effectiveness of docking tools.

2.3. Docking Tools Without Additional Force Field-Based Optimizations

A considerable number of tools perform protein-ligand docking without any additional optimizations. This research analyzes the most popular ones and evaluates whether they could benefit from downstream optimizations.

2.3.1. AutoDock. Similar to CDOCKER, AutoDock [**88**–**91**] uses a global search strategy based on a genetic algorithm to find docking poses. Therefore, AutoDock maps energies of its force field [**92**] onto grids. These are pre-computed for binding pockets before actual docking runs start. No post-optimization is performed.

2.3.2. AutoDock vina. AutoDock vina [**93**] optimizes its objective function with a combination of local and global optimization strategies, similar to that of ICM [**22**, **23**]. Based on X-Score [**94**], this procedure takes all atom pairs in binding pockets into account. Downstream optimizations are neither tested nor part of the protocol of AutoDock vina. One docking run takes approximately $8\,s$ on a single CPU core.

2.3.3. CDOCKER. The *CHARMm-based DOCKER* (*CDOCKER*) [**57**, **95**, **96**] docks ligands into rigid binding pockets. Therefore, force grids based on the CHARMm force field [**97**] are constructed. Subsequently, a SA-based algorithm optimizes randomly placed ligand poses.

Wu et al. [**57**] demonstrate that coupling CDOCKER with gradient-based all-atom force field downstream optimizations improves docking accuracy by 10%. Thus, post-optimization can improve pose accuracy even when complex functions are used for docking. At this point, it is important to apply procedures that only negligibly contribute to CDOCKER's main procedure runtime of about $600\,s$ per complex.

2.3.4. Flex-Screen. Based on a stochastic tunneling approach [**98**], Flex-Screen [**99**, **100**] globally minimizes protein-ligand complexes. A force field projected onto a grid serves as an objective function. It is composed of OPLS-AA [**25**, **26**] terms with additional hydrogen bond parameters from AutoDock [**92**]. No post-optimizations are performed.

2.3.5. FlexX. FlexX by Rarey et al. [**48**] decomposes ligands into basic building blocks and places one of these blocks as a base fragment in the protein binding site. Then, incremental steps build up a complete ligand. Schneider et al. [**101**] introduced a post-optimization and rescoring procedure based on HYDE (see Section 3.8.1) and tested it on docking poses produced by FlexX. However, the effects of optimizations are not detailed. Thus, FlexX constitutes a docking tool without downstream optimizations. It processes one compound in less than $10\,s$ [**102**].

2.3.6. FlipDock. Zhao and Sanner [**103**] developed a data structure to store domain-specific flexibility constraints for molecules. Based on this, they perform a constrained genetic algorithm-based optimization of protein-ligand complexes for docking purposes. In this, side chain rotations are performed according to rotamer libraries. For scoring, the AutoDock force field [**90**] is employed. In contrast to AutoDock, energies are computed on an all-atom basis instead of projecting forces onto grids.

Downstream optimizations are not part of the FlipDock algorithm. However, Zhao and Sanner report that their procedure results in unfavorable side chain conformations. They attribute this observation to the coarse sampling implied by using rotamer libraries. That data suggests that downstream optimizations could compensate for this shortcoming.

2.3.7. SLIDE. SLIDE [**104**, **105**] performs triangle-based matching of ligand anchor fragments and binding sites. Subsequently, collisions are removed by translations of anchor fragments. Then, all further ligand atoms are added to the anchor fragment. Finally, dihedral rotations in side chains and ligands resolve clashes. During this procedure, scoring is performed with a simple clash function.

2.3.8. TrixX. TrixX [**102**, **106**] is a docking tool that places pre-generated ligand conformations in protein binding pockets. By employing database-stored and indexed bitmaps for describing small molecules, TrixX screens large conformation sets in sublinear runtime. In a case study by Schlosser et al. [**102**], TrixX processes one conformation in approximately $0.1\,s$. For scoring ligand poses, TrixX utilizes a hierarchical scoring scheme based on the FlexX [**48**] model.

A downstream optimization method combined with TrixX should maintain its favorable runtime behavior and enhance pose quality at the same time. The method presented in this work fulfills these conditions, as demonstrated in the experimental section.

2.3.9. Summary. Docking procedures without additional force field-based optimizations can roughly be divided into two categories. First, there are those that globally search force field energy surfaces that have been projected onto grids (CDOCKER, FlexScreen, AutoDock). Second, there are tools that rapidly produce docking poses using a simple heuristic scoring scheme (AutoDock vina, FlexX, TrixX). According to Wu et al. [**57**], tools of the former category benefit from downstream optimizations with all-atom force fields and data suggests that this is also true for the latter. The experimental section shows that the hypothesis is true for TrixX. However, only extremely fast force field-based minimizations preserve the runtime characteristics of tools of the latter category. Thus, applying one of the previously introduced stand-alone optimization tools is not advisable.

2.4. Side Chain Optimizations and Post-Optimization

In protein engineering scenarios, force field-based energy minimizations are used for refining amino acid side chains of protein binding pockets. The following discusses a few examples.

2.4.1. AS-Dock with downstream minimizations. Miura et al. [**29**] predict site-directed mutagenesis effects on fructosyl amine oxidase. For this, they employ a pipeline composed of MOE's [**28**] docking module AS-Dock, followed by an energy minimization. Miura et al. report the successful adaptation of the active site of fructosyl amine oxidase to fructosyl valine. However, they do not provide a detailed analysis of their *in-silico* prediction procedure.

2.4.2. ICM. Bordner and Abagyan [**107**] introduce point mutations into binding pockets. Their tool, ICM [**22**,**23**], subsequently energetically minimizes amino acid side chain dihedral angles with the ECEPP/3 force field [**81**–**83**] as an objective function. For this, ICM's BPMC [**108**] module performs up to $100,000$ steps. This module couples local optimizations with a Monte Carlo minimization—a procedure that is comparable to an extensive post-optimization. In total, ICM consumes $720\,s$ on a 1.3 GHz AMD Athlon CPU in this scenario.

2.4.3. TINKER for protein engineering. Zhang et al. [**109**] perform force field-based energy minimizations on wild type and mutated protein binding sites using the tool TINKER [**110**]. The free energy change due to mutations is then computed with the resulting structures. Details on the minimization procedure, including runtimes, are not reported.

2.4.4. Summary. Examples of force field-based downstream optimizations used in protein engineering scenarios were presented. All reported runtimes indicate that faster minimizations would boost the usability of the introduced tools. At the same time, a more thorough assessment of the efficacy of the employed methods would be valuable.

2.5. Conformation Generators with Post-Optimizations

A large number of conformation generation methods have been published thus far. Among them are systematic search algorithms, knowledge-based approaches, random searches, distance geometry-based procedures, and global minimizations using Monte Carlo methods or genetic algorithms. Consecutive force field-based minimizations—some supplemented with constraints on dihedral angle rotations—constitute a widely applied method for downstream refinements. The following sections introduce three conformation generators, two popular ones and one less known tool, all of which employ force field-based post-optimizations. Further, this work analyzes the runtime of this process and whether it allows for constraints on dihedral angle rotations. For a more comprehensive overview on conformation generation tools and methods, refer to Schärfer's dissertation [**6**].

2.5.1. Catalyst. Using the CHARMm force field [**97**] Accelrys's Catalyst [**111**] generates conformations with two selectable levels of precision. Downstream optimizations are also performed with CHARMm. This tool supports harmonic constraints around a given value for dihedral angles. To the author's knowledge, further details on CHARMm's optimization procedure are not published.

2.5.2. ConfGen. ConfGen [**112**] bases its knowledge-based conformation generation on dihedral potentials of the OPLS_2001 force field [**25**, **113**]. Optionally, gradient-based minimizations using the OPLS_2005 force field [**114**] refine generated structures. No constraints are imposed during this procedure. Depending on the selected level of precision, minimizations slow down the conformation generation by factors of 3.8 to 12.9. Processing times for a single ligand range from 4.2 s to 45.8 s on a single core of an Intel Core2 Quad CPU Q6600 at 2.40 GHz.

2.5.3. DG-AMMOS. Lagorce et al. reported the development of DG-AMMOS [**115**], a conformation generator that makes use of distance geometries. This tool corrects generated structures by applying unconstrained conjugate gradient optimizations [**8**] with the AMMP force field sp4 [**116**]. Lagorce et al. report an average runtime of 0.61 s per compound on two 3.0 GHz cores of an Intel Xeon CPU. The proportional runtime of the minimization procedure is not specified.

TABLE 2.2. GPU-accelerated force field kernels along with reported speedups (S_{GPU}) and the number of particles in tested systems. Test machine setups are listed in Table 2.3 and referenced in the last column.

authors	year	kernel type	#particles	S_{GPU}	setup
Stone et al. [**118**]	2007	Coulomb	not specified	40	1
Anderson et al. [**119**]	2008	Lennard-Jones	up to 125,000	5-60	2
van Meel et al. [**120**]	2008	Lennard-Jones	200-70,000	2-40	3
Liu et al. [**121**]	2008	Lennard-Jones	8,192-131,072	11-20	4
Roh et al. [**122**]	2009	all non-bonded	3,204-13,296	33-287	5
Friedrichs et al. [**123**]	2009	full force field	544-5,078	128-735	6
Schmid et al. [**124**]	2010	solvent interactions	5,411-75,129	25-54	7
Eastman and Pande [**125**]	2010	all non-bonded	582-78,207	19-59	8
Ruymgaart et al. [**126**]	2011	all non-bonded	23,536	10	9
Ruymgaart and Elber [**127**]	2012	water interactions	21,036-92,328	37-49	10
Götz et al. [**128**]	2012	full force field	304-25,095	23-203[a]	11
Anthopoulos et al. [**129**]	2013	full force field	2,103-61,418	-	12

[a] extrapolated from runtimes given for two hex core CPUs

2.5.4. Summary. Conformation generators produce small molecule structures; some generators refine these structures with force field-based downstream optimizations. This process should be efficient and thus, not overly enhance the runtime of the core tool, which seems largely to be the case for the analyzed tools.

As Hawkins et al. [**117**] claim, force field-based minimization may bias molecules towards their gas phase conformations. Supplementing force field potentials with knowledge-based constraints derived from co-crystallized conformations can counteract this effect. The torsion libraries developed by Schärfer [**6**] provide such a set of constraints. One of the main advantages lies in the transparent association of angle constraints to specific torsion patterns. To the author's knowledge, no post-optimization method makes use of constraints with such a high degree of specificity.

2.6. Force Field Kernels for GPU

As listed in Table 2.2, various approaches that port force field kernels and MD simulations to GPU have been published in recent years. For assessing their performance, Section 3.1 introduces the *economic efficiency* (E_{eco}) and *speedup* (S_{GPU}) measures. Judging from specified S_{GPU} values, some algorithms Table 2.2 presents are quite efficient. The following sections discuss approaches that include interesting algorithmic aspects. Furthermore, this section analyzes whether they are targeted at scoring protein binding sites in energy minimization scenarios relevant to this work. In particular, a method has to be efficient for molecular systems comprising less than 1,500 atoms. While minimizing their energies, regions with constant interaction forces should be excludable from energy calculations.

TABLE 2.3. Graphics processors used for benchmarking force field kernels listed in Table 2.2.

ID	CPU (single core)	NVIDIA GPU
1	Intel Core 2 Extreme QX6700	GeForce GTX8800
2	Intel Xeon 80546K 3.0 GHz	GeForce GTX8800
3	Intel Xeon 3.2 GHz	GeForce GTX8800
4	Intel Pentium IV 3.0 GHz	GeForce GTX8800
5	Intel Core2 2.93 GHz	GeForce GT8600
6	Intel Xeon 2.66 GHz	GeForce GTX280
7	AMD Athlon 64 X2 3.2 GHz	Quadro FX5800
8	Intel Core 2 Duo 3.0 GHz	GeForce GTX280
9	AMD Phenom IIX4 965 3.4 GHz	GeForce GTX480
10	AMD Phenom 965	GeForce GTX480
11	Intel Xeon X5670 2.93 GHz[a]	GeForce GTX580[b]
12	Intel Xeon E5335	GeForce GTX680

[a] all six cores used in experiments
[b] further experiments with a Tesla C1060

2.6.1. Early full force field portation. Friedrichs et al. [**123**] were among the first to publish a full force field portation for GPUs. According to their approach, all pairs of non-bonded interactions are projected onto a force matrix. Due to its symmetry, only forces above or along its diagonal must be calculated. This part of the matrix is thus split into quadratic tiles whose dimensions reflect the warp size of NVIDIA GPUs (see Section 1.5). Friedrichs et al. designed two specific looping schemes for tiles above and along the diagonal. Basically, a thread loads data of one atom and then loops over all other atoms within the processed tile in both schemes. Interaction pairs to be excluded are identified by an additional matrix. In contrast, neither cutoffs nor selectable rigid regions are supported. This method is therefore not suited for fast optimizations of protein binding sites.

Friedrichs et al. report speedup values ranging from 128 to 735. These values are quite high, as the employed NVIDIA GeForce GTX280 GPU has only 240 cores [**130**]. Still, no explanation for these measurements is provided.

2.6.2. Efficient cutoff mechanism. Eastman and Pande [**125**] proposed a cutoff mechanism for non-bonded interaction kernels. Their algorithm uses the approach of Friedrichs et al. as a starting point. As an extension, bounding boxes are computed for groups of 32 atoms belonging to one tile of the force matrix. For each atom, distances to all other bounding boxes are computed. Based on these, the non-bonded kernel decides whether interactions are calculated. This mechanism results in divergent execution paths and the bounding boxes have to be refreshed once every several time steps. Then again, the total number of non-bonded interaction computations is reduced to $O(n)$. All in all, Eastman and Pande report that measured speedups range from 19 to 59. Furthermore, they state that their cutoff implementation is beneficial for

systems larger than 1,500 atoms. However, the application scenario relevant in this work is centered around systems with less than 1,500 atoms.

2.6.3. State-of-the-art MD package. Götz et al. [**128**] reported the portation of MD simulations with the AMBER force field [**131**] to GPU. Their non-bonded interaction kernels are based on the work of Friedrichs et al. In contrast to them, Götz et al. handle the exclusion of atom pairs by compiling a list of 1-4 interactions (see Section 3.4.1), which is processed in the kernel for bonded interactions.

Runtime and scaling behavior of the GPU-based AMBER algorithm are thoroughly analyzed. Economic efficiencies of 1.48 to 13.14 and speedups of 23 to 203 are computed based on published figures. These calculations refer to measurements with an Intel Xeon X5670 hex core CPU clocked at 2.93 GHz and a NVIDIA GeForce GTX580 GPU on system sizes of 304, 1,231, and 2,492 atoms. Götz et al. state that their algorithm performs best for systems with at least 2,500 atoms. The computed performance figures support this statement.

2.6.4. New approach to handling irregular interactions. Anthopoulos et al. [**129**] presented a full portation of the MMFF94s force field (see Section 3.4.3) to GPU. Their kernels are explicitly optimized for the recent NVIDIA Kepler architecture. Their cutoff mechanism uses a fine-grained cell-list approach. For excluding 1-2 and 1-3 interactions (see Section 3.4.1), atoms are reordered to ascertain that bonded atoms are close to each other. Then, an integer encoding the relative indices of neighboring atoms is assigned to each atom. With this, kernels check a small number of atom pairs for exclusion criteria. Furthermore, an approach for handling 1-4 interactions and all bonded forces in one kernel is presented. It decomposes molecules into pairs of bonded atoms. A data structure associates these with all further atoms that are bound to them. This way, kernels have full information on the molecular graph around given atom pairs. Furthermore, Anthopoulos et al. handle hydrogen bonds by flagging each atom either as an acceptor, a donor, or neither. This permits kernels to switch at runtime to alternative potentials for hydrogen bonds. At the same time, this approach entails divergence of execution paths.

Using their force field algorithm, Anthopoulos et al. perform gradient-based minimizations of molecular systems with approximately 200 to 60,000 atoms and a cutoff of 10.25 $\AA$. For 1,000 minimization steps, runtimes between approximately 0.8 s and 8 s are reported on a NVIDIA GeForce GTX680 GPU. For high throughput and interactive scenarios, these runtimes are insufficient. The data suggests that the approach for excluding close non-bonded interactions is too tedious. It entails loading an additional integer value for each atom and distance checks during force computations. Furthermore, Anthopoulos et al. point out that their cell-based approach for cutoff exclusions is inefficient for systems with less than 7,000 atoms on GPU.

2.6.5. Summary. Over the last years, considerable effort has been devoted to porting force field kernels and MD simulations to GPU. Friedrichs et al. [**123**] published a fundamental algorithm for distributing non-bonded force calculations. Eastman and Pande [**125**], as well as

Anthopoulos et al. [**129**], introduced innovative cutoff schemes relying on cell-lists to enhance computational performance. Furthermore, various ways of excluding undesired non-bonded interactions have been proposed by Friedrichs et al. [**123**], Götz et al. [**128**], and Anthopoulos et al. [**129**]. However, research has been centered around MD simulations for systems comprised of several thousand of atoms. Consequently, the proposed techniques are often inefficient for smaller systems. For example, Eastman and Pande state that "the break-even point at which a cutoff becomes beneficial appears to be $\sim$ 1,500 atoms" [**125**, p. 1271]. Furthermore, Götz et al. find that "the real advantage of the...GPU implementation becomes apparent for...systems with 2,500 to 25,000 atoms" [**128**, p. 1550]. In fact, only Anthopoulos et al. test their approach for minimizing ligands in the presence of a receptor, a scenario corresponding to optimizations in drug discovery pipelines. However, their algorithm is optimized for systems with more than 7,000 atoms. Like all others, it fails to ignore non-bonded interactions that are irrelevant to coarse binding site energy minimizations. Doing so would speed up computations significantly.

Thus far, this research concludes that the potential of GPUs for minimizing smaller systems comprising less than 1,500 atoms has not been exploited. This is, nonetheless, a prerequisite for efficiently employing GPU-accelerated minimizations in drug development scenarios introduced in Section 1.1.

CHAPTER 3

Pre-Existing Methods

3.1. Measuring Implementation Performance

The speedup is a widely used measure for comparing sequential to parallel implementations:

$$S_p = \frac{T_1}{T_p}$$

Here, p is the number of processor cores used to execute the parallel implementation and T_p is its runtime. The execution time of the sequential implementation (T_1) is benchmarked on a single processor core. In accordance with *Gustafson's law* [**132**], S_p depends on the size of the analyzed problem. Larger problem sizes are typically linked to higher speedups. As Gustafson points out, this is due to the diminishing proportion of program parts that are computed sequentially. These parts, in turn, limit the maximum attainable speedup, according to *Amdahl's law* [**133**].

3.1.1. Comparing GPU- to CPU-based implementations. When comparing CPU- to GPU-based implementations, a similar measure is commonly utilized. In this scenario, speedup refers to

$$S_{GPU} = \frac{T_1}{T_{GPU}}$$

where T_1 is the execution time of the CPU implementation on a single core, while T_{GPU} is the execution time of the GPU implementation on a graphics processor. In this work, the term *speedup* refers to S_{GPU}.

S_{GPU} typically depends on the size of a problem, just like its counterpart S_p. However, while implementations are compared on the same processor for computing S_p, the value of S_{GPU} is additionally influenced by the choice of GPU and CPU type. Hence, benchmarking on processors of the same generation and performance segment is a prerequisite for fair comparisons, as Lee et al. [**134**] pointed out. Therefore, this research compares implementations exclusively on high-end processors of the same generation. In detail, test machine I (see Section 5.4) is equipped with Intel Xeon E5420 CPUs and a NVIDIA Tesla C1060 GPU. Both processor types were introduced to the market from 2007 to 2008 and target the high-performance segment. Further comparisons were conducted on the Intel Xeon E5-2680 CPU of test machine III and the NVIDIA GeForce GTX 680 of test machine II. Both high-performance processors were launched in early 2012.

Lee et al. [**134**] argue that for a valid comparison, both GPU and CPU implementation must be optimized for their respective platforms to exploit the full potential of the respective

architectural features. Therefore, the author performed code optimizations for the CPU and GPU implementations to the best of his knowledge.

As already discussed in Section 1.5, data transfer rates between host systems and GPUs are comparatively low. For this reason, Gregg and Hazelwood [**135**] suggest that cross-platform implementation comparisons must include all necessary data transfer processes. The author supports this notion and excludes only molecular initialization procedures from this work's measurements. Data transfer during optimizations and energy evaluations is captured by benchmarks.

3.1.2. Economic efficiency of implementations. The speedup constitutes a fairly good measure to compare execution times of implementations. However, this measure is only weakly linked to practical implications. Therefore, this work captures parts of the economic dimension of implementations, and for this, introduces the economic efficiency (E_{eco}), which is based on a *performance per Watt* (*perf/W*) estimate introduced by Woo and Lee [**136**] for multi-core systems. This measure relates the speedup to the *thermal design power (TDP)* of a processor. The TDP is a upper limit to the power that a processor draws when executing performance-demanding applications.

For deriving an upper limit to $perf/W$ for a CPU with n cores, this work assumes a best-case scenario, in which implementations scale perfectly and all cores run at maximum performance. Thus, the speedup S_p is at its theoretical maximum, namely p. With this, the performance per Watt boils down to

$$\frac{perf}{W}(CPU) = \frac{S_p}{TDP} = \frac{p}{TDP}$$

For GPUs, the theoretical optimal value S_p is substituted for experimentally determined values of S_{GPU}:

$$\frac{perf}{W}(GPU) = \frac{S_{GPU}}{TDP}$$

These two measures are set into proportion to compute the economic efficiency:

$$E_{eco} = \frac{\frac{perf}{W}(GPU)}{\frac{perf}{W}(CPU)}$$

As more efficient processors are developed with each new generation, E_{eco} depends on the compared platforms. However, in contrast to S_{GPU}, this measure provides an estimate by which factor computations are cheaper. Furthermore, it constitutes a lower bound, as a best-case scenario is assumed for $\frac{perf}{W}(CPU)$.

3.1.3. Interactivity. Measuring economic efficiency is relevant when judging the applicability of programs in large-scale usage scenarios. However, a more important factor in a typical workstation-usage scenario is response times. As they decrease, user productivity increases substantially [**137**]. In practice, lowering response times of $3.0\,s$ by as little as $2.5\,s$ enhances productivity [**138**]. Furthermore, delays of more than $15\,s$ [**139**] or even as little as $1\,s$ [**140**] can be demoralizing. As a rule of thumb, the user's flow of thought is interrupted when waiting more

than $1\,s$ for computations to finish [**139**–**141**] . Therefore, a process is defined as *interactive* if it returns its results in less than $1\ s$.

To compare the interactivity of programs executed on GPU and CPU, this work assumes that workstations are equipped with quad-core processors and that CPU-based implementations scale perfectly to all cores. Consequently, measured single core runtimes are divided by four for judging interactivity. Runtimes measured for GPU implementations are not modified.

3.1.4. Comparing GPU-based implementations. Typically, publications on GPU implementations employ S_{GPU} to measuring speedups. As discussed previously, S_{GPU} depends on problem sizes and utilized hardware, which is taken into account when conducting S_{GPU}-based speedup comparisons. Furthermore, speedup values reported for isolated kernels can be orders of magnitude higher than those attainable for complete algorithms. This is partially due to Amdahl's law. Thus, this research only compares similar parts of implementations.

3.2. Measuring Distances

The *root-mean-square deviation (RMSD)* measure is computed to average distances between atoms of molecules:

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\delta_i^2}$$

where N is the number of atom pairs to compare and δ_i is the distance between two equivalent atoms. This research expresses all atomic distances and the RMSD in ångström ($\mathring{A}$), where $1\,\mathring{A} = 0.1\,nm$ [**142**].

The following employs the terms *correct* and *excellent ligand pose*, which refer to poses with an RMSD below $2.0\,\mathring{A}$ and $1.0\,\mathring{A}$ to the ligand's crystal structure, respectively. Furthermore, the term *clash* refers to two atoms i and j whose overlap is greater than zero:

$$overlap_{ij} = r_{VDW_i} + r_{VDW_j} - dist_{ij} - 0.2\mathring{A}$$

Here, r_{VDW_i} and r_{VDW_j} refer to the van der Waals radius of i and j, respectively, while $dist_{ij}$ is the interatomic distance between i and j. Hydrogen bonds are excluded from this rule.

3.3. Algorithms

3.3.1. Simulated annealing. Simulated annealing (SA) [**143**, **144**] is a modification of the Metropolis-Hastings algorithm [**145**], which is a Monte Carlo method. It thus constitutes a stochastic, non-deterministic, heuristic optimization procedure that approximates the global minimum of an objective function. Depending on parameterizations, its tendency to yield local minima varies.

The convergence behavior of SA procedures has been analyzed extensively in numerous publications (see e.g., [**146**–**158**]). In summary, these indicate that choosing a *cooling schedule* appropriate for the targeted objective function is vital for obtaining the global minimum with SA.

Cooling schedules govern the SA process, which encompasses three key steps. As laid out in Algorithm 3.1, these steps are generating a new state, evaluating it with an objective function, and finally accepting or rejecting it. The result of the latter step is probabilistically determined, typically by the Metropolis Criterion **[145]**:

$$\Big(\Delta E < 0\Big) \vee \Big(e^{-\Delta E/T} > Rand[0,1]\Big)$$

It thus depends on the difference between objective function values for the new and the previous state (ΔE), the *temperature* parameter T (named *currTemp* in Algorithm 3.1), and a sample drawn from a uniform distribution on the closed interval $[0, 1]$. Setting T to zero yields an optimization procedure that solely accepts states that entail lower objective function values.

Algorithm 3.1 Basic simulated annealing

```
SimulatedAnnealing
 1  lastScore ← scoreCurrentState()
 2  bestScore ← lastScore
 3  currTemp ← getInitialTempFromCoolSchedule()
 4  while !isCoolScheduleAbort(currTemp) and !isEarlyAbort(currScore)
 5        do currTemp ← updateTempAccordingToCoolSchedule()
 6           generateNextState()
 7           currScore ← scoreCurrentState()
 8           if !isAccepted(currScore, lastScore, currTemp)
 9              then restorePreviousState()
10                   continue
11           lastScore ← currScore
12           if isBetterThan(currScore, bestScore)
13              then bestScore ← currScore
14                   saveCurrentStateAsBest()
15  restoreBestState()
16  return scoreCurrentState()
```

During the SA loop, the temperature parameter T is adapted by cooling schedules, of which a large variety has been developed. Nourani and Andresen **[159]** provide an overview. Cooling schedules are categorized into two classes: static schedules typically reduce T, while adaptive schedules adjust T to the state of the system that is subject to the optimization. Section 4.6 introduces self-developed adaptive schedules. The previously published exponential cooling scheme **[143]** is also employed in this work.

3.3.2. Reductions on GPU. Summations in parallel are termed reductions. To efficiently sum up the elements of an array A on GPU, A is split into subarrays $A_0 \dots A_N$ of equal size. These subarrays are subsequently assigned to GPU multiprocessors that reduce them to a single value. At this point, synchronizing multiprocessors would allow continuing the summation process. However, NVIDIA GPUs do not permit this operation. Therefore, standard algorithms

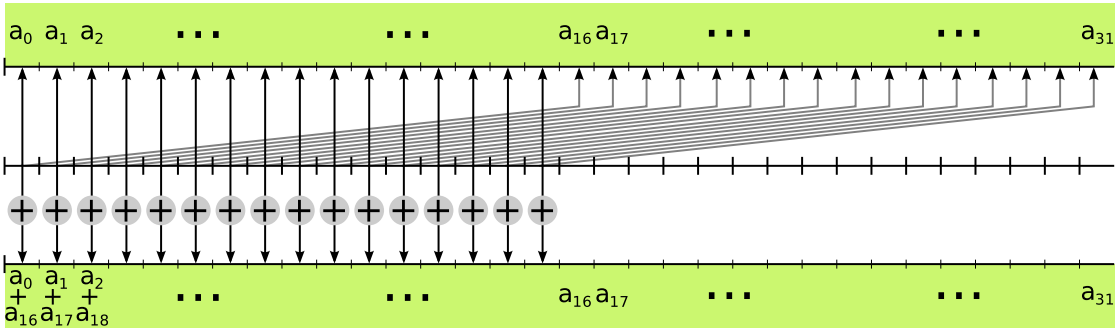


FIGURE 3.1. Reduction with sequential addressing. On-chip shared memory stores array elements $a_0 \dots a_{31}$.To sum them up, threads access a contiguous memory region that spans all shared memory banks. Consequently, each memory bank is accessed once and conflicts are avoided.

emulate synchronization by decomposing reductions into multiple kernel calls. This corresponds to hierarchically merging the sums subarrays $A_0 \dots A_N$ in a tree-like manner.

The same approach is used to reduce a single subarray A_i on a multiprocessor. Efficient implementations accumulate A_i's values with *sequential addressing*. This technique executes the loop specified in Algorithm 3.2 in parallel and thereby reduces A_i to half its size. The sequential

Algorithm 3.2 Basic parallel reduction on GPU

```
1  for i = 0 to SIZE(array)/2
2        do array[i] ← array[i] + array[2 · i]
```

addressing approach avoids bank conflicts, as illustrated in Figure 3.1.

Reduction kernels execute the loop specified in Algorithm 3.2 until only a single resulting value remains. Unrolling the thus introduced outer loop reduces instruction overhead. This entails synchronizing multiprocessor cores after every execution of the inner loop. It is preferable to avoid this time consuming step. Therefore, an approach by Harris [**160**] takes standard warp sizes of NVIDIA GPUs into account. For an in-depth discussion of efficient reduction techniques, please refer to Harris' slides.

3.3.3. Kahan's numerical stabilization for summations. The *Kahan algorithm* [**161**] reduces the numerical error coupled to floating-point summations. Typically, adding two binary represented floating-point numbers a and b leads to the absorption effect. Its magnitude depends on the employed floating-point precision and the difference in magnitude between a and b.

Kahan's algorithm reduces absorption effects by introducing a compensation variable that keeps track of the accumulated error. In Algorithm 3.3, this compensation variable is c, which stores proportions lost due to absorption of the current addend.

Algorithm 3.3 Kahan's summation algorithm

```
KahanSum(array)
1  sum ← 0.0
2  c ← 0.0
3  for i = 0 to len(array) − 1
4        do y ← array[i] − c
5           t ← sum + y
6           c ← (t − sum) − y
7           sum ← t
8  return sum
```

3.3.4. Trilinear interpolation. As depicted in Figure 3.2, trilinear interpolation [**162**] aims to approximate a function value $f(p)$ for point $p = (x_p, y_p, z_p)$ by computing a distance weighted average of function values assigned to the corner points of a $2 \times 2 \times 2$ grid cell.

For this, bilinear interpolations for the x- and y-dimensions of p with respect to the lower and upper grid square are computed in the first step. Therefore, scaling factors d_x and d_y are computed:

$$\begin{aligned} d_x &= (x_p - x_0)/(x_1 - x_0) \\ d_y &= (y_p - y_0)/(y_1 - y_0) \end{aligned}$$

Interpolated values are computed using these factors for projections of p onto the lower [$p_{lower} = (x_p, y_p, z_0)$] and the upper grid square [$p_{upper} = (x_p, y_p, z_1)$] where

$$\begin{aligned} f(p_{lower}) &= (1 - d_y)(1 - d_x) \cdot f((x_0, y_0, z_0)) + d_y(1 - d_x) \cdot f((x_0, y_1, z_0)) \\ &\quad + d_y d_x \cdot f((x_1, y_1, z_0)) + (1 - d_y) d_x \cdot f((x_1, y_0, z_0)) \end{aligned}$$

and analogously

$$\begin{aligned} f(p_{upper}) &= (1 - d_y)(1 - d_x) \cdot f((x_0, y_0, z_1)) + d_y(1 - d_x) \cdot f((x_0, y_1, z_1)) \\ &\quad + d_y d_x \cdot f((x_1, y_1, z_1)) + (1 - d_y) d_x \cdot f((x_1, y_0, z_1)) \end{aligned}$$

A linear interpolation of p_{lower} and p_{upper} with respect to the z-coordinate of p is performed in the last step.

3.4. Molecular Mechanics Force Fields

This work utilizes the molecular mechanics force fields SuperTrAmber [**163**], MMFF94s [**164**], *AMBER94* [**131**], and *AMBER03* [**165**] as objective functions for optimizations. The commercially available tools MOE and YASARA (see Sections 3.9.2 and 3.9.1) are applied for optimizations and include implementations of the two variants of AMBER. MMFF94s and SuperTrAmber are implemented in the force field framework (see Section 4.2). The author selected them for this work as they are both targeted at energy minimizations for protein-ligand complexes. Furthermore, the parameters of MMFF94s are published and a large molecular library

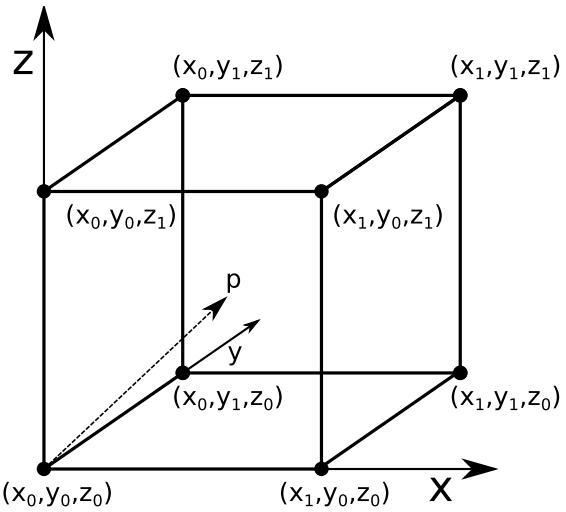


FIGURE 3.2. A $2 \times 2 \times 2$ cubic grid cell with a point p for which an interpolated functional value derived from those of the cell's corner points is computed.

for testing implementations is available. Also, the author of SuperTrAmber provided all parameters and a reference `FORTRAN` implementation of the force field, which facilitated testing the implementation.

3.4.1. SuperTrAmber. The yet unpublished force field SuperTrAmber [**163**] was developed for optimizing protein-ligand complexes. It combines the AMBER3 [**68**, **69**] and the *Tripos force field* (*TAFF*) [**166**] and therefore handles molecules that are assigned to either AMBER3 or TAFF atom types (=SYBYL atom types). Both assignments may be mixed in the same system.

For molecules that are within the scope of application of AMBER3, SuperTrAmber produces the same energies as the original AMBER3. TAFF force field parameters were replaced by AMBER3 parameters by mapping AMBER3 to SYBYL atom types whenever possible. In all other cases, TAFF parameter sets are unmodified. Hence, SuperTrAmber allows for easy parameterization and energy calculations for both proteins specified in pdb format [**167**] and small ligands specified in Tripos MOL2 format [**168**].

Although SuperTrAmber is based on AMBER3, of which several successors have been developed, there is good reason to assume that SuperTrAmber still performs well. Comparative studies benchmarking AMBER3 [**169**, **170**] support this view.

This work utilizes SuperTrAmber's non-bonded van der Waals and electrostatic energy terms, as well as its hydrogen bond and torsional energy terms. The van der Waals energy term includes the attractive dipole-dipole, dipole-induced dipole, and dispersion forces, as well as a repulsive component due to the *Pauli exclusion principle*. All these forces are approximated by a 12-6-Lennard-Jones potential:

$$E_{vdW} = \frac{\varepsilon_{ij} \cdot R_{ij}^{*12}}{R_{ij}^{12}} - \frac{2 \cdot \varepsilon_{ij} \cdot R_{ij}^{*6}}{R_{ij}^{6}}$$

where ε_{ij} is the specific well-depth, R^*_{ij} is the reference distance between atoms i and j, and R_{ij} is the actual distance between atoms i and j. Electrostatic contributions are calculated using the Coulomb potential:

$$E_{electrostatic} = \frac{332.05222 \cdot q_i \cdot q_j}{D \cdot R_{ij}}$$

where q_i and q_j are partial charges and R_{ij} is the distance between atoms i and j while D is the dielectric constant, which set to 4.0.

Atomic partial charges q_i and q_j for small molecules are user-defined. They can, for example, be computed using the *Gasteiger-Marsili method* [**171**] (see Section 3.6.1). Protein atoms are assigned AMBER3 charges [**68**] (see Section 3.6.3). These are selected from a table according to atom and residue (amino acid) types.

Atom pairs that can form hydrogen bonds are predefined and identified by their atom type. They are treated separately, as done in AMBER3. Instead of a 12-6-potential, a 12-10-Lennard-Jones potential is applied to hydrogen bond pairs:

$$E_{Hbond/Coord} = \frac{5 \cdot \varepsilon_{ij} \cdot R^{*12}_{ij}}{R^{12}_{ij}} - \frac{6 \cdot \varepsilon_{ij} \cdot R^{*10}_{ij}}{R^{10}_{ij}}$$

Atom pairs forming coordination bonds, e.g., between a metal cation and a coordinating counter atom, can be defined and treated the same as hydrogen bond pairs in SuperTrAmber. Therefore, specifically adjusted parameters are employed for the well-depth ε_{ij} and the reference distance R^*_{ij}. This makes it possible to assign special geometries to metal complexes without encountering the problems that arise using models with covalent bonds.

Atom pairs separated by one bond (*1-2 interactions*) or two bonds (*1-3 interactions*) are excluded from non-bonded van der Waals, H-bond, and Coulomb energy calculations. Non-bonded *1-4 interaction* contributions are scaled by a factor of 0.5.

Finally, torsional energies are computed for atom quadruples (four consecutive atoms in the molecular graph) using a cosine potential:

$$E_{torsion} = \sum_m 0.5 \cdot V^{ijkl}_m \cdot (1 + cos(n^{ijkl}_m \cdot \Phi_{ijkl} - \gamma^{ijkl}_m))$$

where V^{ijkl}_m is an atom type-dependent force constant, n^{ijkl}_m is the periodicity of the cosine potential, Φ_{ijkl} is the dihedral angle between atoms i, j, k, l, and γ^{ijkl}_m is the phase shift of the cosine potential.

The full SuperTrAmber force field also comprises the harmonic bond stretch and angle bending energy terms, as well as the improper dihedral term as applied in AMBER3 [**68**]. Additionally, the TAFF [**166**] out-of-plane term is integrated in SuperTrAmber. Because the intramolecular degrees of freedom in the presented optimization procedure are restricted to torsions, i.e., rotations about rotatable bonds, rotations, and translations of whole molecules, these terms are not relevant to optimization procedures described in this work.

3.4.2. SuperTrAmber with emphasized hydrogen bonds. A modified version of SuperTrAmber emphasizes the hydrogen bond potential. For this, the well-depth ϵ_{ij} for atom pairs forming hydrogen bonds is multiplied by a factor of eight. This results in local minima in accordance with the HYDE function (see Section 3.8.1). Interestingly, the scoring function of AutoDock **[92]**, which employs a force field-based scoring function, uses the same approach for modeling hydrogen bonds.

3.4.3. MMFF94s. The *Merck Molecular Force Field* (*MMFF94*) was developed by Halgren and became fully accessible through a series of publications **[172–176]** in 1996. The force field was designed to model protein-ligand interactions but also receptors and ligands in isolated form. To achieve this goal, Halgren constructed a large training set consisting mostly of *ab initio* data obtained from quantum mechanical calculations, but also of experimentally derived structures.

MMFF94s **[164]** is a variant of MMFF94 specifically designed for energy minimizations. These two force fields mainly differ when evaluating amides and unsaturated amines. For these moieties, MMFF94s is trained to yield nearly planar geometries at the delocalized trigonal nitrogen center, which accords with the expectations of medicinal chemists.

MMFF94s and MMFF94 share the same terms for computing the potential energy of molecular systems:

$$\begin{aligned} E_{total} \quad = \quad & \sum E_{bond_{ij}} + \sum E_{angle_{ij}} + \sum E_{stretch\text{-}bend_{ijk}} \\ & + \sum E_{out\text{-}of\text{-}plane_{ijk;l}} + \sum E_{torsion_{ijkl}} \\ & + \sum E_{van\text{-}der\text{-}Waals_{ij}} + \sum E_{electrostatic_{ij}} \end{aligned}$$

For optimizations carried out in this work, only van der Waals, electrostatic, and torsion energies are relevant. The term modeling the latter energy type for a torsion angle Φ formed by atoms i, j, k, and l is:

$$E_{torsion_{ijkl}} = 0.5\{V_1[1 + \cos(\Phi)] + V_2[1 - \cos(2\Phi)] + V_3[1 - \cos(3\Phi)]\}$$

where V_1, V_2, and V_3 are constants that depend on the types of the atoms enclosing the torsion angle.

The van der Waals energy of an atom pair (i, j) is approximated by a buffered 14-7 form of the Lennard-Jones potential:

$$E_{van\text{-}der\text{-}Waals_{ij}} = \varepsilon_{IJ} \left(\frac{1.07 \cdot R^*_{IJ}}{R_{ij} + 0.07 \cdot R^*_{ij}} \right)^7 \left(\frac{1.12 \cdot R^{*7}_{IJ}}{R^7_{ij} + 0.12 \cdot R^{*7}_{ij}} - 2 \right)$$

where R_{ij} is the distance between atoms i and j. Depending on the atom types I and J to which atoms i and j correspond, the parameters ε_{IJ} and R^*_{IJ} determine the well-depth in $kcal/mol$ and the minimum energy separation (and thus the location of the potential's minimum) in ångström.

The electrostatic energy is modeled by a slightly modified Coulomb potential:

$$E_{electrostatic_{ij}} = \frac{332.0716 \cdot q_i \cdot q_j}{D(R_{ij} + \delta)^n}$$

where q_i and q_j are the partial charges on atoms i and j, R_{ij} is the interatomic distance in ångstrom, D is the dielectric constant set to 4.0, and $\delta = 0.05\,\mathring{A}$ is the electrostatic buffering constant.

The MMFF94s force field is completed by an atom and a partial charge model, which are introduced in Sections 3.5.3 and 3.6.2.

3.4.4. AMBER94. In 1994, Cornell et al. published AMBER94 [**131**] as a successor to the force field published by Weiner et al. [**68**, **69**] a decade earlier. The growth of computer power permitted explicit solvent representations. In the presence of these, systems were modeled to derive parameters of AMBER94. This resulted in a complete updating of the partial charge model that, in turn, allowed for dropping the explicit 12-10 Lennard-Jones potential modeling hydrogen bonds. Furthermore, new van der Waals atomic parameters were derived from liquid simulations. All other energy terms were retained from Weiner et al. The same applies to most term parameters, for which only minor extensions were added.

3.4.5. AMBER03. AMBER03 [**165**], which was published by Duan et al. in 2003, retains the potential functions of AMBER94 while atomic point charges are derived from condensed phase quantum mechanical simulations of small peptides. According to Duan et al., this allows for more accurate simulations of protein dynamics in solution. Furthermore, AMBER03 comprises more precise torsion parameters for protein backbone Φ and Ψ torsion angles.

3.5. Atom Type Models

Atom type models permit categorizing atoms according to their molecular environments and their hybridization state. This approach allows deriving force field parameters from a molecular moiety and applying them in a similar one. Atom type are thus a means of reducing complexity and making chemical information available for computational processing.

3.5.1. AMBER atom type model. From a quantum mechanical view point, the electron configuration suffices as a description of atoms. Atom types are therefore only based on this property in AMBER3 [**68**]. However, subsuming different electron configuration under a single atom type entails compromises between obtaining the most accurate representation and having a manageable number of types. Counting only the types relevant in this work, namely those representing single atoms, the AMBER3 model consists of 30 atom types. These directly map to peptide and nucleic acid atoms and thus permit a clearly defined assignment of types to protein atoms. For other molecule types, most notably ligands, there is no trivial mapping procedure. The SuperTrAmber force field (see Section 3.4.1) therefore accepts ligands characterized by SYBYL atom types.

3.5.2. SYBYL atom type model. SYBYL atom types were introduced for the force field TAFF [**166**], which is targeted at simulating small molecules. SYBYL types are therefore suitable for representing small organic molecules typically found in the field of drug development.

3.5.3. MMFF94 atom type model. According to the MMFF94 model, each atom is assigned one of 95 available atom types [**172**]. These comprise information on an atom's element type, its hybridization state, the number and types of bonds it forms, and its aromaticity. Furthermore, the MMFF94 atom type is often linked to specific functional group that an atom is part of. Thus, MMFF94 atom types allow for mapping molecular information to a comparatively complex but informative representation.

3.5.4. Naomi. The Naomi model [**177**] represents atoms on three levels of abstraction. The basic level captures information derivable from an atom's element such as the van der Waals radius and whether the atom is a metal. On the second level of abstraction, valence state information is stored, which captures an atom's formal charge, number of bonds, and bond orders. Finally, the third level of abstraction provides information on aromaticity and alternative resonance forms.

3.6. Partial Charge Models

For computing the Coulomb energy, molecular mechanics force fields use partial atomic charges. They represent the distribution of electrons in chemical bonds, which is non-symmetric and fluctuating. Thus, only a time-dependent function can correctly represent molecular charge distributions. Hence, partial charge constitutes an approximative approach. Different models have been developed and the following introduces those relevant to this work.

3.6.1. Gasteiger-Marsili partial charge model. Gasteiger and Marsili [**171**] were the first to develop a method for computing partial atom charges. Their approach distributes charges among the constituent atoms of a molecule according to electronegativity differences. For this, the bonds of a molecule serve as a network for disseminating charges. In an iterative procedure, the initial charges are transferred between neighboring atoms based on the difference in electronegativity. In each step, the electronegativities are recomputed and thus adapted to the newly derived partial atom charges. To avoid an over-accumulation of charges, a damping factor limits the charge transfer. The algorithm exponentially reduces this factor in each round and aborts as soon as it reaches convergence. This work implements the Gasteiger-Marsili partial charge algorithm as part of the force field framework (see Section 4.2).

3.6.2. MMFF94 charge model. The MMFF94 model assigns partial charges to atoms based on their formal charge and on the electronegativity of the atoms they are bound to. In detail, an atom i with formal charge q_i^0 is assigned the partial charge q_i:

$$q_i = q_i^0 + \sum \omega_{KI}$$

where ω_{KI} is the *bond charge increment*, which specifies the partial charge in elementary charge unit that a neighboring atom k of type K transfers to i. The magnitude of ω_{KI} is decided by the polarity of the bond between atoms i and k and thus captures the electronegativity gradient between i and k.

Delocalized charges are handled by distributing formal charges among atoms of the affected functional group. For example, the unit charge in a deprotonated carboxyl group is split between its two oxygen atoms.

3.6.3. AMBER3 charge model. For deriving AMBER3 charges, Weiner et al. **[68]** rely on fitting partial charges to quantum mechanically calculated electrostatic potentials. This method is conformation dependent, only applicable to smaller molecular fragments, and yields varying results for different basis sets. However, Weiner et al. modified their model wherever calculated charges caused significant deviations of AMBER3 potential energies from experimentally derived values.

To establish a charge model for peptides, Weiner et al. first constructed a consistent set of charges for peptide backbone atoms, irrespective of peptide side chains. To achieve this, they mapped a quantum mechanically calculated electrostatic potential surface onto dipeptide nuclei. In the next step, the derived point charges were fitted such that the AMBER3 model reproduced experimentally derived potential energy values.

Subsequently, polar side chains were broken down into the *buffer* and the *chromophore* fragments. The former comprised the α and β carbons with their corresponding hydrogens and the latter comprised the remaining side chain. For chromophores, representative molecules were selected, for which quantum mechanically derived charges were computed. For example, phenol and imidazole were used as chromophores for tyrosine and histidine, respectively. The computed charges were mapped onto the nuclei of the side chain.

Peptides with polar side chains thus consist of a backbone fragment with partial charge -0.246, a buffer, and a chromophore fragment. The latter is either neutral, singly anionic, or singly protonated. Excess charges are distributed between atoms of the buffer fragment. Hydrocarbon side chains are regarded as a buffer fragment as a whole and thus have a partial charge of $+0.246$.

Hence, AMBER3 provides pre-calculated partial charges for all relevant amino acids, including histidines protonated at both the ε and δ positions. These partial charges are looked up according to amino acid types and then assigned to residue atoms.

3.7. Grid-Based Acceleration of Force Field Calculations

To reduce the complexity of computing SuperTrAmber's non-bonded interaction energies (see Section 3.4.1) from $O(N^2)$ down to $O(N)$, the van der Waals and electrostatic forces exerted by proteins were projected onto a cubic lattice. This concept is largely based on unpublished ideas by Klein and is also widely used in docking tools, e.g., ICM **[22, 23]**, Glide **[24]**, AutoDock **[88–91]**, CDOCKER **[95, 96]**, and DOCK **[62–67, 70, 71]**.

The three-dimensional grid encloses the binding pocket region surrounding a given ligand. Each grid point represents a probe atom and stores force field energies for its location. Electrostatic and van der Waals energies are handled in separated grids, as described in the following.

For pre-computing electrostatic energies, a singly charged ($q_g = 1$) probe atom g iterates over all grid points. At each of these, the electrostatic energy resulting when g interacts with all protein atoms within a cutoff radius of 16 $\mathring{A}$ is computed. As SuperTrAmber's electrostatic energy for two atoms i and j is:

$$E_{elec} = \frac{332.05222 \cdot q_i \cdot q_j}{D \cdot R_{ij}}$$

the electrostatic energy at grid point (x_g, y_g, z_g) amounts to:

$$E_{elec}(x_g, y_g, z_g) = \frac{332.05222 \cdot q_g}{D} \cdot \sum_{j=1}^{|P|} \frac{q_j}{\sqrt{(x_g - x_j)^2 + (y_g - y_j)^2 + (z_g - z_j)^2}}$$

where $|P|$ is the total number of atoms within cutoff distance in protein P and (x_j, y_j, z_j) are the coordinates of protein atom j.

In the hypothetical case that a ligand atom l with charge q_l is located exactly on a point of the pre-calculated grid with coordinates (x_l, y_l, z_l), the electrostatic interaction energy between l and all protein atoms boils down to $q_l \cdot E_{elec}(x_l, y_l, z_l)$. In the more realistic case of l being located between grid points, this research applies a trilinear interpolation, as introduced in Section 3.3.4.

Furthermore, SuperTrAmber's van der Waals energy formula is rearranged to pre-compute energies for grid points. For a pair of atoms i and j, the van der Waals energy is given by:

$$E_{vdW_{ij}} = \frac{\varepsilon_{ij} \cdot R_{ij}^{*12}}{R_{ij}^{12}} - \frac{2 \cdot \varepsilon_{ij} \cdot R_{ij}^{*6}}{R_{ij}^{6}} \tag{1}$$

where the van der Waals well-depth ε_{ij} for atoms i and j is derived from the well-depth parameter of the single atoms such that $\varepsilon_{ij} = \sqrt{\varepsilon_i} \cdot \sqrt{\varepsilon_j}$. Furthermore, R_{ij}^* is the sum of the van der Waals radii of atoms i and j and thus $R_{ij}^* = R_{vdW_i} + R_{vdW_j}$. With these substitutions, Equation 1 can be rewritten as:

$$E_{vdW_{ij}} = \sqrt{\varepsilon_i} \cdot \sqrt{\varepsilon_j} \left(\frac{(R_{vdW_i} + R_{vdW_j})^{12}}{R_{ij}^{12}} - \frac{2 \cdot (R_{vdW_i} + R_{vdW_j})^{6}}{R_{ij}^{6}} \right)$$

This allows pre-computing the van der Waals energy between probe atom g with $\sqrt{\varepsilon_g} = 1$ and all atoms within cutoff distance of protein P at all grid points:

$$E_{vdW}(x_g, y_g, z_g) = \sum_{j=1}^{|P|} \sqrt{\varepsilon_j} \left(\frac{(R_{vdW_g} + R_{vdW_j})^{12}}{R_{gj}^{12}} - \frac{(R_{vdW_g} + R_{vdW_j})^{6}}{R_{gj}^{6}} \right) \tag{2}$$

As the van der Waals radius R_{vdW_g} of atom g cannot be extracted from Equation 2, five are computed grids using probe atoms with radii 1.0 $\mathring{A}$, 1.6 $\mathring{A}$, 1.7 $\mathring{A}$, 1.85 $\mathring{A}$, and 2.0 $\mathring{A}$, respectively. These probe atoms represent generic hydrogen, oxygen, nitrogen, carbon, and sulfur according to the AMBER3 atom model (see Section 3.5.1).

The van der Waals energy of an ideal atom l with well-depth parameter $\sqrt{\varepsilon_l}$, a van der Waals radius matching that of one of the grids, and coordinates (x_l, y_l, z_l) equal to those of a grid point can now be computed as $\sqrt{\varepsilon_l} \cdot E_{vdW}(x_l, y_l, z_l)$. If the van der Waals radius of

atom l does not match that of any of the five pre-computed grids, this research applies a linear interpolation. Energies for atoms located between grid points are again approximated with the trilinear interpolation method described in Section 3.3.4.

3.8. In-House Software Tools

3.8.1. HYDE. To approximate the free energy of binding, Reulecke et al. [**178**] and Schneider et al. [**179**] developed HYDE, a scoring function based on a consistent model for hydrogen bond and dehydration energies in protein-ligand complexes. HYDE takes the formation and breaking of hydrogen bonds and thus, the dehydration of ligands and binding sites into account. This way, the HYDE function captures major energetic contributions determining the formation of protein-ligand complexes. In contrast to other available scoring methods, HYDE's parameters are not trained on experimental binding affinity data. Instead, logP increments mapped to atoms serve as a basis for evaluating protein-ligand complexes. HYDE was successfully evaluated in the course of a docking and scoring symposium that took place during the 241$^{\text{st}}$ ACS National Meeting [**101**].

3.8.2. ProToss. ProToss [**180**] assigns hydrogen atoms to binding pockets of protein-ligand complexes. Commonly used crystallographic methods do not resolve hydrogen atoms. Their positions depend on tautomeric and protonation states, as well as torsional changes and are thus reflect a given molecular environment. ProToss employs a dynamic programming algorithm to place hydrogen atoms in accordance with a scoring model similar to that of FlexX [**48**].

3.8.3. TrixX. TrixX [**102**, **106**] is a docking tool that places pre-generated ligand conformations in protein binding pockets. By employing database-stored and indexed bitmaps for describing small molecules, TrixX screens large conformation sets in sublinear runtime. For scoring ligand poses, TrixX utilizes a hierarchical scoring scheme based on the FlexX [**48**] model.

3.8.4. TrixX Conformation Generator. Griewel et al. [**181**] developed the *TrixX Conformation Generator* (*TCG*) to generate ensembles of molecular conformations that exhibit a low RMSD to biologically active conformers. The TCG represents molecules in a tree-based data structure. This permits an incremental construction procedure for conformers in which rotatable dihedral bonds constitute degrees of freedom. Consequently, the algorithm has the means to reproduce large parts of a molecule's conformational space. This can entail a considerable computational effort. Thus, the TCG allows setting a quality level to vary the balance between accuracy and number of generated conformations.

3.8.5. CONFECT. Like the TCG, CONFECT [**182**] generates ensembles of molecular conformations. While both tools are algorithmically similar, CONFECT's evaluation of generated conformations is based on frequency distributions of torsional bond angles extracted by Schärfer et al. [**183**] from the *Cambridge Structural Database* (*CSD*) [**184**]. According to Schärfer's model, substructures defining a torsion angle are associated with a set of legal angle intervals. For each substructure, this set is derived from standard deviations of observed torsional angle

distributions. Threshold factors—named level I and II—are applied to the standard deviations to define the parts of the distributions to be considered. For accessing angle interval sets, CONFECT provides an internal interface.

3.9. External Software Tools

3.9.1. YASARA. YASARA [**40**] is a molecular modeling and simulation package that provides scripted access to force field-based scoring and minimizations. For performing the latter, the author prepares molecules with YASARA's `Clean` function, which adds missing hydrogen atoms, and then employs YASARA's `Experiment:Minimization` procedure as follows. The structure is energy-minimized, to remove bumps and correct the covalent geometry, using the AMBER03 force field (see Section 3.4.5) when optimizing ligands in rigid binding pockets. All other scenarios employ the AMBER94 force field (see Section 3.4.4). Force cutoffs are imposed, as described in the respective section. After removal of conformational stress by a short steepest descent minimization, the procedure continues by SA (time step 2 fs, atom velocities scaled down by 0.9 every 10th step) until convergence is reached, i.e., the energy improves by less than 0.05 kJ/mol per atom during 200 steps.

3.9.2. MOE. The *Molecular Operating Environment* (*MOE*) [**28**] offers a wide range of molecular modeling and simulation tools. Scripted access is granted by the included `svl` language. MOE's force field-based scoring, and particularly its gradient-based deterministic minimization procedure, is employed. In effect, this procedure is a succession of three methods: steepest descent [**9**], conjugate gradient [**8**], and truncated newton. MOE's reference manual provides explicit directions for applying it.

In detail, the AMBER03 force field (see Section 3.4.5) serves as objective function for optimizing ligands in rigid binding pockets. Otherwise, this research employs the AMBER94 force field (see Section 3.4.4). Preparatory measures include adding hydrogen atoms to proteins according to MOE's model. Furthermore, partial charges matching the employed force field are computed. For fixing atoms during minimization procedures, this work sets their tether weight values to 100,000. For implementing a cutoff on non-bonded interactions, MOE employs a switching function. The experimental descriptions section specifies its interval. Finally, MOE's `MM[]` function carries out the optimization until the energy gradient becomes smaller than $0.05\,RMS\frac{kcal}{mol}/\mathring{A}^2$.

Two versions of MOE were used for this work: version 2010.10 for minimizing ligands in rigid binding pockets and version 2012.10 in all other cases.

3.10. Hypothesis Testing

In this work, hypothesis testing boils down to the following problem: given the *samples* a and b drawn from *population distributions* P_A and P_B, respectively. Determine whether P_A and P_B differ. Obviously, various characteristics allow comparing P_A with P_B. For this work's experiments, the most relevant among those is the mean (μ). Additionally, the median ($\tilde{x}$) derived from samples facilitates statistical inferences on the underlying population distribution.

To employ these key characteristics for reasoning, this work adheres to the following standard procedure for hypothesis testing:

(1) Formulate a *null hypothesis* H_0. This work assumes that P_A and P_B are the same.
(2) Set the criterion for deciding whether H_0 is true. As P_A and P_B are inaccessible, the decision is based on characteristics derived from the samples. In detail, this work computes $p(b)|P_A$ (abbr. p), which is the probability of drawing b from P_A. H_0 is rejected if p is below the *level of significance*.
(3) Select and compute the test statistics. Depending on the characteristics of the *sample distributions* derived from a and b, an appropriate test for computing p is selected.
(4) Accept or reject H_0. Unless p is less or equal to the standard level of significance, which is 0.05, it is assumed that H_0 is true. Otherwise, the alternative hypothesis H_1, which is the logical opposite of H_0, is true. Further levels of significance used in this work are $p \leq 0.01$ and $p \leq 0.001$. They signify an enhanced confidence that H_0 is false.

3.10.1. Test statistics. For appropriately choosing test statistics, the experiments derive from the samples whether the population distributions are normally distributed and assume that this is not the case if a sample contains severe outliers or if a sample's mean and median value exhibit large differences. For normally distributed populations, this research compares sample distribution mean values. If at least one population is not normally distributed, the experiments resort to median-based test statistics.

A further step decides on which specific test to apply. For this, *Levene's test* [**185**] assesses whether the variances of the population distributions differ. Therefore, this work utilizes the `levene` function of the `scipy.stats` package [**186**, **187**] for Python.

Mean testing. Given two population distributions P_A, P_B, the null hypothesis H_0 assumes that the means of the population distributions are equal $\mu_{P_A} = \mu_{P_B}$. However, P_A and P_B are inaccessible. Currently available are the samples a and b. As $\mu_{P_A} = \mu_{P_B}$, the *mean difference of the sample means* $\mu_{\bar{A}-\bar{B}}$ should be zero. As this is an average value, the difference of the means of the samples $x = \bar{a} - \bar{b}$ most probably differs from $\mu_{\bar{A}-\bar{B}}$.

Given the samples a and b, *t-statistics* allow for deriving the probability that $x = \bar{a} - \bar{b}$ under the pre-condition that $\mu_{\bar{A}-\bar{B}} = 0$. This is because the *t-distribution* yields an empirical value for $\mu_{\bar{A}-\bar{B}}$ derived from a *sample mean*, a *sample variance*, and a sample size that is termed *degrees of freedom*. Also, the t-distribution allows for computing the likelihood of the empirically derived mean difference of sample means. Therefore the *t-value* has to be computed. Depending on characteristics of the examined population distributions, different methods are applicable for this task.

Assuming that P_A and P_B are normal distributions with equal variance, *Student's t-test* [**188**] is applicable. Based on the samples a and b of size n and m, respectively, the t-value is computed:

$$t = \frac{\bar{a} - \bar{b}}{\sigma_{\bar{A}-\bar{B}}} \cdot \sqrt{\frac{nm}{n+m}}$$

where $\sigma_{\bar{A}-\bar{B}}$ is the pooled standard deviation of the distribution of the differences of sample means. It is approximated by weighting the corrected sample standard deviations s_a and s_b according to the sample sizes:

$$\sigma_{\bar{A}-\bar{B}} = \sqrt{\frac{(n-1)s_a^2 + (m-1)s_b^2}{n+m-2}}$$

Furthermore, the degrees of freedom $df = n + m - 2$ are computed. If $|t| > t(1 - 0.525, df)$ then the probability that $x = \bar{a} - \bar{b}$ is less that 0.05 given that H_0 is true, and therefore H_0 is rejected.

If Levene's test indicates that the variances of P_A and P_B differ, the t-value is computed according to *Welch's t-test* [**189**]:

$$t = \frac{\bar{a} - \bar{b}}{\sqrt{\frac{s_a^2}{n} + \frac{s_a^2}{m}}}$$

Here, s_a and s_b are the corrected sample standard deviations of a and b, respectively. The degrees of freedom are approximated with the *Satterthaite-Welch adjustment* [**190**] such that

$$df = \frac{(\frac{s_a^2}{n} + \frac{s_b^2}{m})^2}{\frac{\frac{s_a^2}{n}}{n-1} + \frac{\frac{s_b^2}{m}}{m-1}}$$

The criterion for rejecting H_0 is the same as for Student's t-test.

All t-tests are computed with the `ttest_ind` function that is part of the `scipy.stats` package [**186**, **187**] for Python.

Median testing. Whenever P_A or P_B is not normally distributed, median-based statistical tests are applied. They are centered around the assumption that given P_A and P_B are equal it follows that on average the difference of the sample medians must be zero.

Under the pre-condition that the variances of P_A and P_B are equal, the *Mann-Whitney U test* [**191**] is applicable. It merges and then ranks the *observations* that two samples a and b are composed of according to the observation values. Then, the sum of ranks R_a and R_b are computed for a and b. With these values, the test computes the U-statistics

$$U = R - \frac{n(n-1)}{2}$$

which allows for deriving the probability of the difference of the medians of s_a and s_b given that the H_0 is true. This work computes the Mann-Whitney U test with the `mannwhitneyu` function, which is part of the `scipy.stats` package for Python.

If the variances of P_A and P_B are unequal, *Mood's median test* [**192**] is applied. It merges samples a and b to determine the common median. Subsequently, the merged observations are split into those smaller and those larger than the common median. For both obtained sets, a chi-squared test is performed to assess whether samples from a and b exhibit differing distributions.

CHAPTER 4

Developed Methods

Quick force field-based protein binding pocket and small ligand minimizations necessitate a fast and versatile optimization method with easily adaptable functionality. Furthermore, a compact and standardized molecular representation is required. Therefore, in Section 4.1, a data structure for uniformly representing all types of molecules occurring in protein binding pockets is introduced. Operating on it, a force field framework, which is outlined in Section 4.2, facilitates implementing, mixing, and switching between energy terms from different all-atom force fields. Based on this framework, the force fields SuperTrAmber and MMFF94s (see Sections 3.4.1 and 3.4.3) were fully implemented. For these force fields, accelerated GPU- and grid-based algorithms were developed that overcome runtime limitations otherwise coupled to force field-based minimizations, which Sections 3.7, 4.3, and 4.4 discuss. The sections that follow introduce a broad spectrum of force field-based objective functions. To harness them, a framework for performing SA-based optimizations was developed, which is introduced in Sections 4.5 and 4.6. This framework seamlessly integrates with pharmacophore-like constraints, as discussed in Section 4.7. Finally, data preparation methods for manipulating binding sites are presented in Section 4.9.

4.1. Molecule Data Structure

Force field-based optimizations evaluate and modify molecule conformations. To achieve this, they require atom coordinates and molecular connectivity information. This work condenses this information in the molecular representation, which provides an unified interface for accessing proteins, cofactors, and other small molecules.

4.1.1. Molecule and atom representation. This molecular representation, which is shown in Figure 4.1, is centered around the `Molinfo` component. It provides access to its `FFAtom` components that store coordinate, type, and charge data of single atoms. Furthermore, the `Molinfo` component contains the *component tree* that stores all molecular connectivity information required during optimization runs.

In contrast, during the initialization phase of force field-based scoring the complete molecular graph is required. It is only comprised in the external all-purpose molecular data structures of Naomi (see Section 3.5.4). An interface to these is provided by the `Molinfo` and `FFAtom` components.

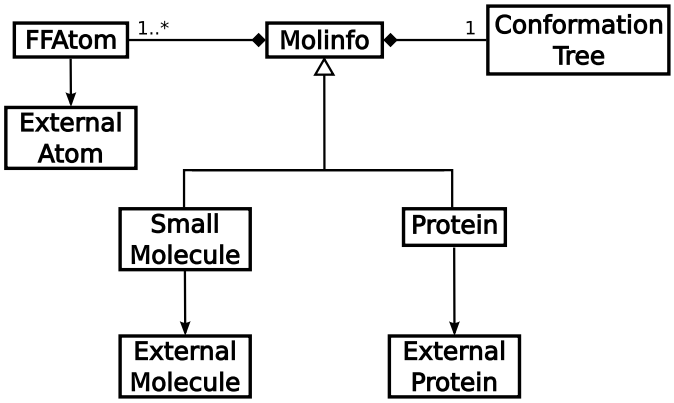


FIGURE 4.1. Components of the molecule data structure. `Molinfo` provides a common interface for accessing proteins and small molecules. Atom-specific data, most notably coordinates, are stored in `FFAtom` components. Information stored in external data structures only is accessed via interface functions.

To represent proteins and small molecules, the components `Protein` and `SmallMolecule` derive from `Molinfo`. They implement specialized functionality to which the `Molinfo` component provides a common interface.

4.1.2. Component tree. The component tree is an undirected graph $G = (V, E)$ whose vertices V represent *components*. A component comprises a set of atoms that are connected by *non-rotatable bonds*, which are defined as bonds that are not *rotatable bonds*. These, in turn, are single bonds connecting non-terminal heavy atoms. In a component tree, rotatable bonds are represented by edges E. Figure 4.2 illustrates the division of a molecule into components that are connected by rotatable bonds.

The component tree is constructed using the `DetectComponents` algorithm (see Algorithm 4.1). It is based on a *breadth-first-search* (*BFS*) over the components V of a molecule. The algorithm is initialized by inserting an arbitrary atom of a given molecule into the *seed queue*. This instance contains atoms that are passed to the `AnalyzeComponent` procedure (see Algorithm 4.2). For each member of the seed queue, this procedure starts the detection of connected atoms that then form a new component $v \in V$. For this, the `AnalyzeComponent` procedure employs a modified

Algorithm 4.1 Adapted BFS algorithm for constructing a component tree

```
DETECTCOMPONENTS(molecule)
1  PUSH(seedQueue, GETFIRSTATOM(molecule))
2  visited ← NEWVECTOR(GETNOFATOMS(molecule))
3  SETALLFALSE(visited)
4  while !ISEMPTY(seedQueue)
5        do APPEND(newComponents, ANALYZECOMPONENT(seedQueue, visited))
6  return newComponents
```

depth-first-search (*DFS*). It pushes the top atom of the seed queue onto a DFS stack. Then, the algorithm loops over the bonds of the top atom of the DFS stack.

If a bond is rotatable, the algorithm creates a new edge e, adds it to the edges E of the component tree and associates it with the new component v. The newly discovered opposite atom a reachable through e is pushed to the front of the seed queue.

If a bond is non-rotatable, the opposite atom a is added to the new component v and pushed on top of the DFS stack. The algorithm then restarts its main loop and processes the top element of the DFS stack. After the `DetectComponents` algorithm has finished, the elements of

Algorithm 4.2 Adapted DFS algorithm for detecting atoms forming a component

```
AnalyzeComponent(seedQueue, visited)
 1  push(DFSStack, pop(seedQueue))
 2  while !isEmpty(DFSStack)
 3        do stackSizeBefore ← size(DFSStack)
 4           topAtom ← top(DFSStack)
 5           for bond in getBonds(topAtom)
 6                 do oppositeAtom ← getOpposite(bond, topAtom)
 7                    if ! visited[oppositeAtom]
 8                       then visited[oppositeAtom] ← true
 9                            if isRotatable(bond)
10                               then pushBack(newComponent.bonds, bond)
11                                    push(seedQueue, oppositeAtom)
12                               else push(DFSStack, oppositeAtom)
13                                    break
14           if stackSizeBefore = size(DFSStack)
15              then pushBack(newComponent.atoms, pop(DFSStack))
16  return newComponent
```

the component tree are post-processed. Therefore, all its edges E are copied and then stored separately. Additionally, unique IDs are created for the rotatable bonds of the analyzed molecule. These IDs are mapped to the component tree edges E. Furthermore, the post-processing assures that each component $v \in V$ stores all edges connected with it. In components that contain atoms that should not be transformed during an optimization, a flag signaling rigidity is set.

Subsequently, information on reachable components is added to the edges E. Each edge $e \in E$ has a "from" and a "to" direction. For each edge and each direction, a DFS algorithm compiles a list of reachable components. These lists are stored in the edge data structure and are subsequently employed to add two flags to each edge. These tell whether there are rigid components in "from" and "to" direction. Figure 4.2 depicts a fully initialized component tree resulting from the described procedure.

Later, the component tree is utilized to quickly determine which components are affected by rotations around dihedral bonds and whether there are any rigid ones among them.

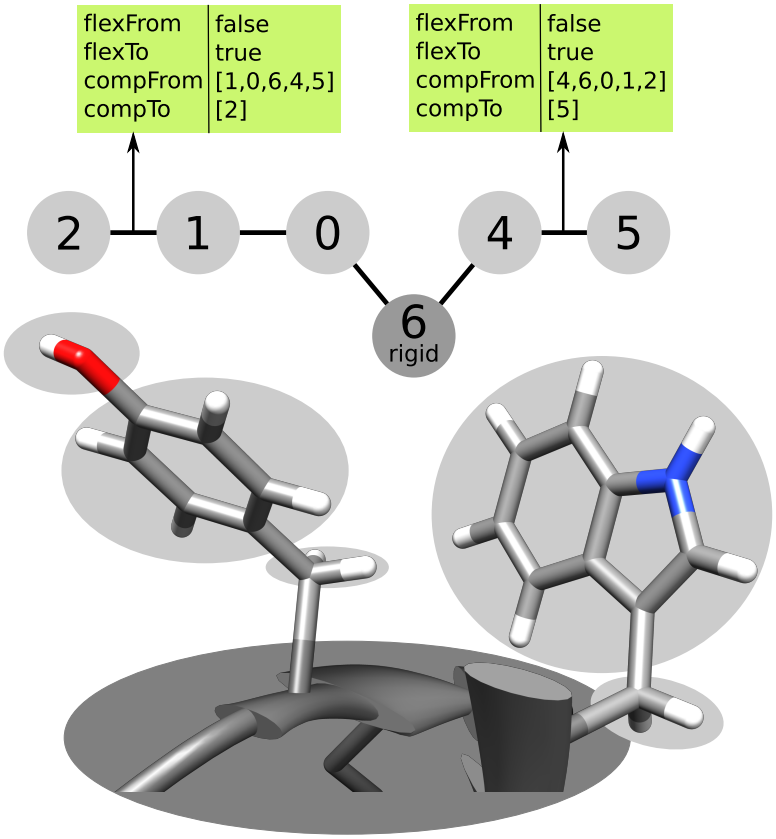


FIGURE 4.2. Component tree representing flexible amino acid side chains (light gray) and a rigid bulk that encompasses backbone and non-flexible side chain atoms (dark gray). Component tree vertices are labeled as rigid when representing a non-flexible region (see vertex 6). Component tree edges contain the `compFrom` and `compTo` lists that store the IDs of vertices reachable when traversing the edge either leftwards or rightwards. If during this traversal a rigid component is encountered, the `flexFrom` and `flexTo` flags are set to `false` accordingly.

4.1.3. Modifying the component tree for protein flexibility. For processing flexible amino acid side chains, this work modifies the component tree initialization procedure. Therefore, the user initially either spatially defines a flexible region with a set of spheres or provides a list of flexible amino acids. Irrespective of the selection method, proline, alanine, and glycine as well as metals and cofactors are treated as rigid. Subsequently, the `DetectComponents` procedure (see Algorithm 4.1) is started. As before, it assigns side chain atoms of flexible residues to components. To exclude backbone and non-flexible amino acid atoms, the procedure considers C_α atoms as terminal. Consequently, they are not pushed into the seed queue. When the `DetectComponents` procedure finishes, all atoms of the backbone and of non-flexible amino acid

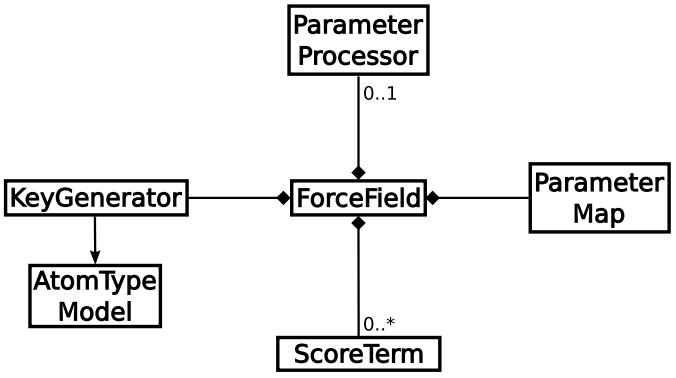


FIGURE 4.3. Components of the force field representation. The `KeyGenerator` generates keys for extracting parameters from the `ParameterMap`, a parameter database. An optional `ParameterProcessor` then optimizes the extracted parameters while `ScoreTerm` components get a set of parameters as input for computing energy values.

side chains are assigned to a single component, which is flagged as rigid. The initialization procedure then continues as described previously. Figure 4.2 depicts an example of a component tree representing flexible amino acid side chains.

4.2. Framework for All-Atom Force Fields

This section presents a framework that provides a unified implementation and access scheme for all-atom force fields. It facilitates scoring molecular systems with different force fields via a consistent interface.

4.2.1. Common functionality of force fields. The design of the framework is derived from the observation that commonly used classical all-atom force fields share certain functionality. SuperTrAmber, MMFF94(s), AMBER (see Section 3.4), as well as OPLS-AA [**25**, **26**] and CHARMM [**97**], evaluate non-bonded atom pairs and molecular moieties to produce an energy value. These molecular moieties are invariably atomic bonds, angles, dihedral angles, and out-of-plane bonds. Additionally, 2$^{\text{nd}}$ generation force fields [**193**] like MMFF94 can include cross terms. As for atom pairs, some force fields treat hydrogen bonds differently. Still, the similarities among all-atom force fields permit a unified detection procedure for molecular fragments and atom pairs.

Using force field-specific atom type models, these fragments and atom pairs are then categorized. This allows for applying a consistent parameterization scheme. In this way, force fields impose a model on molecules that transforms them into lists of variable atom coordinates and constant parameters. The latter also comprises atomic partial charges computed with suitable models. Finally, term-specific functions operate on the parameter lists to compute potential energy values.

4.2.2. Components of force fields. The sections following this analysis divide the functionality of all-atom force fields into the components depicted in Figure 4.3. Thus, a *strategy pattern* **[194]** arises that allows for altering functionality by replacing components.

According to this work's model, a given combination of atoms along with an energy term type is passed to the `KeyGenerator` by the `ForceField` component. This component constructs an energy term-specific key. During this process, it utilizes the `AtomTypeModel` component to determine atom types for the given combination of atoms. The `ForceField` component subsequently passes the resulting key to the `ParameterMap` component. If the key refers to a valid entry, this triggers the extraction of scoring parameters. Optionally, the `ForceField` component passes these parameters to the `ParameterProcessor` component, which optimizes or dynamically adapts them to certain molecular fragments. Finally, the `ForceField` component returns the processed atom combination along with the extracted parameters.

To obtain energy values, a set of scoring parameters along with the desired energy term type is passed to the `ForceField`. This component consequently selects the suitable `ScoreTerm` component, which then employs the scoring parameters to compute an energy value.

4.2.3. Force field initialization. To initialize force fields, the author designed an XML-based file format that stores information for registering force fields, their energy term types, and parameters for defined atom combinations. The file format permits mixing terms of several force fields and stores their parameters in a human-readable way.

This file format is formally defined by Grammar 4.1. Therein, ⟨*forcefield-collection*⟩ is a start symbol and {'`float`','`int`', ⟨*string*⟩, ⟨*float*⟩ ,⟨*int*⟩} are terminal symbols. The latter three represent arbitrary character sequences that are casted to the denoted type.

GRAMMAR 4.1. XML-based file format for storing force field parameter and term type information

⟨*forcefield-collection*⟩ ::= ⟨*forcefield*⟩ | ⟨*forcefield-collection*⟩ ⟨*forcefield*⟩

⟨*forcefield*⟩ ::= ⟨*name*⟩ ⟨*term*⟩

⟨*term*⟩ ::= ⟨*name*⟩ ⟨*key_value_pair*⟩

⟨*key_value_pair*⟩ ::= ϵ | ⟨*atom_type*⟩ | ⟨*key_component*⟩ | ⟨*parameter*⟩
| ⟨*key_value_pair*⟩ ⟨*key_value_pair*⟩

⟨*atom_type*⟩ ::= ⟨*name*⟩ ⟨*val*⟩

⟨*key_component*⟩ ::= ⟨*name*⟩ ⟨*val*⟩

⟨*parameter*⟩ ::= ⟨*type*⟩ ⟨*name*⟩ ⟨*val*⟩

⟨*type*⟩ ::= '`float`' | '`int`'

⟨*name*⟩ ::= ⟨*string*⟩

⟨*val*⟩ ::= ⟨*string*⟩ | ⟨*float*⟩ | ⟨*int*⟩

When initialized, the `ForceField` component parses a parameter file. During this process, it passes pairs consisting of an atom type combination and a parameter set to the `ParameterMap` database for storage. For loading the energy terms specified in the XML file, the `ForceField` component has to be provided with force field-specific mappings from energy term types to `ScoreTerm` components. Using this mapping, energy terms are registered in the force field and associated with an energy term type.

4.2.4. Scorer component. For energetically evaluating sets of molecules, the author developed the `Scorer` component. It detects molecule moieties as well as atom pairs and utilizes the `ForceField` component for obtaining matching scoring parameters. With these and the help of the `ForceField` component, the `Scorer` component computes potential energy values.

Detecting Bonded Interactions. The `Scorer` component detects fragments that constitute the following bonded interaction types: bonds, angles, dihedrals, out-of-plane, and stretch-bend. To perform this task, it traverses the molecular graph.

Bonds connecting two atoms are detected with the `FindBonds` algorithm (see Algorithm 4.3) that is an adapted DFS. Starting from a given atom a, the algorithm iterates over all bonds (a, b)

Algorithm 4.3 An adapted DFS algorithm for detecting bonds in a molecular graph.

```
FindBonds(parentAtom, currAtom, color)
 1  ▷ color code: white=undetected, gray=detected, black=fully processed
 2  color[currAtom] ← GRAY
 3  while neighAtom ← getNextNeighborAtom(currAtom)
 4        do if neighAtom = parentAtom
 5              then continue
 6           if color[neighAtom] = WHITE or color[neighAtom] = GRAY
 7              then addBond(currAtom, neighAtom)
 8                   if color[neighAtom] = WHITE
 9                      then FindBonds(currAtom, neighAtom, color)
10  color[currAtom] = BLACK
```

to neighboring atoms b. If b is not already marked as completely processed, the algorithm stores (a, b). If b is not marked as already visited, the algorithm recursively calls itself with b as the atom to process next. After having visited all its neighbors, the algorithm marks a as completely processed.

By detecting all unique paths of length two in the molecular graph, molecular parts that form scorable angle and stretch-bend interactions are identified. For this, the `Scorer` component again utilizes an adapted DFS algorithm. It traverses the molecular graph and iterates for each newly discovered atom a over pairs of neighboring atoms b, c such that $b \neq c$. Thus, the algorithm detects the set all unique paths $P = \{\langle b_0, a_0, c_0\rangle \dots \langle b_n, a_n, c_n\rangle\}$.

Atoms associated to paths of length three in the molecular graph form dihedral angles. The `Scorer` component offers two methods for detecting them. The first one solely considers dihedral

angles with central bonds that are rotatable. These are detected by iterating over all edges of the component tree. Each edge represents a rotatable bond (a, b). With (a, b) as central bond, all unique paths $\langle c, a, b, d\rangle$ where $c \neq d$ are enumerated. The second method detects all dihedral angles in a molecule. It iterates over all molecular bonds (a, b) and then enumerates the set of all unique paths like the first method.

Molecular mechanics force fields utilize out-of-plane energy terms to force atoms into a plane often spanned by member atoms of rings. Without this correction, the dihedral term would drive the affected molecule parts into energetically unfavorable conformations. The out-of-plane term therefore operates on fragments formed by a central atom that is covalently bound to three neighbor atoms. These fragments are detected by iterating over all atoms and checking whether they have three distinct neighbor atoms.

Detecting non-bonded interactions. In general, all-atom force fields evaluate all pairs of atoms in molecular systems. However, some force fields employ differing parameters for hydrogen bonds. Furthermore, intramolecular interactions between two atoms (a, b) for which a path $p_{a,b}$ of length one or two (1-2 interactions and 1-3 interactions) exists are typically ignored and 1-4 interactions for which $p_{a,b}$ of length four exists are oftentimes dampened (see e.g., SuperTrAmber force field in Section 3.4.1). Furthermore, runtime-lowering cutoffs introduce a threshold c according to which all atom pairs (a, b) with $dist(a, b) > c$ are ignored.

Consequently, the `Scorer` component iterates over all atom pairs (a_i, a_j) with $j > i$, imposes a cutoff c by ignoring all (a_i, a_j) with $dist(a_i, a_j) > c$ and determines the path length of p_{a_i,a_j} if a_i and a_j are part of the same molecule. For this, the `Scorer` component employs a BFS over the molecular graph starting at a_i. This search terminates as soon as it is certain that the path length is larger than three.

To determine whether the pair (a_i, a_j) forms a hydrogen bond according to the selected force field model, the `Scorer` component requests hydrogen bond scoring parameters for (a_i, a_j) from the `ForceField` component. If that attempt fails, it is assumed that (a_i, a_j) does not form a hydrogen bond.

4.2.5. ParameterProcessor component. Following the detection of molecule moieties, the `ParameterProcessor` can modify parameters. This allows for pre-calculating force field-specific constant values and for dynamically adapting parameters to unusual moieties.

Parameter processing constitutes the end of the initialization phase. The `Scorer` component now stores all relevant scoring parameters in term-specific arrays. This permits computing energy values for all initialized energy terms.

4.2.6. SuperTrAmber implementation. Components for the force field framework that fully implement the SuperTrAmber force field (see Section 3.4.1) were developed. In this process, a main design objective was the fast calculation of non-bonded interactions. Hence, these are processed in a single `ScoreTerm` component. This allows to compute the distance for each atom pair just once and then using it in all non-bonded interaction terms.

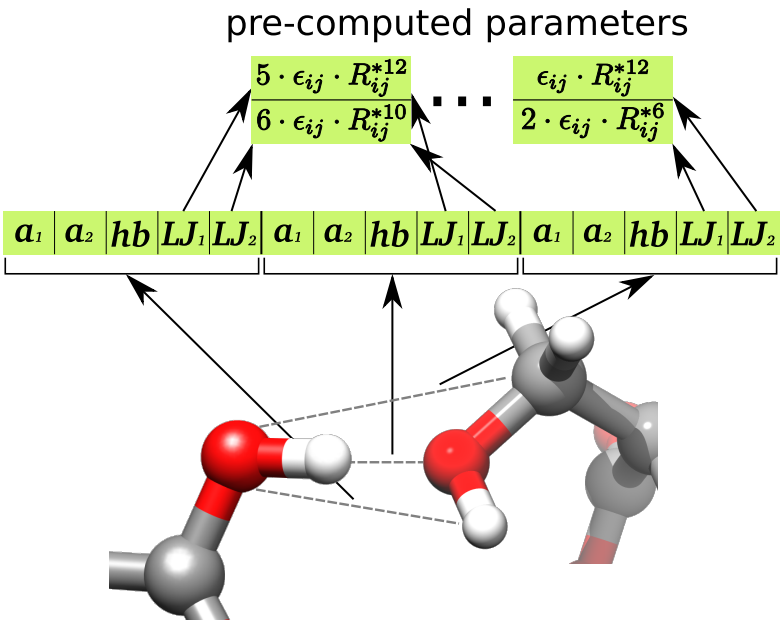


FIGURE 4.4. Data structure for non-bonded interaction scoring parameters of the SuperTrAmber force field. For each atom pair, pointers to the respective `FFAtom` data structures (a_1 and a_2) are stored. Furthermore, a hydrogen bond flag (hb) permits selecting an alternative potential for hydrogen bonds at runtime. Finally, the numerator of the Lennard-Jones potential is pre-calculated and stored for all combinations of atom types in advance. LJ_1 and LJ_2 point to the numerator corresponding to their respective atom pair.

Furthermore, a `ParameterProcessor` component was developed to pre-calculate constant parameters for non-bonded interactions. In detail, when analyzing SuperTrAmber's van der Waals term (defined in Section 3.4.1):

$$E_{vdW} = \frac{\varepsilon_{ij} \cdot R_{ij}^{*12}}{R_{ij}^{12}} - \frac{2 \cdot \varepsilon_{ij} \cdot R_{ij}^{*6}}{R_{ij}^{6}} \tag{3}$$

and its hydrogen bond term:

$$E_{Hbond/Coord} = \frac{5 \cdot \varepsilon_{ij} \cdot R_{ij}^{*12}}{R_{ij}^{12}} - \frac{6 \cdot \varepsilon_{ij} \cdot R_{ij}^{*10}}{R_{ij}^{10}} \tag{4}$$

it becomes obvious that the numerators in both terms are constant for a given atom type combination. This bears optimization potential, especially because calculating the square root function in $\varepsilon_{ij} = \sqrt{\varepsilon_i \cdot \varepsilon_j}$ is computationally demanding. The `ParameterProcessor` component thus computes the numerators of Equations (3) and (4) once for all combinations of atom types. Resulting values are stored separately. This permits the non-bonded scoring parameters to point to them. For that, a data structure as depicted in Figure 4.4 is constructed.

4.2.7. MMFF94s implementation. In the course of his Master's project [**195**], Gent developed components for the force field framework that fully implement the MMFF94s force field (see Section 3.4.3). One of the project's main focuses was the proper validation of the implementation with the MMFF94s validation data suite [**196**]. As the MMFF94s model necessitates the dynamic computation of empirical parameters for certain atom type combinations, an adapted dynamic `ParameterProcessor` component had to be designed and implemented.

4.3. Porting the SuperTrAmber Force Field to GPU

To speed up computations, the SuperTrAmber force field (see Section 3.4.1) was ported to GPU. Consequently, all data essential for energy calculations is pre-processed and arranged so the GPU accesses it with maximum speed. Additionally, kernels that compute non-bonded interaction and dihedral energies were tuned for high computational performance.

Section 1.5 introduces basic GPU terminology and details on hardware layout, which constrains the design of algorithms and data structures described in this section. Readers unfamiliar with GPU computing are referred to the Introduction prior to continuing.

4.3.1. Atom and parameter data on GPU. Computing energy values requires atom coordinates and force field parameters. These parameters differ depending on the evaluated energy term. Among these, the non-bonded interaction terms read atom and parameter data most frequently. Therefore, this work optimizes their arrangement for rapid access. Consequently, x-, y-, and z-coordinates of an atom along with its partial charge q are stored in a 16-byte data structure. It constitutes the basic building block of the *coordinate array*. An atom's van der Waals radius R_{vdW} and specific Lennard-Jones potential well-depth parameter ε are stored together in a 8-byte data structure. It is the basic building block of the *parameter array* (see Figure 4.5).

Access to global GPU memory is performed coalesced via reading 32-, 64-, or 128-byte chunks. Splitting atom-related data and storing it in two independent data structures is therefore advantageous. First, it avoids unnecessary memory transfer, as 128 byte is an integral multiple of the size of both basic data structures. Furthermore, it allows accessing atom coordinates without reading all additional parameters and thus, reduces overhead. Lastly, the two arrays provide efficient read access to atom coordinates and parameters. This work ensures this by arranging atom data belonging to the same molecule contiguously. For data of atoms belonging to the same component, this research applies the same arrangement strategy.

Dummy atoms are added to each molecule. Their coordinate and parameter data form a padding at the end of both the coordinate and parameter array (see Figure 4.5). This ensures that the number of atoms in a molecule is an integral multiple of 32. As discussed later, this is helpful when optimizing the intramolecular kernel.

Dihedral data. SuperTrAmber's dihedral energies are computed for all flexible molecules. This requires the coordinates of four atoms i, j, k, and l, their atom type-dependent constant (V_m^{ijkl}) as well as the specific phase shift and the periodicity of the cosine potential (γ_m^{ijkl} and

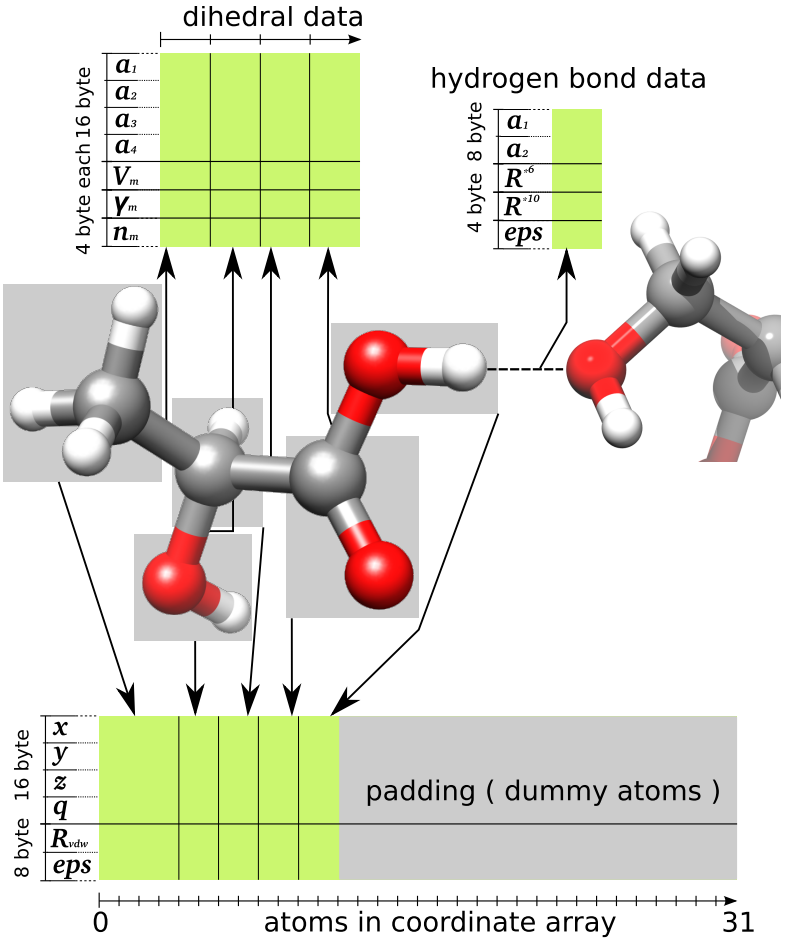


FIGURE 4.5. Storage model for coordinate and parameter data in GPU memory. For each atom, the coordinate array stores coordinates (x,y,z) and the partial charge q. The parameter array comprises the specific well-depth for the van der Waals term (eps/ε) and the van der Waals radius (R_{vdW}). Both data arrays are ordered according to molecular components and end with a padding of dummy atoms. This enlarges each molecule so that its atom count is an integral multiple of 32.
Furthermore, parameters and positional data for dihedral angles and hydrogen bonds are stored. In the dihedral array, $a_1 \ldots a_4$ denote positions of atoms in the coordinate array, V_m is an atom type-dependent force constant, γ_m is the phase shift of the cosine potential, and n_m is the periodicity of the cosine potential. For hydrogen bonds, the sum of the van der Waals radii to the power of six and ten (R_{ij}^{*6} and R_{ij}^{*10}) and the specific well-depth for the van der Waals term (eps/ε) are stored.

n_m^{ijkl}). These parameters are stored in 4-byte memory blocks. In contrast, atom coordinate data is read from the coordinate array. This necessitates storing the coordinate array positions of i, j, k, and l in a block of size 16 byte. This work names the overall data structure comprising

the described dihedral data the *dihedral array*. It is depicted in Figure 4.5, along with data structures discussed in the following paragraphs.

Hydrogen bond data. To calculate hydrogen bond energies, this GPU portation stores the coordinate array positions of two atoms i and j, the sum of their van der Waals radii to the power of six and ten ($R_{ij}^{*^6}$ and $R_{ij}^{*^{10}}$), as well as the square root of their multiplied *eps* values. All of these parameters are stored in 4-byte memory blocks and form the *hydrogen bond array*.

Rescaling matrices. According to the SuperTrAmber model, intramolecular non-bonded 1-4 interactions are scaled down, while 1-3 and 1-2 interactions (see Section 4.2.4) are excluded. Two bit matrices are created to implement these constraints on GPU. For each atom pair, one of the matrices indicates whether the pair forms a 1-4 interaction and the other matrix indicates whether the pair constitutes a 1-3 or 1-2 interaction.

Storing energy values. Energy kernels copy their results to the *global energies array*. The GPU portation initializes this data structure so that it contains at least as many 4-byte storage locations as there are total thread blocks.

4.3.2. Data preparation for non-bonded interactions kernels. Efficient computation of non-bonded interactions on GPU requires preparatory steps. With these, unnecessary data transfer is avoided and irrelevant computations are excluded. This work thus defines the *energy matrix*, a symmetric N-by-N matrix based on a coordinate array of length N. Its entries represent all non-bonded interatomic energy values. The upper left triangular part of the matrix is partitioned into intermolecular and intramolecular interaction energy regions (see Figure 4.6). They are assigned to the intramolecular and the intermolecular energy kernel, respectively. For that, the intramolcular region is divided into tiles of size 1×32 and the intramolecular region into tiles of size 4×32.

This coarse division produces some unfavorable tiles covering both intra- and intermolecular energy values. Furthermore, tiles located at the right border of the matrix refer to undefined segments of the coordinate array. These two issues are circumvented by introducing the previously described padding of dummy atoms (see Figure 4.7).

To assign tiles to thread blocks, a *start index array* is computed for the intra- and intermolecular data. These arrays contain start index pairs (x, y) that specify the upper left corner for each tile with respect to the energy matrix. Thus, x and y also refer to positions in the coordinate array where atom data associated to a tile is stored.

In-component interaction energies remain constant throughout an optimization run. These interactions are therefore excluded by skipping (i.e., not storing) start indices of tiles corresponding to them. For example, the atom data of a component c of length n occupies a contiguous segment from $start_c$ to $start_c + (n-1)$ in the coordinate array. Thus, the GPU portation skips tiles with indices $(start_c \ldots start_c + (n-1), start_c \ldots start_c + (n-1))$. Figure 4.7 shows this concept.

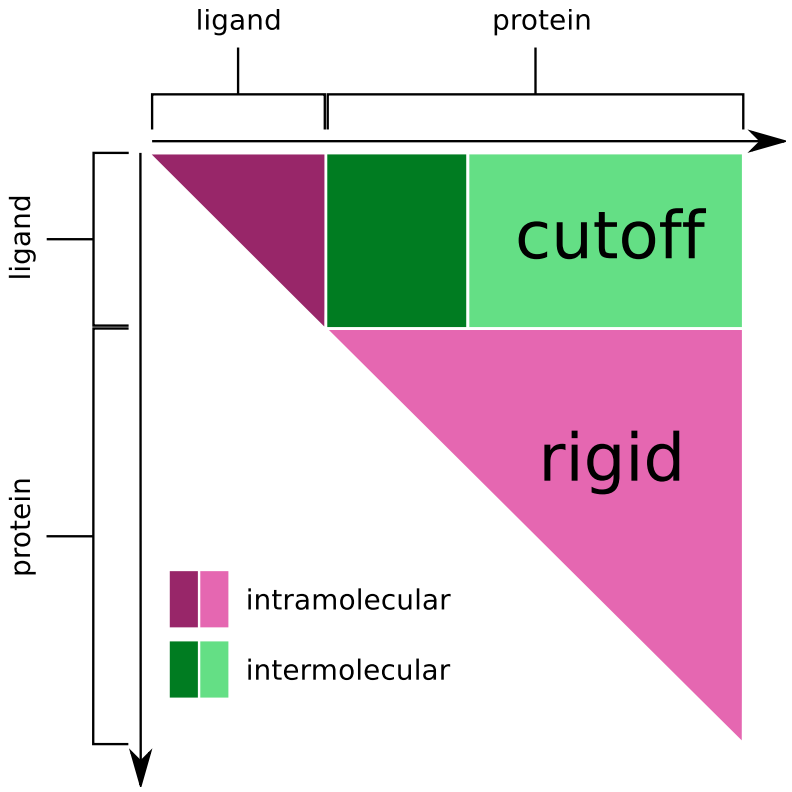


FIGURE 4.6. Simplified $N \times N$ energy value matrix for an exemplary protein-ligand complex. Intramolecular energy values of proteins that are labeled as rigid are ignored. Additionally, atom pairs whose interatomic distance exceeds a cutoff threshold are excluded. Most often, the cutoff block is not as clearly separated from the other entries as in the illustrated scenario.

To further reduce the number of non-bonded interaction tiles, the following introduces the concept of rigidity. The user can label a molecule as rigid. This entails the exclusion of all interactions within rigid molecules (see Figure 4.6).

Cutoffs constitute an additional measure to reduce the computational complexity of non-bonded energy calculations. Based on a given cutoff distance, an algorithm analyzes all tiles to determine whether they contain at least one atom pair whose distance is lower than the cutoff. Should that not be the case, the algorithm excludes that respective tile from the energy calculations by deleting its index from the start indices array, as shown in Figure 4.8.

4.3.3. Non-bonded interactions kernels. Atom and parameter data on GPU is now prepared for energy calculations. The following discusses these starting with the non-bonded interaction kernels that compute van der Waals, electrostatic and hydrogen bond energies.

Intermolecular kernel. Each thread block executing the intermolecular kernel first loads the starting indices (x, y) of its associated tile into the registers of its SM. The threads with IDs

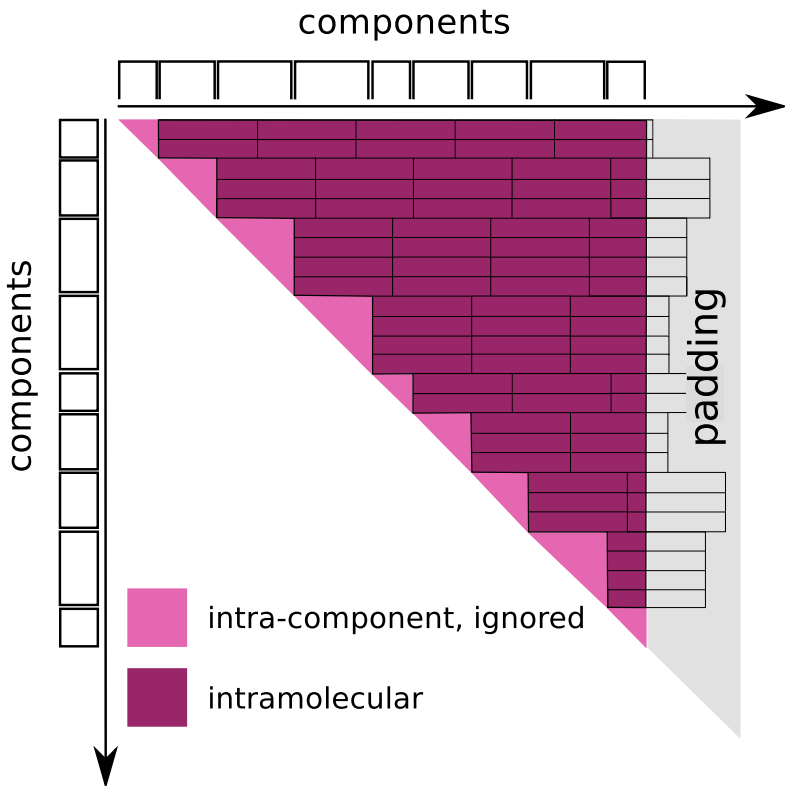


FIGURE 4.7. Region of an energy matrix containing intramolecular energy values. Intracomponent interactions are ignored. The rest of the matrix is split into tiles of size 1×32. Each of these is assigned to one thread block. Contrary to the depiction, one tile often spans several components. Occasionally, a tile's length excels the number of remaining atoms. In these cases, the algorithm processes padding region dummy atoms instead of causing errors by accessing data belonging to the next molecule.

$0 \ldots 31$ then copy the coordinate and parameter array segments with index range $x \ldots x+31$ and $y \ldots y+3$ to shared memory. There, the x-, y-, and z-coordinates, as well as the parameters, are stored in separate arrays. For saving computed energy values, an array of size 4×32 is allocated in shared memory, which is named the *local energies array*.

Based on the copied coordinate and parameter data, each thread computes van der Waals and Coulomb energies for one atom pair. The resulting values are stored in the local energies array at a position according to the ID of the thread.

To compute hydrogen bond energies, the intermolecular kernel is provided with the total number of hydrogen bonds. Each thread determines whether this number is larger than its unique incremental ID. If that is the case, the thread loads the entry from the hydrogen bond array whose position corresponds to its ID. Additionally, the thread loads the van der Waals term parameters for the atoms of its hydrogen bond. With these parameters, the van der Waals

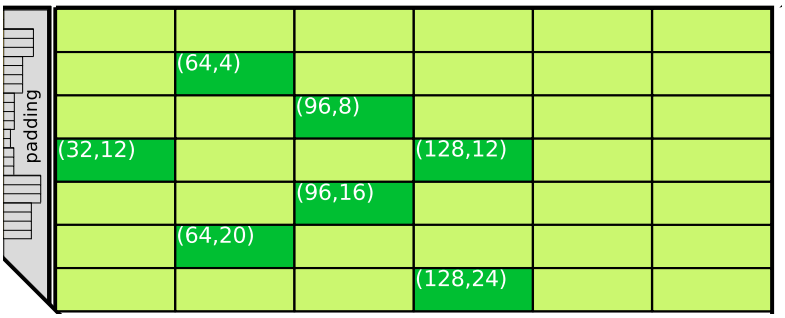


FIGURE 4.8. Indexing scheme for blocks of intermolecular interactions. A tile for intermolecular interactions comprises 4×32 energy values. For all included tiles, the coordinate array indices (x, y) of the atom pair in the tile's upper left corner are stored. Thread blocks read them for identifying their associated atoms $a_x \ldots a_{x+31}$ and $a_y \ldots a_{y+3}$. A tile is excluded if all atom pairs represented by it have distances above the cutoff threshold. Consequently, no energy values are computed for these atom pairs.

energy is computed and then subtracted from the thread's local energies array storage location. Eventually, the hydrogen bond energy is added. Thus, the van der Waals energy of an atom pair is replaced by its hydrogen bond energy.

Finally, the intermolecular kernel reduces all values of the local energies array to a single one with the implementation of the reduction algorithm described in Section 3.3.2. Each thread block copies its final result to its storage location in the global energies array.

Intramolecular kernel. The intramolecular kernel starts just like its intermolecular energies counterpart. As the tile size for intramolecular energies is just 1×32, this kernel copies only coordinate and parameter array segments with index range $x \ldots x + 31$ and y to shared memory.

Subsequently, the relevant tiles of the rescaling bit matrices are copied to shared memory. From these matrices, each thread derives the scaling factor for the van der Waals and Coulomb terms. In case of an 1-4 interaction, this factor is 0.5, in case of an 1-2 or 1-3 interaction, it is 0.0.

Each thread then computes accordingly scaled van der Waals and Coulomb energy for one atom pair. Results are stored in the local energy array, which is allocated for each thread block as previously described for the intermolecular energy kernel. The intramolecular kernel employs the same algorithm as the intermolecular kernel for finally reducing its energy values.

Remarks on efficacy. Harvesting the full computational potential of GPUs requires some deeper insights into their hardware architecture and their memory access model. Using this knowledge, this research optimized the non-bonded interaction kernels at different key points for performance. The storage model allows for accessing coordinates and parameters for one warp perfectly coalesced and without any overhead (see Depiction 4.9). This applies for all GPU compute capabilities, although the number of memory request and transactions differs.

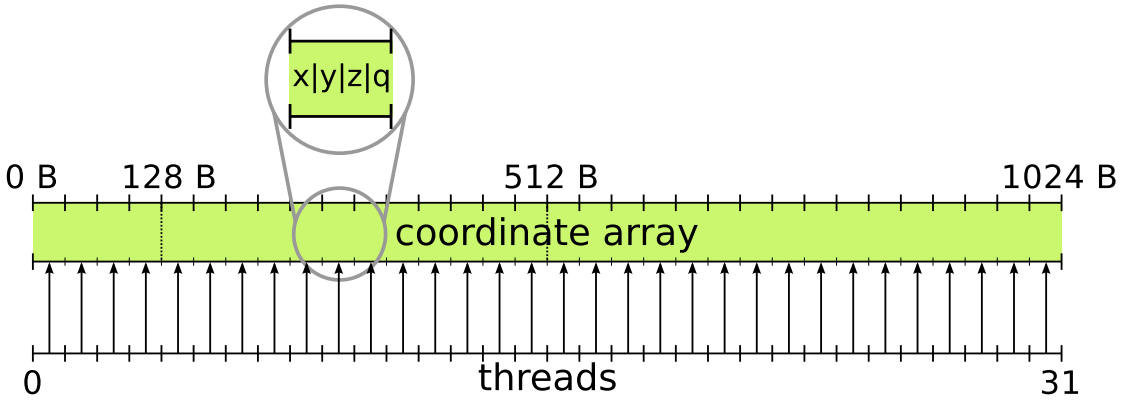


FIGURE 4.9. A warp of thread accesses the coordinate array. The coalesced storage of coordinate and charge data enables warps to read a contiguous memory region whose length is a multiple of the optimal memory transaction size. This maximizes throughput and access speed when reading from the coordinate array.

Furthermore, this GPU algorithm stores data in separate arrays in shared memory—one for x-, y-, and z-coordinates, respectively, and one for each parameters type. This way, the algorithm avoids shared memory bank conflicts when accessing these values.

When reducing a thread block's energy values, the algorithm employs sequential addressing and loop unrolling (see Section 3.3.2) and thus circumvent memory bank conflicts and instruction overhead. Additionally, this work avoids synchronizing the threads of a warp by not allowing for divergent execution branches.

Replacing the `sqrtf` function with its reciprocal counterpart `rsqrtf` yields further performance gain on hardware of the Kepler generation as the Newton-Raphson method [**197**, Ch. 9 p. 456] operates without any division operations.

Finally, rescaling interaction in the intramolecular kernel is coupled to a minimum diversion of control flow. The algorithm achieves this by multiplying the van der Waals and Coulomb terms with a variable scaling factor that each thread modifies depending on its respective rescaling matrix entries.

4.3.4. Dihedral energies kernel. For dihedral energy calculations, thread blocks of size 1×32 are created. Each thread computes the energy of one dihedral angle. Therefore, the kernel accesses the dihedral array to load four positions of atoms in the coordinate array along with three parameters specific to the dihedral angle. Then, atom coordinates are loaded from the coordinate array and the dihedral energy value is computed. It is written to shared memory, reduced on-chip, and thereafter transferred to the global energies array.

4.3.5. Reductions. After having finished all energy calculations, results stored in the global energies array are reduced to a single total energy value. For this, Algorithm 4.4 is employed, which performs reductions on a single SM utilizing only one thread block of size

1×32. Consequently, only a small fraction of the GPU's computational power is utilized. At the same time, synchronizations of GPU multiprocessors are circumvented. For enhancing the efficiency of this implementation, sequential addressing and full loop unrolling, as described in Section 3.3.2 are employed.

Algorithm 4.4 Reduction algorithm for a single SM

```
SingleStepReduce
1  while globalPos < nofValues
2        do for all threads in parallel
3              do tempVals[threadID] ← tempVals[threadID] + energyArray[globalPos]
4                 globalPos ← globalPos + blockSize
5  localReduce(tempVals)
6  return tempVals[0]
```

4.4. Accelerated GPU Algorithm for Flexible Amino Acid Side Chains

To sustain the performance of the GPU algorithm when considering flexible amino acid side chains, several adaptations are necessary. These mainly concern the selection of a flexible molecular region and the intramolecular kernel that copes with at least one order of magnitude more input data.

4.4.1. Altered data preparation. The altered preparatory phase either starts with the selection of a spatial region within the provided molecules or a set of amino acid side chains. Atoms contained in this selection form the set *FlexRegion*. Members of this set are subject to transformations during the minimization procedure (see Section 4.6). A further set, named *OptAtoms*, encompasses all atoms within a radius of 8 $\AA$ around members of *FlexRegion*. An atom a is thus member of *OptAtoms* if there exists an atom $b \in FlexRegion$ for which $distance(a, b) < 8\,\AA$. The following initialization procedure applies to members of *OptAtoms*. All other atoms are ignored by the accelerated GPU algorithm.

All atoms of *OptAtoms* are grouped into components, as defined in Section 4.1.2. Ordered according to their component numbers, the atom data of members of *OptAtoms* are uploaded to the GPU. There, atom coordinates form the *coordinate array*, while atom parameters constitute the *parameter array*. Additionally, the component number of each atom is stored in the *component array*. Furthermore, the coordinate array positions of atom pairs forming hydrogen bonds are stored in the *hydrogen bond array*, while positions of atom pairs with 1-2, 1-3, and 1-4 interactions (see Section 3.4.1) form the *rescaling array*. Figure 4.10 outlines the altered molecular representation.

This representation bears two advantages. First, it completely excludes atoms that are irrelevant to changes in the energy value of the molecular system to be optimized. This allows for a far more efficient cutoff mechanism for intramolecular interactions in proteins. Second, it abolishes the rescaling bit matrices and introduces the rescaling array instead. This consumes

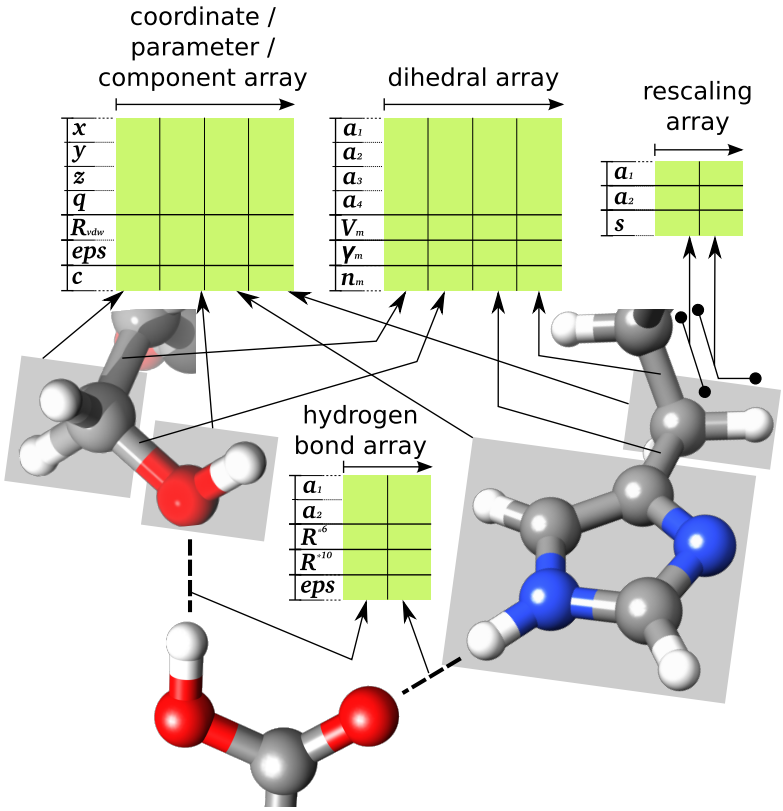


FIGURE 4.10. Adapted storage model for processing intramolecular interactions more efficiently. Most data correspond to those introduced in Figure 4.5. For each atom, the component ID (c) is added to the coordinate array. Furthermore, all intercomponent 1-2, 1-3, and 1-4 interactions are comprised in the newly introduced rescaling array. It consist of the coordinate array positions a_1 and a_2 of interacting atom pairs and the respective factors s for rescaling the non-bonded energy terms. The dihedral and hydrogen bond array remain unchanged.

far less memory and speeds up calculations. Reasons for this are reduced memory transfers and flow of control divergences as explained in the next section.

4.4.2. Kernel for intramolecular energies. To compute intramolecular energies, the intramolecular part of the energy matrix (see Figure 4.6) is split into tiles of size 4×32. Then, a cutoff is imposed by following the procedure described in Section 4.3.2.

When called, the intramolecular kernel copies coordinates, force field parameters, and component IDs of the atoms of its thread block to shared memory. Based on these values, the kernel computes intramolecular van der Waals and Coulomb energies. With the help of the component IDs, calculations of in-component interactions are skipped. After this, each entry of the rescaling array is associated to a thread. Each thread reads the rescaling factor as well as the atom coordinates and force field parameters of its atom pair. With these values, a scaled non-bonded energy

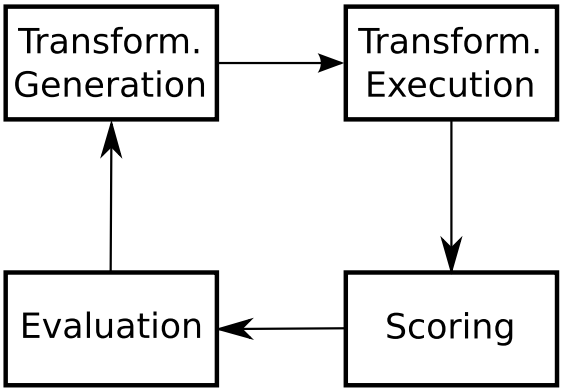


FIGURE 4.11. The optimization cycle: a decomposition of the SA algorithm into functional units. The flow of control is symbolized by arrows.

value is computed for the atom pair. The thread subsequently subtracts this value from its storage position in the local energies array. Thus, 1-2, 1-3, and 1-4 interactions are appropriately rescaled.

Energies of 1-2 interactions tend to be quite high in comparison to other non-bonded interaction energies. As this algorithm operates with single-precision floating-point numbers, this can entail numerical instabilities due to loss of significance accuracy problems. The introduction of Kahan's algorithm (see Section 3.3.3) tackles this issue for reductions and summations within the intramolecular energy kernel.

4.4.3. Reductions. The results stored in the global energies array are reduced to a single value with the `TwoStepReduce` algorithm. In the first step, it employs Algorithm 4.4—the `SingleStepReduce`—to reduce the global energies array with 32 thread blocks of size 8×32 such that all GPU multiprocessors are utilized. The 32 remaining values are subsequently reduced to a single one by utilizing the `SingleStepReduce` algorithm again, this time with one thread block of size 1×32 and thus a single multiprocessor.

4.5. Optimizations for Flexible Ligands in a Rigid Binding Site

For optimizing ligand molecules in rigid binding sites, this research employs a basic SA approach as outlined in Section 3.3.1. It operates according to a fixed exponential cooling schedule and terminates after a user-defined number of steps. Optionally, an early abort-mechanism interrupts the procedure if a given set of conditions is fulfilled.

4.5.1. Optimization cycle. The following decomposes the SA procedure into four functional units, named *Transformation Generation*, *Transformation Execution*, *Scoring*, and *Evaluation*. These units form the *optimization cycle* (Figure 4.11), which reflects the main loop of the SA algorithm.

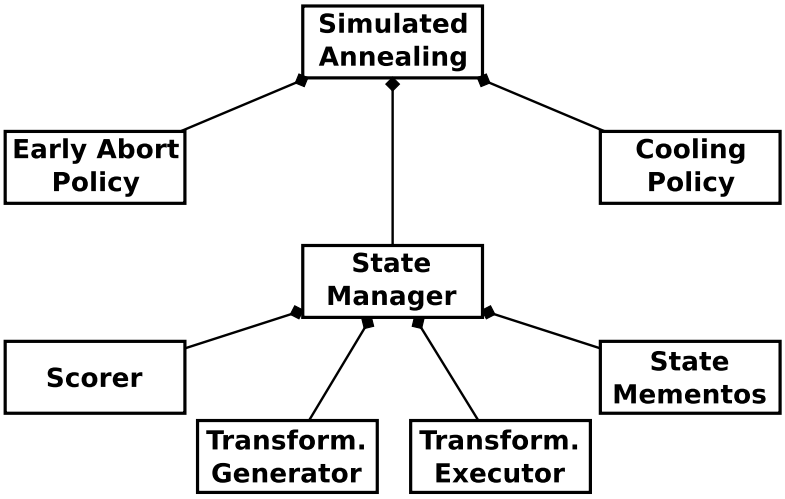


FIGURE 4.12. Exchangeable components of the SA algorithm. The `StateManager`, `CoolingPolicy`, and `EarlyAbortPolicy` components are orthogonal and can thus be altered independently. This controls and changes the behavior of the SA procedure.

In the Transformation Generation Unit, a degree of freedom is selected. Depending on its type, parameters such as rotation angle, rotation axis, and translation vector are generated. Using these parameters, the Transformation Execution Unit subsequently generates a new state after saving the old one. Thereafter, the Scoring Unit computes the value of the objective function for the newly generated state. Based on this value, the Evaluation Unit either accepts or rejects the new state. In the latter case, the cycle continues after having restored the saved previous state. Furthermore, the Evaluation Unit breaks the optimization cycle if the maximal number of steps is exceeded or conditions for an early abort event are fulfilled.

4.5.2. Components forming the optimization. Altering the functionality of units of the optimization cycle changes the overall behavior of the SA procedure. This work groups orthogonal functionality into components with the result that modules arise that are exchangeable without side effects (see Figure 4.12). This design allows for controlling aspects of the behavior of the algorithm according to the strategy pattern **[194]**.

The main `Simulated Annealing` component controls the flow of the SA procedure according to the optimization cycle. It therefore makes use of functionality comprised in the components described next. Furthermore, it contains the acceptance criterion of the Evaluation Unit. In all SA protocols described in this work, a new state is accepted if its energy is lower than that of the previous one or the Metropolis Criterion **[145]** is fulfilled:

$$\Delta E < 0 \vee e^{-\Delta E/T} > Rand[0,1]$$

Acceptance thus depends on the difference between objective function values for the new and the previous state (ΔE), the temperature parameter T, and a sample drawn from a uniform distribution on the closed interval $[0, 1]$.

The `CoolingPolicy` component controls parameters that guide the SA process. In detail, these are the temperature (T), the *cooling factor* (c_f), the number of steps per cooling cycle, and the maximum *total number of optimization steps* (s_{max}).

The `EarlyAbortPolicy` component stores a set of conditions under which the optimization procedure instantly terminates without performing the maximum number of steps.

The `StateManager` subsumes components that allow for altering, storing, and scoring the state of molecules. These components are non-orthogonal, as their interfaces depend on data structures used for molecular representation and transformation operations. The `Scorer` component provides the functionality of the Scoring Unit, the `Transformation Generator` component the functionality of the Transformation Generation Unit. The functionality of the Transformation Execution Unit is split between two components named `Transformation Executor` and `StateMementos`. The former transforms molecules according to parameters provided by the `Transformation Generator` component. The latter saves and restores atom coordinates.

4.5.3. CPU- and GPU-based algorithms. The author designed the SA procedure such that it either operates mainly on GPU or on CPU. The strategy pattern allows for distinguishing the components both algorithms share and those which encapsulate differing implementation aspects.

4.5.4. Common components. The `CoolingPolicy`, `EarlyAbortPolicy`, and `Transformation Generator` components are shared between the CPU- and GPU-based algorithms. Parameters for two cooling policies were derived from optimizations, as described in Sections 6.1.3 and 6.4.2. One policy is designed for local optimizations and the other for global optimizations. The former starts with temperature $T = 0$, while the latter starts with varying temperatures, as described in Section 4.6.3 and in the experimental section, and follows an exponential cooling scheme according to which $T_{new} = T_{old} * c_f$ with $c_f = 0.98$. A cooling event is triggered every 100 steps. Both cooling policies allow for a maximum number of $s_{max} = 3{,}000$ optimization steps.

The shared `EarlyAbortPolicy` component comprises two conditions that, whenever at least one of them is fulfilled, lead to the termination of the optimization. These conditions are the maximum number of consecutively rejected states $ea_{rej_{max}}$ and the minimal absolute energy reduction over the last 10 minimal states $|ea_{red_{min}}|$. These parameters are set to:

$$\begin{aligned} ea_{rej_{max}} &= 200 \\ |ea_{red_{min}}| &= 0.1\ kcal/mol \end{aligned}$$

When initialized, the shared `Transforansformation Generator` component uses the current system time as a seed for its random number generator. This stores which transformation types

are allowed for each of the provided molecules. There are three types of transformations available, namely dihedral rotations, molecular rotations, and molecular translations. The optimization algorithm for protein-ligand complexes with flexible ligands permits all transformation types for ligands and none for all other molecules. The `Transformation Generator` randomly selects one of the flexible molecules and a valid transformation type. Subsequently, it randomly determines parameters for the selected transformation.

For dihedral rotations, these parameters comprise the ID of the rotatable bond, the rotation angle Φ, and the rotation direction. The value of the latter parameter is either `from` or `to`. This refers to the directions stored component tree edges (see Section 4.1.2). This way, it is determined which molecular part adjacent to the rotatable bond is transformed. For molecular rotations, the `Transformation Generator` randomly selects a rotation axis and a rotation angle ρ while for molecular translations a direction vector $\vec{t}$ along with its length $|\vec{t}|$ are determined.

User-provided parameters set the maximum angle for molecular and dihedral rotations (ρ_{max} and Φ_{max}) as well as the maximum length of the translation vector $[max(|\vec{t}|)]$. For all ligand minimizations in rigid binding pockets (excluding parameterization runs), these parameters are fixed:

$$\begin{aligned} \rho_{max} &= \frac{\pi}{48}\, RAD \\ \Phi_{max} &= \frac{\pi}{48}\, RAD \\ max(|\vec{t}|) &= 0.1\, \mathring{A} \end{aligned}$$

4.5.5. Differing components. As described in Section 4.3, this work implemented the SuperTrAmber force field on GPU. Thus, the `Scorer` component differs in the CPU and GPU optimization procedure. Additionally, the `Transformation Executor` component has architecture-specific implementations.

4.5.6. Components of the CPU-based algorithm. The author designed a basic CPU-based `StateManager` component. Exchanging its `Scorer` component allows for switching between three objective functions as illustrated in Figure 4.13. Namely, these are the MMFF94s force field (see Section 3.4.3) and two flavors of the SuperTrAmber force field. These differ in the way hydrogen bonds are scored. Either the default model is used or hydrogen bonds are emphasized, as described in Sections 3.4.1 and 3.4.2.

The `Scorer` components compute the energy of a given combination of molecules using the Coulomb, van der Waals, hydrogen bond, and dihedral energy terms of their respective force field. To speed up calculations, a cutoff of $12\,\mathring{A}$ is imposed on non-bonded interactions during the initialization phase. To further reduce the number of non-bonded interactions, proteins and their cofactors are labeled as rigid. Consequently, all interactions within these molecules are excluded. Furthermore, the scoring component utilizes the reduced dihedral angle detection method described in Section 4.2.4 and thus computes dihedral energies only for rotatable bonds that are represented by an edge of the component tree.

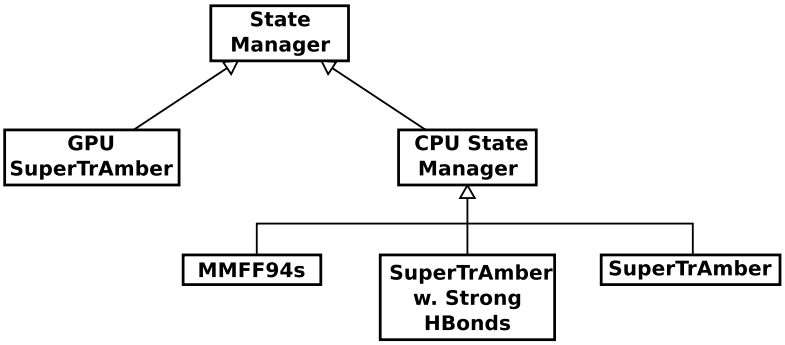


FIGURE 4.13. Overview on available state managers. All state manager derive their interface from the basic `StateManager` component and provide an objective function for the SA procedure. Four state managers are available that provide access to the MMFF94s force field, the accelerated GPU-based implementation of the SuperTrAmber force field, and two CPU-based SuperTrAmber flavors of which one emphasizes hydrogen bond energies.

The `Scorer` components employs the partial charge model of their respective force field. Thus, SuperTrAmber-based scorers assigns AMBER3 charges (see Section 3.6.3) to proteins and Gasteiger-Marsili charges (see Section 3.6.1) to all other molecules. In contrast, MMFF94s-based scorers assign partial charges according to the MMFF94 model (see Section 3.6.2) to all types of molecules.

The `Transformation Executor` component computes a 4×4 affine transformation matrix based on parameters generated by the `Transformation Generator` component. This transformation matrix is multiplied with all atom coordinates of a selected molecule in case of global rotations and translations. In case of dihedral rotations, the `Transformation Executor` utilizes the component tree to obtain all atoms in `from` or `to` direction of a selected rotatable bond. Coordinates of thus identified atoms are subsequently transformed.

The `StateMementos` component stores atom coordinates for all non-rigid molecules. This way, it saves three states of molecular systems: the current, last, and best state. Molecule coordinates are always in the current state, which resulted from transforming the last state. The best state is the energetically most favorable one yet detected. The interface of the `StateMementos` component allows for saving the last or the best state and for restoring the best state. Consequently, the `StateMementos` component is unaware of the meaning of the single states.

4.5.7. Components of the GPU-based algorithm. To accelerate the optimization procedure, the `Scorer` component is relocated to GPU. This is clearly the most promising measure as the scoring functionality consumes most of the runtime. However, force field-based scoring requires atom coordinates, force field parameter data and molecular connectivity information. The algorithm uploads all these data in advance to avoid data traffic between host system and device. This, in turn, necessitates porting the `Transformation Executor` component to GPU.

Consequently, the management of molecular states and thus the `StateMementos` component has to be ported to GPU as well. In short, a `StateManager` component for GPUs is required.

The GPU-based `StateManager` component assigns partial charges to proteins according to the AMBER3 model, which is introduced in Section 3.6.3. For all other types of molecules, the Gasteiger-Marsili model as described in Section 3.6.1 is employed. Furthermore, the GPU state manager sets the cutoff value to $12\,\mathring{A}$. Then, it calls the data preparation of SuperTrAmber's GPU implementation as described in Section 4.3.1. This functionality is encompassed by the GPU-specific `Scorer` component. The main task of this component is calling the dihedral and non-bonded interaction kernels, which are described in Sections 4.3.4 and 4.3.3 and then calling the reduction kernel described in Section 4.3.5. When finished, the `Scorer` component returns a single-precision floating-point energy value, which is transferred back to the host.

The GPU-specific `Transformation Executor` component operates on the coordinate array (see Section 4.3.1) and executes transformations for which the `Transformation Generator` component creates parameters on CPU. To globally rotate molecules, corresponding parameters are uploaded to GPU. There, the targeted segment of the coordinate array is divided into tiles of size 1×32. For each of these, a thread block computes an affine transformation matrix and applies it to the coordinates of its assigned segment.

For the translation kernel, the relevant segment of the coordinate array is again divided into tiles of size 1×32. Each thread block then simply adds the uploaded translation vector to the atom coordinates of its assigned segment.

For each rotatable bond, there is a fixed set of components affected by a dihedral rotation. These components occupy contiguous segments of the coordinate array. Their positions and lengths are stored on GPU. This allows for quick access when a dihedral rotation is requested. To perform a rotation, each affected component is assigned to a thread block of size 1×32. A thread block then loads the coordinate array position and length of its segment. Subsequently, it constructs an affine transformation matrix and applies it to the coordinates of its segment.

The `StateMementos` component stores atom coordinates of the last and the best state in separate coordinate arrays on GPU. To avoid unnecessary kernel calls, copy operations for generating and restoring states are handled in the transformation kernels. Consequently, they operate on the coordinate array for the last state whenever a restore operation is requested and on the current coordinate array otherwise. The kernels apply their respective transformations and save the resulting coordinates to the coordinate array for the current state. This way, restoring and transforming coordinates is handled at the same time.

4.5.8. A modified optimization cycle for GPU-based optimizations. The described modifications for porting the optimization algorithm to GPU result in an altered optimization cycle as depicted in Figure 4.14. With this cycle, the computational power of the GPU is harnessed while data transfer is kept at merely 164 byte per optimization cycle.

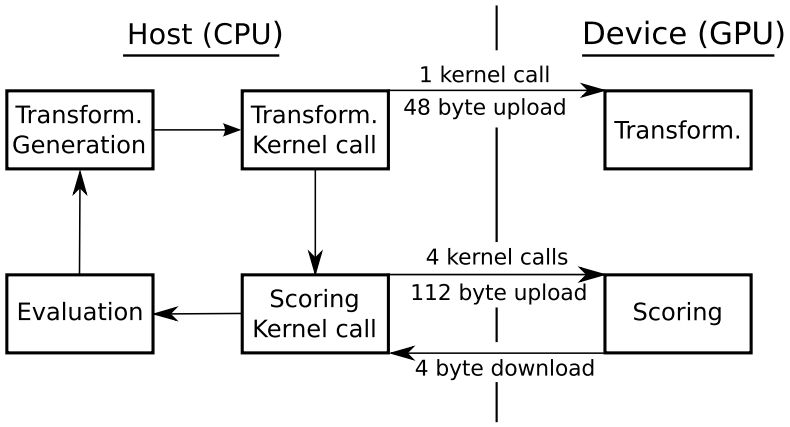


FIGURE 4.14. Modified optimization cycle for including GPU-based scoring into the SA procedure. Scoring and all operations on atom coordinates are conducted on GPU. This approach leads to sparse data exchange between the GPU device and its host system. Only 4 byte of data are downloaded from the device and 160 byte are uploaded to it per optimization cycle. Furthermore, the number of kernel calls is kept at a minimum by integrating intermolecular and hydrogen bond energy computations into a single kernel. The computation of dihedral and intramolecular energies require one kernel call, respectively. This also applies to the reduction of all calculated energy values. All in all, the force field energy of protein-ligand complexes is computed with four kernel calls. Restoring, saving, and transforming is handled in a single kernel.

4.6. Extension for Flexible Side Chains

The following describes adaptations to the optimization procedure that permit handling flexible amino acid side chains. As an additional step, a set of these are selected. Therefore, the user either provides a list of amino acids encoded in three letter code or a reference ligand and a radius r. In the latter case, all amino acid side chains that contain atoms located within at least one sphere with radius r around a ligand atom are considered flexible.

4.6.1. Adaptations to SA protocol. Rotatable amino acid side chains entail a larger number of degrees of freedom. To cope with these, the temperature parameter T of the SA algorithm is coupled with the energy deviations ΔE, which result from molecular transformations. Furthermore, parameters of the `Transformation Generator` component, which determine the magnitude of molecular transformations, are periodically scaled down. These changes entail the introduction of a `JournalPolicy` component and modifications to the `Transformation Generator`, `CoolingPolicy`, `Scorer`, and `EarlyAbortPolicy` components.

4.6.2. Optimization procedure. With the newly introduced parameters and modified components, the SA procedure—now named `Adaptive SimulatedAnnealing`—adapts the temperature at the end of each cooling cycle. For this, the median energy deviation value ΔE_{med} is utilized to determine the temperature T_{new}, which entails an average acceptance rate of 0.5

according to the Metropolis criterion:

$$T_{new} = \frac{\Delta E_{med}}{log(accRate_{des})}$$

Furthermore, the transformation parameters ρ_{max}, Φ_{max}, and $max(|\vec{t}|)$ are rescaled with the transformation downscaling factor $rescale_{trans}$. The newly introduced parameters and modified components are discussed in the following.

Algorithm 4.5 Adaptive SimulatedAnnealing

```
AdaptiveSimulatedAnnealing
 1  nofSteps ← 0
 2  transParameters ← initTransParameters()
 3  lastScore ← scoreCurrentState()
 4  bestScore ← lastScore
 5  nofStepsPerCycle ← getConstantNofStepsPerCycle()
 6  while !isCoolScheduleAbort(currTemp) and !isEarlyAbort(lastScore)
 7        do ▷ periodically update temperature and transformation parameters
 8           if nofSteps  mod  nofStepsPerCycle = 0
 9              then currTemp ← adaptTemperature()
10                   transParameters ← rescaleTransParameters()
11           ▷ update and evaluate state of the system under optimization
12           generateNextState(transParameters)
13           currScore ← scoreCurrentState()
14           nofSteps ← nofSteps +1
15           if !isAccepted(currScore, lastScore, currTemp)
16              then restorePreviousState()
17                   continue
18           lastScore ← currScore
19           if isBetterThan(currScore, bestScore)
20              then bestScore ← currScore
21                   saveCurrentStateAsBest()
22  restoreBestState()
23  return scoreCurrentState()
```

4.6.3. Cooling policy. The `CoolingPolicy` component is altered by setting the maximum number of steps in a complete optimization procedure s_{max} to 15,000. Furthermore the initial value for the temperature T is set to 30 and the number of steps per cooling cycle to 100. Two new parameters are introduced. First, the transformation downscaling factor $rescale_{trans} = 0.97$ and, second, the desired acceptance rate $accRate_{des} = 0.5$.

4.6.4. Journal policy. The `SimulatedAnnealing` component depicted in Figure 4.12 now encompasses an additional subcomponent: the `JournalPolicy`. It keeps track of the energy, temperature, and acceptance rate values during the optimization. Consequently, it also provides median ΔE values, which are used to determine new temperatures T_{new}, as described previously.

4.6.5. Early abort policy. The `EarlyAbortPolicy` component undergoes only slight modifications. Its maximum number of consecutively rejected states parameter ($ea_{rej_{max}}$) stays constant and is thus set to 200. Conversely, the minimal absolute energy reduction over the last 10 minimal states parameter ($|ea_{red_{min}}|$) is altered and set to 0.2 $kcal/mol$.

4.6.6. Transformation generator. The parameter set utilized by the `Transformation Generator` component is adapted. It still comprises the maximum angle for molecular and dihedral rotations (ρ_{max} and Φ_{max}), as well as the maximum length of the translation vector $[max(|\vec{t}|)]$. In all experiments but the parameterization phase, the initial values of these parameters are fixed to:

$$\begin{aligned} \rho_{max} &= \frac{\pi}{16}\, RAD \\ \Phi_{max} &= \frac{\pi}{2}\, RAD \\ max(|\vec{t}|) &= 0.2\, \mathring{A} \end{aligned}$$

An interface for rescaling these parameters with the $rescale_{trans}$ factor is provided.

During the initialization of the `Transformation Generator` component, the total number of degrees of freedom is now computed. For proteins with flexible side chains this number is equal to the number of rotatable dihedrals ($\#rotatable\_dihedrals$). For fully flexible molecules, the six degrees of freedom for rotations and translations are additionally considered. The total number thus amounts to $(6 + \#rotatable\_dihedrals)$. During the optimization, degrees of freedom are randomly selected for performing transformation operations.

4.6.7. Scoring components. In all `Scorer` components, the cutoff on non-bonded interactions is reduced to 8 $\mathring{A}$. Furthermore, non-bonded interactions between rigid parts of molecules are excluded, which mainly affects rigid protein regions and cofactors.

This work employs the altered algorithm described in Section 4.4 for scoring on GPU. The flexible region *FlexRegion*, described in this section is determined by applying the procedure specified in the introductory paragraph to Section 4.6. The advanced GPU algorithm enhances the number of kernel calls to five for the scoring functionality in the optimization cycle (see Figure 4.14). At the same time, the amount of data uploaded remains constant. These effects are mainly due to a second kernel call for performing reductions, as introduced in Section 4.4.3.

4.7. Introducing Constraints

By modifying the `Transformations Generator` component, this research introduces constraints to the optimization procedure. This approach is centered around limiting dihedral rotations and atom translations.

4.7.1. Constraining dihedral rotations. For constraining dihedral rotations, quadruples of atoms that form a dihedral angle and have a rotatable central bond can be associated to a list of legal angle intervals (see Figure 4.15). This list is constructed from internal data structures

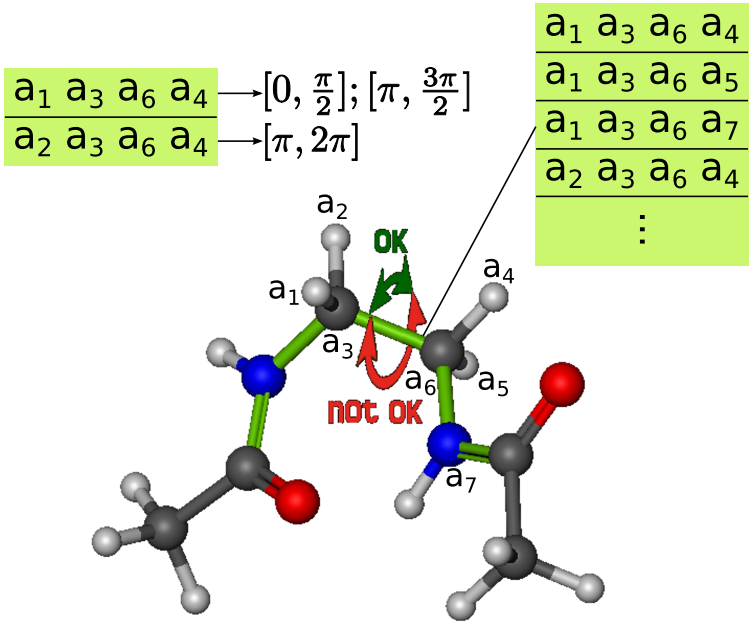


FIGURE 4.15. Limiting rotation angle intervals for quadruples of atoms forming dihedral angles. The quadruples are associated to their respective rotatable central bond. This facilitates fast access when checking constraints.

of CONFECT (see Section 3.8.5). When the `Transformation Generator` component computes a dihedral rotation, all corresponding angle constraints are looked up and checked. In case of a constraint violation, an alternative dihedral rotation is generated and the constraint check is repeated.

4.7.2. Spatial constraints. The `StateManager` component allows registering radial distance constraints for the atoms of a molecule. For this, the initial coordinates of the selected atoms are saved. Whenever the `Transformation Generator` component produces a transformation affecting these atoms, it checks whether any of the distance constraints is violated. In that case, the generated transformation is rejected and a new movement is generated.

4.8. Trooper and GPUperTrooper

Trooper and GPUperTrooper employ the methods introduced in this chapter for small molecule and protein binding pocket optimizations. They operate in two distinct modes: the *local minimization mode* or *local parameterization* and the *global minimization mode* or *global parameterization*. The first one is used for minimizing small molecules in rigid protein binding sites. In contrast, the second mode is employed whenever amino acid side chain conformations are also optimized.

TABLE 4.1. Default parameterization of the basic SA procedure for minimizing ligands in rigid binding pockets.

parameter name	symbol	value	section
objective function	-	SuperTrAmber	3.4.1
maximum number of steps	s_{max}	3,000	4.5.4
steps per cycle	-	100	4.5.4
initial temperature	T	0	4.5.4
cooling factor	c_f	0.98	4.5.4
maximum angle for molecular rotations	ρ_{max}	$\frac{\pi}{48}\ RAD$	4.5.4
maximum angle for dihedral rotations	Φ_{max}	$\frac{\pi}{48}\ RAD$	4.5.4
maximum length of translation vector	$max(\lvert\vec{t}\rvert)$	$0.1\,\mathring{A}$	4.5.4
maximum number of rejected states	$ea_{rej_{max}}$	200	4.5.4
minimal absolute energy reduction	$\lvert ea_{red_{min}}\rvert$	$0.1\ kcal/mol$	4.5.4
cutoff	-	$12\,\mathring{A}$	4.5.5

4.8.1. Local minimization mode. Trooper's and GPUperTrooper's local minimization mode use the basic SA algorithm introduced in Section 4.5. Hence, the optimization procedure follows a predefined exponential cooling scheme. Furthermore, GPUperTrooper scores molecules with the basic GPU portation of SuperTrAmber, which Section 4.3 introduces. Table 4.1 provides an overview on all parameters defining the local minimization mode. They are introduced in detail in the preceding sections and experimentally derived in Section 6.1.3.

4.8.2. Global minimization mode. In the global minimization mode, Trooper and GPUperTrooper are based on the `Adaptive SimulatedAnnealing` algorithm, which Section 4.6 introduces. Furthermore, GPUperTrooper uses the advanced GPU portation of SuperTrAmber presented in Section 4.4. Table 4.2 provides an overview on all further relevant parameters of the global mode. They are introduced in detail in the preceding sections and experimentally derived in Sections 6.4.2 and 6.6.

4.9. Binding Site Manipulation

4.9.1. ActiveSiteWobbler. To apply random transformations to molecules in binding sites, the author developed the `ActiveSiteWobbler`. It employs the previously introduced `Transformation Generator` and `Transformation Executor` components. With these, global molecule rotations and translations, as well as dihedral rotations are performed. Thus, the `ActiveSiteWobbler` operates on the same degrees of freedom as the optimization method.

To utilize the `ActiveSiteWobbler`, the user provides molecules containing binding sites. In these, flexible molecules and regions must to be selected with the procedure described in Section 4.6. Then, a user-defined number of transformations are randomly generated and executed. For these, the maximum angle for molecular and dihedral rotations (ρ_{max} and Φ_{max}), as well as

TABLE 4.2. Default parameterization of the adaptive SA procedure for minimizing flexible ligands and amino acid side chains.

parameter name	symbol	value	section
objective function	-	SuperTrAmber	3.4.1
maximum number of steps	s_{max}	15,000	4.6.3
steps per cycle	-	100	4.6.3
initial temperature	T	30	4.6.3
transformation downscaling factor	$rescale_{trans}$	0.97	4.6.3
desired acceptance rate	$accRate_{des}$	0.5	4.6.3
maximum angle for molecular rotations	ρ_{max}	$\frac{\pi}{16}\ RAD$	4.6.6
maximum angle for dihedral rotations	Φ_{max}	$\frac{\pi}{2}\ RAD$	4.6.6
maximum length of translation vector	$max(\lvert\vec{t}\rvert)$	$0.2\,\mathring{A}$	4.6.6
maximum number of rejected states	$ea_{rej_{max}}$	200	4.6.5
minimal absolute energy reduction	$\lvert ea_{red_{min}}\rvert$	$0.2\ kcal/mol$	4.6.5
cutoff	-	$8.0\,\mathring{A}$	4.6.7

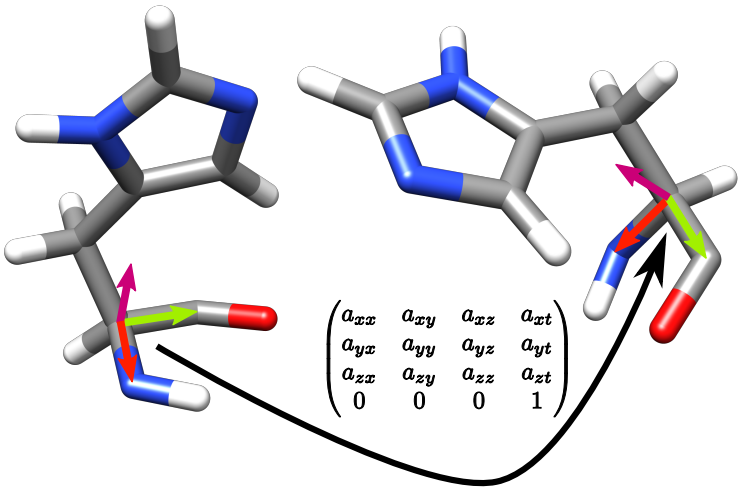


FIGURE 4.16. The `ConformationPorter` transfers side chain conformations between matching residues. This operation employs affine transformation matrices. These are constructed by matching fixed points on protein backbones.

the maximum length of the translation vector $[max(|\vec{t}|)]$, are set to:

$$\begin{aligned} \rho_{max} &= \frac{\pi}{48}\,rad \\ \Phi_{max} &= \frac{\pi}{48}\,rad \\ max(|\vec{t}|) &= 0.1\,\mathring{A} \end{aligned}$$

The `ActiveSiteWobbler` finally saves proteins to `PDB` [**167**] or an internal database format and small molecules to `mol2` format [**168**].

4.9.2. Transferring side chain conformations. The `ConformationPorter` transfers side chain conformations between binding pockets. Therefore, it constructs planes formed by backbone C_α, C, and N atoms for selected pairs of source and target residues. Then, normals to the planes are computed. This procedure yields four points that describe the backbone conformations of source and target residues. The `ConformationPorter` computes transformations between these point sets and applies them to the atom coordinates of selected side chains. In this way, side chain conformations are transferred from source to target residues. The process is shown in Figure 4.16.

CHAPTER 5

Data Sets and Machines

5.1. Astex Diverse-Set

For a large part of experiments in this work, data sets formed of protein-ligand complexes of the Astex Diverse Set [**3**] are utilized. The following introduces them in detail.

5.1.1. Selected docking poses. The experimental section describes how optimizations for flexible ligands in rigid binding pockets are parameterized, the runtime behavior of the SuperTrAmber GPU implementation is tested, and the speedup of the GPU-based optimization procedure is measured. For these tasks, subsets of the Astex Diverse Set were used. This set includes pre-assigned protonations for proteins according to the ChemScore [**54**] model. They were taken as provided. At the same time, water molecules were removed. As proteins were kept rigid, their conformations were taken directly from the data set. In contrast, ligand conformations resulted from docking runs performed with the docking tool TrixX, which Section 3.8.3 introduces.

ADS I. This set comprises 13,255 ligand poses. All in all, there is at least one pose for 83 of the 85 targets in the Astex Diverse Set. Among these is at least one correct docking pose (term defined in Section 3.2) for 61 and at least one excellent docking pose for 22 protein targets. No poses exist for targets tryptophane synthase (*1k3u*) and prostaglandin H2 synthase 1 (*1q4g*).

ADS II. This set is a subset of ADS I. It comprises 125 ligand poses with an RMSD between 1.5 $\mathring{A}$ and 3.0 $\mathring{A}$ to the respective crystal structure. The average RMSD is 2.22 $\mathring{A}$. ADS II contains poses for 63 of the 85 protein targets of the Astex Diverse Set.

ADS III. This is another subset of ADS I which is disjointed from ADS II. ADS III comprises 67 ligands poses and covers 60 protein targets of the Astex Diverse Set. Again, the RMSDs of the ligand poses range between 1.5 $\mathring{A}$ and 3.0 $\mathring{A}$. the average RMSD is 2.35 $\mathring{A}$.

ADS IV. This set is a further subset of ADS I and encompasses 15 protein-ligand complexes with one randomly selected docking pose as ligand. Table 5.1 provides a detailed list of the complexes and the number of atoms in their binding pockets.

ADS V. The fifth subset of ADS I. Its members are specified in Section 6.1.8.

TABLE 5.1. Protein targets comprised in data sets ADS IV and ADS VI, marked by an X. For set ADS IV, the numbers of ligand atoms and binding site atoms within 12.0 $\mathring{A}$ of the ligand are listed.

pdb code	name	ADS IV	#ligand atoms	total #binding site atoms	ADS VI
1g9v	deoxy hemoglobin	-	-	-	X
1gkc	matrix metalloprotease 9	-	-	-	X
1gm8	penicillin G acylase	-	-	-	X
1hnn	phenylethanolamine N-methyltransferase	-	-	-	X
1hq2	6-hydroxymethyl-7,8-dihydropterin pyrophosphokinase	X	23	974	X
1j3j	dihydrofolate reductase	-	-	-	X
1jd0	carbonic anhydrase XII	X	19	1,161	-
1jje	metallo β-lactamase	X	41	1,000	-
1jla	HIV-1 reverse transcriptase	-	-	-	X
1mmv	neuronal nitric-oxide synthase	-	-	-	X
1mzc	protein farnesyltransferase	X	69	1,707	-
1nav	thyroid hormone receptor α1	X	38	1,459	X
1of6	DAHP synthase	X	24	1,185	-
1p2y	cytochrome P450cam	X	27	1,294	-
1r1h	neprilysin	X	56	1,655	-
1s19	vitamin D nuclear receptor	X	70	1,821	-
1sj0	estrogen receptor α	X	63	871	-
1t40	aldose reductase	X	37	1,244	-
1tow	adipocyte fatty acid-binding protein	-	-	-	X
1u1c	uridine phosphorylase	X	36	1,444	X
1v0p	protein kinase 5	X	54	1,294	-
1ygc	factor VIIa	X	67	1,453	-
2br1	Chk1	X	50	1,160	-

5.1.2. ADS VI. As opposed to the previously introduced sets, ADS VI is not a subset of ADS I. Instead, it is formed of 11 proteins of the Astex Diverse Set. Their total atom count covers the range from 2,058 to 22,434. All members are detailed in Table 5.1.

5.1.3. ACS challenge modifications. Section 6.1.7 describes a cognate docking experiment that evaluates the optimization method for flexible ligands in rigid binding sites. For this experiment, a revised version of the Astex Diverse Set, referred to as *ADS VII*, is processed. It was compiled for the Docking and Scoring Symposium, which took place during the 241st ACS National Meeting. The goal of this Symposium was to evaluate the performance of current docking and scoring methods. The most relevant research groups in the field of docking participated

in the Symposium. Their results were published in 2012 in the 6^{th} issue of volume 26 of the Journal of Computer-Aided Molecular Design.

Schneider et al. [**101**] comprehensively describe the preparation process that resulted in the revised version of the Astex Diverse Set. This set includes all ligands to be found in crystal structures of the original set. Therefore, the number of distinct binding pockets increased from the originally 85 to 151.

5.1.4. Wobbled side chains. Starting from Section 6.4, this work parameterizes and evaluates optimizations for binding sites with flexible side chains in the presence of a rigid ligand. For this, this research constructed a set of binding sites named *ADS VIII*. It encompasses modified complexes of the Astex Diverse Set. To create it, water molecules were removed and dihedral angles of amino acid side chains were randomly rotated. In detail, this procedure defined all amino acid side chains containing atoms within a radius of 4.0 $\mathring{A}$ around atoms of native ligands as flexible. Then, the `ActiveSiteWobbler`, introduced in Section 4.9, randomly changed dihedral angles of rotatable bonds. For each protein-ligand complex, the `ActiveSiteWobbler` produced 11 wobbled binding sites by performing 0, 1,000, 5,000, 10,000, 20,000, 30,000, 40,000, 50,000, 100,000, 250,000, and 500,000 random transformations. To quantify their effect, the RMSD (see Section 3.2) of the atoms of the wobbled side chains to the respective atoms in the crystal structure was computed. Figure 5.1 shows the resulting deviations.

Protonations for this data set were generated with the Naomi library introduced in Section 3.5.4. From this process, some histidines with an erroneous protonation at the δ- and ε-nitrogens of the imidazole resulted. Binding sites containing these were excluded. Consequently, data set ADS VIII overall encompasses 748 protein-ligand complexes.

5.1.5. Wobbled binding sites. With the `ActiveSiteWobbler`, the author created *ADS IX*, another wobbled data set based on the Astex Diverse Set. This work utilizes it for parameterizing and validating optimizations of flexible ligands in binding sites with flexible amino acid side chains in Section 6.6. For this, this research created 330 protein-ligand complexes by introducing 0, 500, 1,000, and 2,000 random transformations to the members of the Astex Diverse Set. The definition of the flexible region was the same as in the last section. This time, the ligand was subject to the wobbling procedure as well. Deviations of ligands from their respective crystal conformation are shown in Figure 5.2.

5.2. Astex Non-Native Data Set

For evaluating rigid ligand-flexible side chain optimizations, this work utilized an unmodified version of the Astex Non-Native data set [**198**]. All in all, this set comprises 1,112 protein-ligand complexes. For constructing it, binding sites sequentially identical to any of those in the Astex Diverse Set were gathered from the PDB [**53**]. Matching *non-native* binding sites were superposed with the corresponding *template* structure from the Astex Diverse Set. In addition to this, protonation states were adjusted to that of the respective template structure. The following names the unmodified Astex Non-Native Set *NN I*.

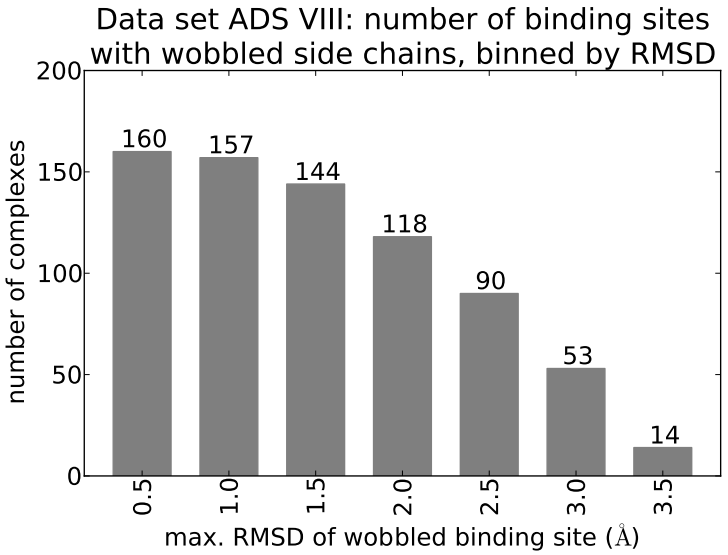


FIGURE 5.1. RMSDs of wobbled binding sites of the ADS VIII data set to the respective crystal structures. Binning scheme corresponds to RMSD limits plotted on the x-axis where labels to the left of a bin constitute the respective lower limit. Structures with binding site RMSDs above 3.5 $\mathring{A}$ were not considered in this depiction.

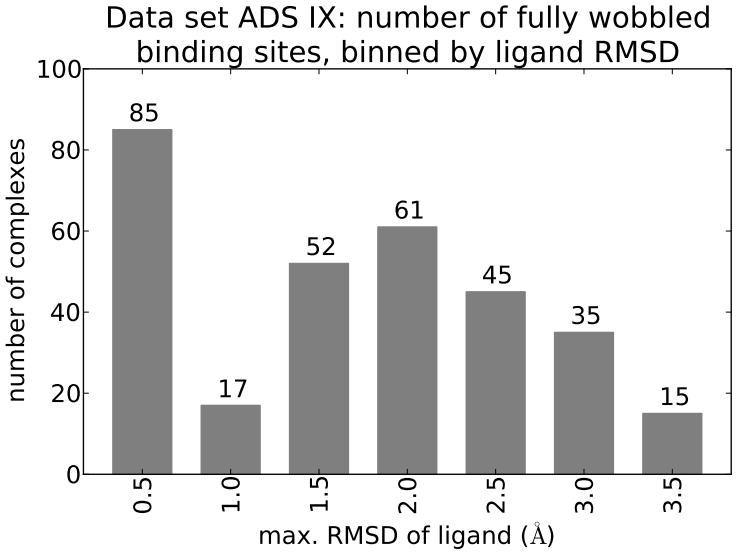


FIGURE 5.2. RMSDs to the respective crystal structures of wobbled ligands of the ADS IX data set. Binning scheme corresponds to RMSD limits plotted on the x-axis where labels to the left of a bin constitute the respective lower limit. Structures with RMSDs above 3.5 $\mathring{A}$ are excluded in this depiction.

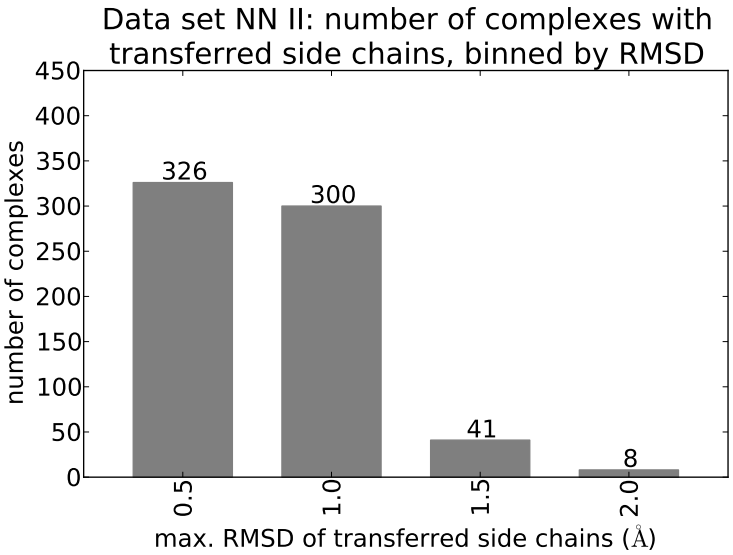


FIGURE 5.3. RMSDs of transferred amino acid side chains of the NN II data set to the respective crystal structures. Binning scheme corresponds to RMSD limits plotted on the x-axis where labels to the left of a bin constitute the respective lower limit. Structures with side chain RMSDs above 2.0 $\AA$ were not considered in this depiction.

5.2.1. Transferred side chains. The author created data set *NN II* by modifying the Astex Non-Native Set. With the `ConformationPorter`, which Section 4.9 describes, the research transferred binding site amino acid side chain conformations from non-native structures to the backbone of the respective template structure. All side chains having atoms closer than 8.0 $\AA$ to any atom of the respective native ligand were transferred. During this process water molecules were removed and protonation states were adapted with the Naomi library introduced in Section 3.5.4. Again, some histidines with an erroneous protonation at the δ- and ε- nitrogens of the imidazole resulted. Binding sites containing these were excluded. This method produced 686 distinct complexes. For these, Figure 5.3 shows deviations of transferred side chains from the template structure conformation. It is noteworthy that the Astex Non-Native Data Set is biased towards certain proteins. For example, there are overall 373 conformations of HIV-1 protease, thrombin, and carbonic anhydrase II.

5.3. CONFECT Test Data Set

For testing downstream optimizations in conjunction with CONFECT, this work applied this tool to generate a set of 4,997 small ligand conformations. They were constructed from a subset of the CSD [**184**], which encompasses 75 ligands. These, in turn, were extracted from protein-ligand complexes of the PDB [**53**].

TABLE 5.2. Machines used for benchmark experiments. The OpenSUSE operating system (*OS*) was installed on all systems. Different versions of the `gcc` and `nvcc` were used.

Machine	CPU type ID (Table 5.3)	GPU type ID (Table 5.4)	RAM (GByte)	OS	gcc	nvcc (gcc)
I	1	1	16	11.3	4.5.0	4.0 (4.4.5)
II	2	2	8	12.2	4.7.1	5.0 (4.4.5)
III	3	-	128	12.2	4.7.1	-
IV	3	3	128	12.2	4.7.1	5.5 (4.7.1)
V	4	4	12	11.3	4.5.0	4.2 (4.5.0)

TABLE 5.3. Processors of benchmarking systems. Their TDP values (see Section 3.1) are specified along with their release dates.

ID	Intel CPU	GHz	#cores	TDP (W)	launch quarter ($Q/YYYY$)	ref.
1	Xeon E5420	2.5	4	80	4/2007	[**199**]
2	Xeon E5-2609	2.4	4	80	1/2012	[**200**]
3	Xeon E5-2680	3.5	8	130	1/2012	[**201**]
4	Xeon E5630	2.66	4	80	1/2010	[**202**]

TABLE 5.4. Graphics processors of benchmarking systems. Their TDP values (see Section 3.1) are specified along with their release dates.

ID	NVIDIA GPU	chip type	core clock (MHz)	#cores	TDP (W)	launch ($Q/YYYY$)	ref.
1	Tesla C1060	GT200	1,296	240	187.8	1/2008	[**203**]
2	GeForce GTX680	GK104	1,058	1,536	195	1/2012	[**204**]
3	Tesla K20	GK110	706	2,496	225	4/2012	[**205**]
4	Quadro 4000	GF100	475	256	142	4/2010	[**206**]

5.4. Machines

Runtime evaluation experiments were carried out on workstations with differing setups. Table 5.2 provides an overview on them. When compiling CUDA application with `nvcc`, the author set the flags `O3` and `use_fast_math`. For x86 code compilation with `gcc`, the author set the flags `O3`, `fomit-frame-pointer`, `funroll-loops`, and `ftracer`.

CHAPTER 6

Experiments

6.1. Optimizing Ligands in Rigid Binding Pockets

The following experiment assesses whether force field-based ligand minimizations can enhance results produced by docking tools. It therefore compares local and global optimization methods and gauges the effects of employing different force fields. Based on thus gained insights, the research parameterizes and subsequently validates and externally evaluates Trooper.

6.1.1. Assessing the potential of downstream optimizations. To gauge the impact of downstream optimizations on the geometrical quality of docking poses, the experiment optimized the 13,255 ligand poses of data set ADS I (see Section 5.1.1) with YASARA's standard minimization procedure (see Section 3.9.1) using the AMBER03 force field (see Section 3.4.5) as an objective function.

As the ligand poses of data set ADS I result from a cognate docking run, receptor conformations correspond to those of the respective crystal structure. Thus, the experiment kept binding sites rigid in all optimization runs. To speed up computations while assuring that YASARA captures all relevant non-bonded interactions, a 13.1 $\mathring{A}$ cutoff was imposed.

The following judges the usefulness of downstream optimizations in terms of the number of correct and excellent ligand poses (see Section 3.2) gained. The pre-optimized data set ADS I contains at least one correct docking pose for 61 and at least one excellent docking pose for 22 protein targets. YASARA's optimization raises the former number to 66 and the latter to 50. Figure 6.1 provides an overview on all measured results.

Discussion. Evidently, a force field-based optimization can improve the geometry of docking poses. This particularly applies to poses with an RMSD to the crystal structure below 1.5 $\mathring{A}$. These measurements coincide with those of Nabuurs et al. [**41**] who report reductions by 0.31 $\mathring{A}$ to 0.41 $\mathring{A}$ for poses with an initial RMSD below 1.5 $\mathring{A}$ with YASARA.

For poses with an RMSD above 2.0 $\mathring{A}$, the results indicate that optimization effects are less apparent. Furthermore, YASARA's optimization procedure seems to drive poses with initial RMSDs larger than 3.0 $\mathring{A}$ away from crystal structure poses. This is acceptable, this work aims at refining pre-determined conformations that are already close to biologically active ones. For discovering genuinely new binding modes, other approaches have to be applied prior to quick optimization methods. Thus, this focuses on poses with an RMSD in the range of 1.5 $\mathring{A}$ to 3.0 $\mathring{A}$ in

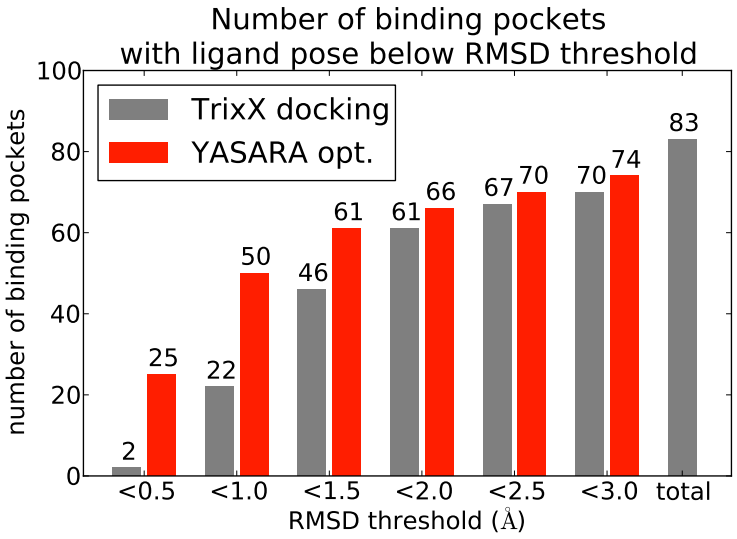


FIGURE 6.1. Effects of downstream optimizations with YASARA on ligand poses resulting from a cognate docking experiment with TrixX. The bars depict the number of binding pockets for which there is at least one pose with an RMSD below the threshold value denoted on the x-axis. Grey bars represent the pre-optimized and red bars the optimized data set. The rightmost bar illustrates the number total number of binding pockets in the experiment's data set.

the following and analyze whether a less complex and thus presumably faster local optimization method can sustain YASARA's performance or even produce better results.

6.1.2. Comparing global to local optimizations. This experiment assesses whether less complex local optimizations produce better or equal results compared to a global optimization procedure. Theoretically, global optimization schemes could drive ligands towards minimal poses far away from the respective crystal structure.

This experiment therefore minimizes data set ADS III, which comprises ligand poses, with RMSDs to their respective crystal structures in the range of 1.5 $\AA$ to 3.0 $\AA$. As explained in the preceding section, this RMSD range is of high relevance.

Global and local optimizations were performed with YASARA and MOE, respectively. Protein atoms were fixed during the experiment as the ligands of data set ADS III result from a cognate docking run. For YASARA, this experiment employed its standard minimization procedure (see Section 3.9.1) with AMBER03 (see Section 3.4.5) as an objective function and imposed a cutoff of 13.1 $\AA$ on non-bonded interactions. As YASARA performs stochastic optimizations, the data are averaged over the results of three optimization runs when computing RMSD values.

With MOE, a single run of its deterministic gradient-based optimization procedure (see Section 3.9.2) was conducted using AMBER03 as an objective function. Proteins were prepared

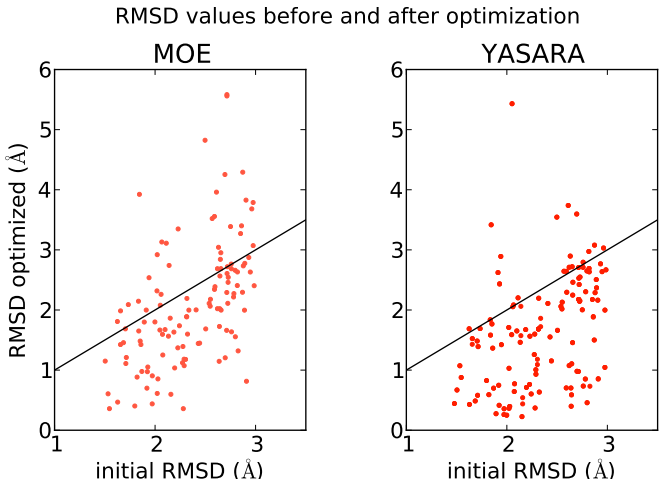


FIGURE 6.2. RMSD values of ligands before and after optimization in their rigid binding sites. MOE employed a local and YASARA a global minimization scheme.

by adapting their protonation states and assigning partial charges. Furthermore, the experiment set the boundaries of MOE's cutoff switching function to 12 $\mathring{A}$ and 14 $\mathring{A}$.

RMSDs for ligand poses were calculated with MOE's integrated method. The average RMSD of the data set's ligands to their respective crystal structure is reduced from an initial value of 2.35 $\mathring{A}$ down to 2.07 $\mathring{A}$ by MOE and down to 1.67 $\mathring{A}$ by YASARA. Figure 6.2 shows the experiment's results.

Discussion. YASARA's global minimization procedure outperforms MOE's local one, especially on ligand poses with a RMSD above 2.0 $\mathring{A}$ to the respective crystal structure. This is striking in the light of the analysis of downstream optimization procedures in Section 2.2. All therein introduced tools apply local optimization techniques. However, this experiment suggests to learn from YASARA's approach and employ a global minimization strategy based on SA. Trooper was developed in accordance with the findings. It is parameterized in the next experiment.

6.1.3. Parameterizing Trooper's SA algorithm. Results of the preceding experiments suggest to employ a SA-based method for ligand optimizations. For this, parameterization runs are required.

For minimizing the 125 ligands of training data set ADS II (see Section 5.1.1), Trooper was tested various settings. While protein structures were kept rigid and SuperTrAmber (see Section 3.4.1) served as an objective function in all runs, the maximum angle for molecular and dihedral rotations (ρ_{max} and Φ_{max}) as well as the maximum length of the translation vector $[max(|\vec{t}|)]$ and the initial temperature (T) were subject to the tuning procedure. All combinations of the assignments $T = \{0, 10, 100\}$, $\rho_{max} = \{\frac{\pi}{4}, \frac{\pi}{12}, \frac{\pi}{48}\}$, $\Phi_{max} = \{\frac{\pi}{4}, \frac{\pi}{12}, \frac{\pi}{48}\}$, and

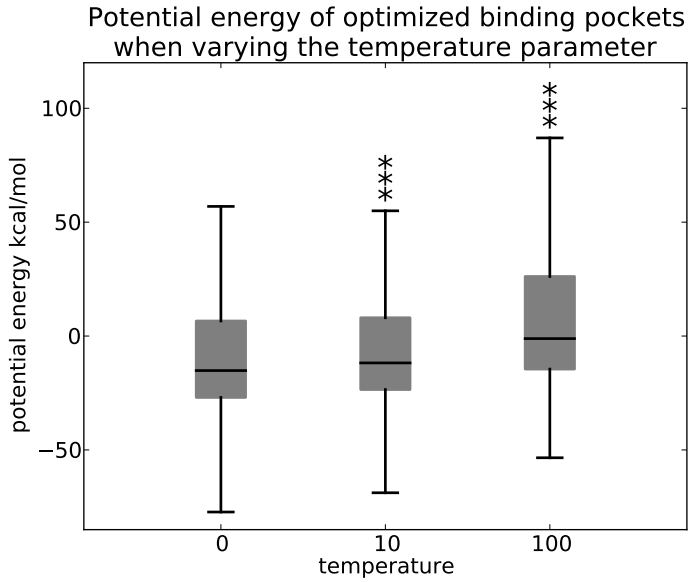


FIGURE 6.3. Median potential force field energy of protein-ligand complexes resulting from ligand minimizations. The median energy values were computed for minimization runs conducted with the same initial temperature parameter T. For all other parameters, different combinations were tested.
Symbols above the whiskers indicate whether the respective median energy value differs significantly from that produced by optimization runs with temperature setting $T = 0$. Three stars (***) denote a significance level of 0.1%.

$max(|\vec{t}|) = \{0.1, 0.2, 0.4\}$ were tested. For all optimization runs, the maximum number of steps was set to 3,000 and the cooling factor (c_f) to 0.98.

For selecting the best performing parameter combination, the author computed the median potential energy values of optimized complexes for all optimization runs conducted with the same initial temperature T. The obtained results are shown in Figure 6.3.

Furthermore, Table 6.1 lists the median potential energy of structures resulting from Trooper's minimization procedure for all parameter combinations when keeping initial temperature T fixed to 0.

Discussion. Apparently, Trooper's optimization performance is best when setting the initial temperature to 0 ($p \leq 0.001$). This results in a hill climb-like and thus local minimization procedure. As the previous experiment with MOE and YASARA suggests that YASARA's adaptive SA procedure is superior to MOE's gradient-based local optimization, data suggests that the non-adaptive exponential cooling schedule (see Section 4.5.4) employed in the tested version of Trooper could be the reason for the worse performance measured for global optimizations. However, the work sets the initial temperature T to 0 for the following ligand minimizations without protein flexibility. An adaptive cooling schedule is tested for optimizing amino acid side chains in Section 6.4.

TABLE 6.1. Median potential energies (E) of binding pockets resulting from ligand minimizations performed with Trooper utilizing SuperTrAmber as an objective function. For parameterization purposes, different combinations of the maximum angle for molecular and dihedral rotations (ρ_{max} and Φ_{max}) as well as the maximum length of the translation vector $[max(|\vec{t}|)]$ were tested. The last column specifies whether the respective median potential energy value significantly differs from the one in the first row of the table. The minus symbol (-) denotes no significance.

$max(\|\vec{t}\|)\,(\mathring{A})$	$\rho_{max}\,(RAD)$	$\Phi_{max}\,(RAD)$	median $E\,\frac{kcal}{mol}$	significance
0.1	0.78	0.06	-17.39	-
0.1	0.26	0.78	-17.34	-
0.4	0.06	0.78	-16.94	-
0.1	0.06	0.78	-16.92	-
0.1	0.26	0.06	-16.32	-
0.4	0.26	0.26	-15.96	-
0.4	0.78	0.26	-15.86	-
0.4	0.06	0.26	-15.71	-
0.2	0.06	0.78	-15.65	-
0.4	0.78	0.06	-15.64	-
0.4	0.26	0.78	-15.36	-
0.2	0.78	0.06	-15.33	-
0.2	0.06	0.26	-15.19	-
0.2	0.26	0.78	-15.12	-
0.1	0.06	0.06	-15.10	-
0.2	0.26	0.26	-14.95	-
0.4	0.26	0.06	-14.93	-
0.1	0.78	0.78	-14.91	-
0.1	0.06	0.26	-14.80	-
0.2	0.26	0.06	-14.71	-
0.2	0.78	0.26	-14.58	-
0.4	0.78	0.78	-14.54	-
0.1	0.26	0.26	-14.52	-
0.1	0.78	0.26	-14.38	-
0.4	0.06	0.06	-14.37	-
0.2	0.06	0.06	-14.36	-
0.2	0.78	0.78	-13.78	-

For all other parameters, the experiment yields no indications on a preferable combination. As this research focuses on optimizing ligands with distance less than $3\,\mathring{A}$ to their respective crystal pose, it assumes that rather small movements suffice for the purpose of this work. Therefore, this research sets $\rho_{max} = 0.06\,RAD$, $\Phi_{max} = 0.06\,RAD$, and $max(|\vec{t}|) = 0.1\,\mathring{A}$ for the following ligand optimization experiments. This setting, along with the SA parameterization introduced previously is named the local parameterization. Section 4.8.1 summarizes it.

6.1.4. Force field comparison: MMFF94s vs SuperTrAmber. To assess whether utilizing the MMFF94s force field (see Section 3.4.3) instead of SuperTrAmber as an objective function yields better results when minimizing ligands within rigid binding pockets, Gent conducted a study as part of his Master's thesis [**195**]. Its results are relevant to this work as all following optimization experiments could be performed with MMFF94s, which is part of the force field framework (see Section 4.2).

Gent optimized the 67 docking poses of data set ADS III (see Section 5.1) with Trooper. The initial temperature T was set to 0 and the maximum number of steps s_{max} to 3,000. Parameters of the movement magnitudes for molecule transformations are not specified in Gent's work but were kept constant in all optimization runs. Of these, Gent conducted 20 with MMFF94s and SuperTrAmber as an objective function.

RMSDs of resulting ligand poses to their respective crystal structure were computed with an unspecified method. Averaging over the thus obtained values for all runs yields an RMSD value of 2.03 $\AA$ for SuperTrAmber and 2.05 $\AA$ for MMFF94s. The difference between these two results is not significant ($p = 0.5$) according to Student's t-test (see Section 3.10).

Discussion. Optimizations with Trooper using either the MMFF94s or the SuperTrAmber force field as an objective function yield the same results. This finding correspond to that of Perola et al. [**19**] who reported similar downstream optimization performance for the MMFF94 and OPLS-AA [**25**, **26**] force fields.

At the same time, SuperTrAmber's parameter set and its atom type model (see Section 3.5) are simpler than those of MMFF94s. In accordance with Occam's principle of parsimony, this work selects the simpler model and conducts the following experiments with SuperTrAmber as an objective function. In the next experiments, this work proceeds by evaluating the now fully parameterized Trooper by comparing its performance to external tools.

6.1.5. Optimizing selected docking poses. For evaluation purposes, this experiment optimized the 67 ligands of test data set ADS III (see Section 5.1.1) in their respective binding pocket with Trooper's local minimization mode and compared the results to those obtained with MOE and YASARA from the experiment described in Section 6.1.2.

During all optimization runs, protein atoms were kept fixed and the local parameterization specified in Section 4.8.1 was employed. A cutoff of 12 $\AA$ around ligand atoms was introduced for non-bonded interactions. The experiment performed 10 optimization runs and computed an average RMSD (using MOE's internal implementation) of 1.95 $\AA$ for the minimized ligands to their crystal structure. As shown in Figure 6.4, Trooper performs best on structures with an RMSD between 1.5 and 2.5 $\AA$. The quality of poses with an initial RMSD above 2.5 $\AA$ is enhanced in some cases while most often, only negligible effects are observable.

Discussion. Trooper's local optimization mode performs better than MOE's and is outperformed by YASARA's global minimization, especially on ligand poses with an initial RMSD above 2.5 $\AA$ to the respective crystal structure. This is another indicator that testing an adaptive

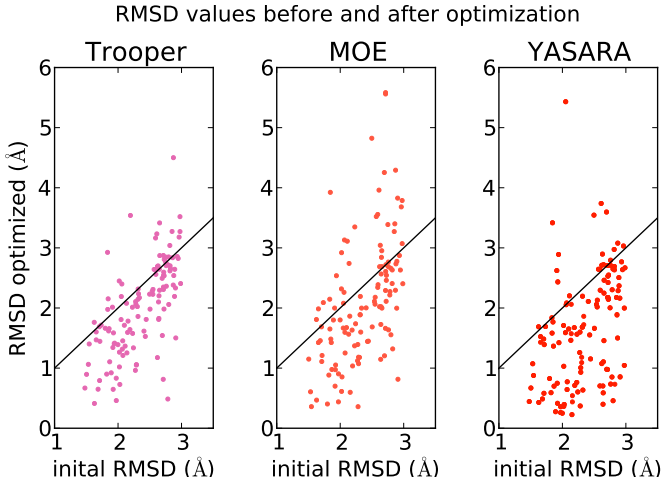


FIGURE 6.4. RMSD values of ligands before and after optimization in their rigid binding sites. MOE, YASARA, and Trooper's local parameterization were employed for minimizing the structures.

cooling scheme as employed by YASARA might be worthwhile. Sections 6.4 and 6.6 therefore assess this methodology. At the same time, Trooper apparently already performs sufficiently well to employ it for downstream optimizations in cognate docking. The following experiments test this assumption.

Results produced by Trooper in this section's experiment are apparently slightly better than those reported in Gent's experiments (see Section 6.1.4). At the same time, a comparison is hardly valid as Gent's parameterization is partly unknown and the employed method for RMSD value calculations is not specified.

6.1.6. Downstream optimization of cognate docking results. To gauge Trooper's potential to ameliorate cognate docking results, this experiment optimized the docking poses of data set ADS I (see Section 5.1.1). For this, Trooper's local parameterization was employed (see Section 4.8.1). Furthermore, a cutoff of 12 $\AA$ was imposed on non-bonded interactions. The work then compares the thus produced results with those obtained when optimizing with YASARA (see Section 6.1.1). For this, the RMSD (see Section 3.2) of an optimized ligand's atoms to the respective crystal structure atoms are computed. Figure 6.5 summarizes the results. Therein, it is shown that the number of binding pockets for which there is at least one correct docking pose (defined in Section 3.2) is enhanced from 61 to 66 by YASARA and to 68 by Trooper. For excellent poses, YASARA increases this number from 22 to 50 and Trooper to 44.

Discussion. When considering the number of binding pockets for which excellent or correct poses are produced, results produced by Trooper are almost on par with YASARA's especially for the region below 1.5 $\AA$. Certainly, the up to 200 starting poses per binding pocket entail a high number of attempts to produce a correct or excellent pose. However, this experiment still

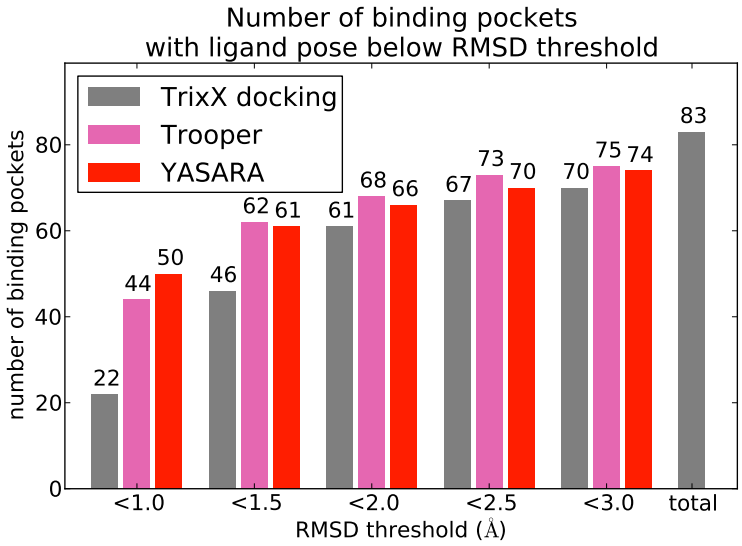


FIGURE 6.5. Effects of downstream optimization with YASARA and Trooper's local minimization mode on ligand poses resulting from a cognate docking experiment with TrixX. The single bars represent the number of binding pockets for which there is at least one pose with an RMSD below the threshold value denoted on the x-axis. The rightmost bar illustrates the number total number of binding pockets in the experiment's data set.

demonstrates that Trooper can enhance the results of cognate docking runs. The next experiment assesses whether this observation holds true when rescoring with HYDE (see Section 3.8.1) to select the top-ranking poses.

6.1.7. The ACS docking challenge. This research evaluated Trooper's potential to improve results of cognate docking experiments by optimizing and rescoring ligand poses produced by TrixX (see Section 3.8.3). In detail, this research post-processed docking poses by optionally minimizing them with Trooper's local mode, followed by a rescoring with HYDE (see Section 3.8.1), which optionally employed its internal numerical optimization. Thus, four pipelines for ligand refinement were tested:

(1) HYDE rescoring
(2) HYDE optimization → HYDE rescoring
(3) Trooper's force field optimization → HYDE rescoring
(4) Trooper's force field optimization → HYDE optimization → HYDE rescoring

This work prepared the docking run by generating conformations for each of the 151 ligands of data set ADS VII (see Section 5.1.3) with the TCG (see Section 3.8.4), whose clustering threshold was set to 1.2 and the quality level to 1. The research subsequently set TrixX's clash parameter

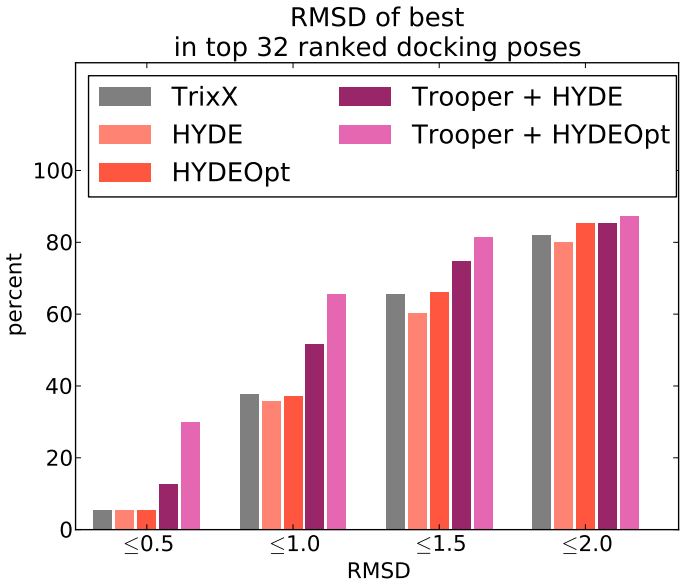


FIGURE 6.6. Effects of Trooper's downstream optimizations in combination with HYDE rescoring on the results of a cognate docking run with TrixX. Four refinement pipelines (*HYDE*, *HYDEOpt*, *Trooper + HYDE*, and *Trooper + HYDEOpt*) were tested. The single bars represent the percentage of binding pockets for which there is at least one ligand pose among the top 32 ranked with an RMSD to the respective crystal structure below the threshold value denoted on the x-axis.

to 0.5 and attempted to dock all generated conformations into their respective binding pocket. This resulted in a maximum of 200 docking poses per ligand.

For downstream optimizations, this experiment used Trooper's local parameterization (see Section 4.8.1) and imposed a cutoff of 12 $\AA$ on non-bonded interactions.

Rescoring was performed with the HYDE tool with parameters `-Hnetwork 1` and `-optimize 2`. This enabled the numerical optimization and the ProToss algorithm (see Section 3.8.2) for improving the hydrogen bond network in binding pockets.

A comprehensive overview on the experiment's results is given in Table 6.2 and Figure 6.6. Due to the large amount of data therein, the following discusses selected points.

Discussion. It is evident that downstream optimizations with Trooper enhance the number of excellent docking poses (term defined in Section 3.2). This conclusion is based on the results produced by pipeline 3, which increases the number of binding pockets with at least one excellent pose from 59 to 85 when considering the top 200 ranked poses (see last row of Table 6.2). Here, the ranking is irrelevant for deriving the number of excellent poses and thus HYDE's rescoring has no influence. Hence, Trooper is the sole cause for the improved results.

Remarkably, Trooper furthermore boosts the performance of HYDE's internal optimizer. As visible in the last row of Table 6.2, for the number of binding pockets with at least one excellent

TABLE 6.2. Effects of Trooper's downstream optimizations in combination with HYDE rescoring on the results of a cognate docking run with TrixX. Table cells contain the number of binding pockets for which at least one ligand pose with an RMSD below the given threshold is ranked among the top n. The numerical column labels refer to the four downstream refinement pipelines that are defined in experiment's description. Columns containing raw TrixX results are labeled with '-'.

	RMSD (Å)																			
	≤ 0.5					≤ 1.0					≤ 1.5					≤ 2.0				
Rank	-	1	2	3	4	-	1	2	3	4	-	1	2	3	4	-	1	2	3	4
≤ 1	6	4	1	10	27	34	17	20	38	67	61	37	56	55	88	82	56	79	72	98
≤ 5	8	6	4	12	37	49	30	41	56	83	81	68	84	84	108	101	93	105	101	117
≤ 10	8	7	6	13	44	53	39	49	65	89	94	77	90	101	114	116	101	115	116	125
≤ 20	8	8	8	16	45	57	52	53	74	95	96	86	99	109	119	120	118	124	125	130
≤ 32	8	8	8	19	45	57	54	56	78	99	99	91	100	113	123	124	121	129	129	132
≤ 200	9	9	9	20	47	59	59	59	85	101	104	104	104	121	126	134	134	134	135	135

docking pose among the top 200 ranked, no improvement is observable when comparing the results of pipelines 1 and 2. Thus, HYDE's internal optimizer has no effect in this scenario. However, as visible when comparing the result of pipelines 3 and 4, the number of binding pockets with at least one excellent ligand pose among the top 200 ranked is enhanced by 16. Clearly, the cause of this effect is that HYDE's optimizer operates on poses minimized by Trooper. This suggests combining the HYDE and SuperTrAmber functions for improving docking results in future work.

When considering rescoring effects, HYDE's optimizer again benefits from preceding minimizations with Trooper. Looking at the top-ranked pose (see first row of Table 6.2), one can compare the result of pipelines 1 and 2. Here, the internal HYDE optimization additionally yields three excellent and 23 correct poses. In contrast, the number of excellent and correct poses is increased by 29 and 26, respectively, when performing HYDE optimizations downstream on poses refined by Trooper. This follows from comparing the figures of pipelines 3 and 4.

Furthermore, Trooper outperforms HYDE's internal optimizer in terms of producing excellent poses that at the same time match with the HYDE model. When scoring with HYDE after using its internal optimization (pipeline 2), 56 binding pockets with at least one excellent ligand pose are among the top 32 ranked results (see next to last row in Table 6.2). In contrast, when replacing HYDE's optimization with Trooper in pipeline 3, 78 such binding pockets result. This contradicts a hypothesis by Cole et al. [**4**] and O'Boyle et al. [**5**]. They independently claimed that rescoring, especially with non-smooth functions, is ineffective when preceded by optimizations with another objective function

To sum it up, Trooper's local minimization mode by itself bears the potential of improving the placement of docking poses. Further pose enhancements result from a combination with

TABLE 6.3. Number of binding pockets for which the specified docking pipeline generated excellent and correct ligand poses. The number of top-ranked poses that were considered are enclosed in parentheses.

RMSD	pipeline 4	FlexX	ICM	SurFlex	Lead Finder	GOLD	DOCK6	MOE
$\leq 1.0\mathring{A}$	66 (32)	69 (32)	86 (3)	-	-	76 (25)	-	62 (30)
$\leq 2.0\mathring{A}$	87 (32)	89 (32)	95 (3)	93 (20)	91 (20)	91 (25)	92 (32)	87 (30)

HYDE. This scoring function benefits from an enhanced number of geometrically sound poses that at the same time match its scoring model.

External evaluation. For externally evaluating docking pipeline 4, this work compares its results in Table 6.3 to those reported after the Docking and Scoring Symposium that took place during the 241st ACS National Meeting. For this Symposium, the work groups developing FlexX [**101**], ICM [**207**], SurFlex [**208**], Lead Finder [**209**], GOLD [**210**], DOCK [**211**], and MOE [**212**] tested their respective docking tool on data set ADS VII.

GOLD and ICM apparently produce a higher number of excellent poses than all other tools. However, pipeline 4 performs on par with the external tools at generating correct poses. Consequently, the following focuses on the time consumption of Trooper. Clearly, next to the quality of results, a low runtime is a vital trait of a well-engineered docking pipeline.

6.1.8. Optimization runtime. To evaluate optimization runtimes, this research compiled a subset of data set ADS I (see Section 5.1.1) which comprises the complexes carbonic anhydrase XII (*1jd0*), DAHP synthase (*1of6*), protein kinase 5 (*1v0p*), uridine phosphorylase (*1u1c*), protein farnesyltransferase (*1mzc*), and vitamin D nuclear receptor (*1s19*). The binding pockets of complexes *1jd0* and *1of6* contain a relatively small number of atom pairs while those of complexes *1v0p* and *1u1c* contain an average number of atom pairs. Furthermore, the ligands of complexes *1mzc* and *1s19* are comparatively large and contain a high number of rotatable bonds, which enhances the complexity of minimizations. In all enumerated pairs of complexes, the total number of atoms differ significantly.

With the compiled test set, the author aimed at deriving whether binding pocket or total protein sizes determine the runtime of the tested tools and therefore minimized 200 ligand poses for each complex while keeping the protein rigid using Trooper, MOE, and YASARA. For both MOE and YASARA, the respective standard minimization procedure, described in Sections 3.9.2 and 3.9.1 was employed. The upper and lower limit of MOE's cutoff switching function were set to 12 $\mathring{A}$ and 14 $\mathring{A}$, while YASARA's cutoff on non-bonded interactions was set to 13.1 $\mathring{A}$. Trooper was tested with its local parameterization, which Section 4.8.1 specifies, and a cutoff of 12 $\mathring{A}$. The experiment was executed on test machine I (see Table 5.2) using a single CPU core.

YASARA consumes 700 s and MOE 1,363 s for minimizing 200 ligand poses in the binding pocket of complex $1of6$. For both external methods, this constitutes the highest runtime in this

TABLE 6.4. Runtime for optimizing 200 poses of the specified protein's cognate ligand. Furthermore, the number of atom pairs considered by Trooper is stated along with the protein's total number of atoms.

Complex pdb code	#pairs	#atoms	Trooper [s]	MOE [s]	YASARA [s]
1jd0	11,649	8,142	9.95	187.87	129.60
1of6	15,670	41,839	14.25	1,363.09	700.00
1u1c	22,365	22,641	23.51	475.50	411.62
1v0p	30,308	8,753	24.31	182.50	131.05
1mzc	39,108	11,698	40.05	337.50	300.00
1s19	47,222	4,055	46.16	227.79	168.65

experiment. At the same time, $1of6$ has $41,839$ atoms and is thus the largest complex in the test set.

In contrast, Trooper exhibits its highest runtime ($46.16\,s$) when optimizing 200 ligand poses for complex $1s19$, whose binding pocket has the largest number of atom pairs (47,222) in the test set.

On average, an optimization run with Trooper takes about $30\,s$, while YASARA's and MOE's procedure consume $307\,s$ and $462\,s$, respectively, and are thus one order of magnitude slower. All further results are presented in detail in Table 6.4.

Discussion. YASARA's high time consumption corresponds to results by Nabuurs et al. [**41**], which are discussed in Section 2.2. Evidently, the runtimes of both external methods correlate with the total number of atoms while Trooper's runtime depends on the number of atom pairs in binding pockets. This highlights the benefits of focusing on binding site minimizations. The external tools apparently spend time calculating intramolecular interactions for protein atoms, although this experiment explicitly set all protein atoms as rigid. In contrast, the number of rotatable ligand bonds seemingly has no major impact on the optimization runtime, as no method exhibits larger aberrations for complexes *1mzc* and *1s19*.

Trooper's runtime behavior is on par with that of local optimization procedures used in RosettaLigand and SurFlex (see Section 2.2). However, significant accelerations are necessary for enhancing the throughput of force field-based optimizations in computational drug design contexts. In virtual screening runs, they constitute a runtime bottleneck and in lead optimization scenarios, methods that allow for an interactive workflow (see Section 3.1) are preferable. Therefore, the following focuses on this issue.

6.2. Grid-Based Acceleration

This section compares Trooper's local mode to the alternative accelerated grid-based version, which Section 3.7 introduces. This experiment therefore minimized the ligands of data set ADS III (see Section 5.1.1) in their respective binding pocket, which was kept rigid. As in the preceding experiments, this experiment used the local parameterization (see Section 4.8.1) for Trooper but

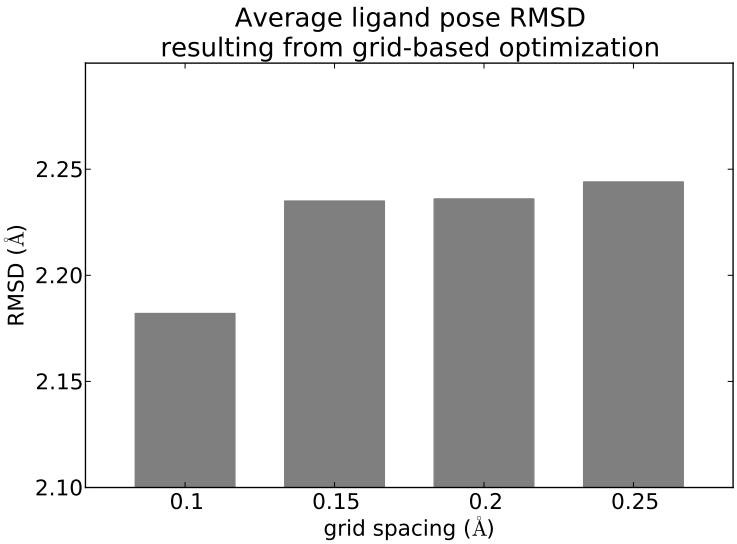


FIGURE 6.7. Average RMSD to the respective crystal structure of ligand poses optimized in binding pockets with the accelerated grid-based procedure. Grid spacings as specified in ångstrom by the labels of the x-axis were tested.

employed SuperTrAmber-based grid potentials as an objective function. This work evaluated grid spacing settings 0.1 $\mathring{A}$, 0.15 $\mathring{A}$, 0.2 $\mathring{A}$, and 0.25 $\mathring{A}$. Details on the grid construction procedure are laid out in Section 3.7

Further, this experiment computed energy values by applying the trilinear interpolation scheme (see Section 3.3.4). As shown in Figure 6.7, computing the average RMSD of the minimized ligands to their respective crystal structure yields values between 2.18 $\mathring{A}$ and 2.24 $\mathring{A}$, depending on the employed grid size.

A further experiment tested the runtime of grid-based scoring and grid construction. To test this, the experiment built the grids described in Section 3.7 with grid spacing 0.1 $\mathring{A}$ in the binding pocket of carbonic anhydrase XII (*1jd0*). This pocket was defined as the spatial region enclosed by spheres with radius 6 $\mathring{A}$ around the 19 atoms of the native ligand.

After constructing the grids, the experiment computed the non-bonded interaction energies between ligand and binding pocket atoms 10,000 times. This computation took less than 1 s on a single CPU core of test machine I (see Table 5.2) while building the grids consumed 419 s.

Discussion. The grid-based version of Trooper performs significantly worse than its standard method (see Section 6.1.5) irrespective of the selected grid spacing. These findings accord with those of Wu et al. who report that "No significant differences in docking accuracy were observed using a grid spacing ranging from 0.25 to 1.0 $\mathring{A}$" [**57**, p. 1551] and "full force field minimization leads to a significant improvement in docking accuracy" [**57**, p. 1556]. At the same time, preliminary computations for building the grid consume several minutes. Thus, data

suggests that the grid-based method lacks an additional benefit. Therefore, this work focuses on GPU-based optimizations.

6.3. Accelerating Ligand Optimizations with GPUperTrooper

All in all, four experiments evaluate and validate GPUperTrooper. The first experiments assesses the runtimes of the CPU- and GPU-based force field energy function implementations on complete protein-ligand complexes and compares them to MOE and YASARA. The following two experiments demonstrate that the speedup (see Section 3.1) obtained with the GPU-based force field energy function implementation is sustainable when scoring and optimizing ligands in binding pockets. As discussed in Section 1.5, harnessing the speedup obtained through implementing energy functions on GPU in a complete optimization procedure constitutes a major challenge. Finally, the fourth experiment compares the quality of results produced by GPUperTrooper and Trooper. The following sections employ GPU-specific terminology and therefore readers are recommend to read Section 1.5 before continuing.

6.3.1. Scoring full protein-ligand complexes. The following experiment assessed the potential speedup, which the GPU-based force field energy kernels yield. For this, total energies for the 11 proteins in data set ADS VI (see Section 5.1) were computed 1,000 times. In this way, runtimes of MOE, YASARA, and the CPU- and GPU-based versions of the SuperTrAmber force field were compared.

MOE (see Section 3.9.2) was set up by selecting the AMBER03 force field (see Section 3.4.5). Then, missing hydrogen atoms were added to the complexes and partial charges were assigned. For calculating energies, this experiment utilized the `Potential[]` function, which is part of MOE's `SVL` interface. For YASARA (see Section 3.9.1), this experiment prepared the complexes with the `Clean` routine and then calculated AMBER03 force field energies using the `Energy` function.

To evaluate the effect of large systems on runtime behavior, no cutoffs were imposed in any of the experiments. For YASARA, this proved to be impossible, as it requires the user to set a minimum cutoff of 20.97 $\AA$ for van der Waals interactions.

All CPU-based experiments were conducted on a single core of test machine I (see Table 5.2), while GPU-based computation were performed on a NVIDIA Tesla C1060 with 240 cores. Comparing results of the CPU- and GPU-based force field implementations yields a speedup of roughly 300. Furthermore, the GPU-based implementation performs about 100 times faster than MOE. Measurements for YASARA are ambiguous, as discussed next. Figure 6.8 provides an overview on all results.

Discussion. As expected, runtimes for computing force field energies correlate with the number of atoms in the target system. This is not perfectly apparent for YASARA, as it does not allow for switching off the cutoff for van der Waals interactions. Furthermore, YASARA repetitively performs parameterizations for dihydropterin pyrophosphokinase (*1hq2*) and thus

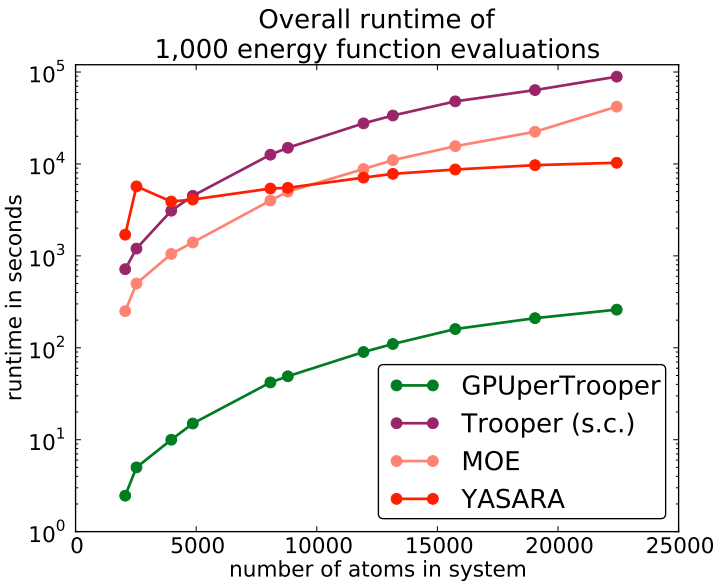


FIGURE 6.8. Runtime for computing the force field energy for protein-ligand complexes. Runtimes are specified in seconds on the log-scaled y-axis while complex sizes are specified on the x-axis. The experiment was conducted with YASARA, MOE, and the CPU-based version of the SuperTrAmber force field on a single CPU core. For SuperTrAmber, the GPU-based algorithm was tested as well.

produces a runtime peak. For these reasons, runtimes measured for YASARA are difficult to compare.

In contrast, MOE clearly performs computations faster than the CPU-based force field implementation. This is not surprising, as this method is mainly optimized for evaluating ligands in binding pockets and smaller molecular systems.

Despite this, the GPU-based kernels consistently produce force field energies two orders of magnitude faster than MOE. This speedup is in the same range as speedups reported by other groups. Stone et al. [**118**] measured a speedup of 40 when benchmarking their GPU and CPU implementations of a Coulomb summation. Their comparison runs were executed on a NVIDIA GeForce 8800 GTX with 128 CUDA cores and a single core of a 2.6 GHz Intel Xeon processor.

Van Meel et al. [**120**] ported a Lennard-Jones potential to GPU. This resulted in a speedup of 80 on a NVIDIA GeForce 8800 GTX, in comparison to a single core of a 3.2 GHz Intel Xeon processor. The experiment of van Meel et al. covers system sizes ranging from 200 up to 70,000 atoms and their implementation takes about 50 s to compute the pairwise Lennard-Jones potential 1,000 times for 10,000 atoms. In comparison, this research additionally calculates the Coulomb potential and dihedral angle energies and evaluates a system with 11,937 atoms 1,000 times in 90 s.

Schmid et al. [**124**] designed a full non-bonded kernel for solvent–solvent interactions. They observed speedups in the regime of 50 for long-range and 25 for short-range interactions when computing energies for systems with 5,411 up to 75,129 atoms. Their implementations were executed on a NVDIA Quadro FX 5800 equipped with 240 CUDA cores and a single AMD Athlon X2 core running at 3.2 GHz.

Friedrichs et al. [**123**] employed a 2.66 GHz Intel Xeon and a NVIDIA GeForce GTX 280 with 240 CUDA cores to obtain a speedup of 128, up to 735, with their full force field kernel implementation. Tested system sizes ranged from 544 up to 5,078 atoms. However, the rescaling of intramolecular interactions between neighboring atom pairs was neglected.

Recently, Götz et al. [**128**] evaluated their GPU portation of the AMBER force field on a NVIDIA Tesla C1060 GPU, which this research also employed. For systems of sizes 2,492 and 25,094 atoms, respectively, their full force field kernels took $5.48\,s$ and $432\,s$ to execute 1,000 times. For this work's GPU algorithm, the measured runtimes were $5\,s$ and $260\,s$ for systems with 2,525 and 22,434 atoms, respectively. Even on larger systems, this work's GPU algorithm thus performs on par with a commercially available portation.

In conclusion, the GPU-based energy kernels yield a speedup for larger molecular systems capable of competing with recent implementations. The next experiment assesses whether this speedup is sustainable when focusing on binding pockets.

6.3.2. Focused scoring of binding pockets. With the following experiment, the speed of the objective function employed in actual optimization runs was evaluated. In contrast to the full force field kernels tested in the preceding experiment, this objective function is designed to operate on binding pockets. The function was tested on the 15 binding pockets of data set ADS IV (see Section 5.1). This set covers the range of non-bonded atom pairs occurring in binding pockets of the Astex Diverse Set.

The experiment imposed a cutoff of $12\,\mathring{A}$ on non-bonded interactions around ligand atoms. For both the CPU and GPU methods, the kernels constituting the objective function of the optimization were called 1,000,000 times and the accumulated runtime was measured.

On average, computations on the Tesla C1060 GPU of test machine I (see Table 5.2) took about $100\,s$ and are thus 102 times faster than on a single core of the CPU of test machine I. The reader is referred to Figures 6.9.a and 6.9.b for a detailed visualization of measured runtimes and speedups.

Discussion. As apparent in Figure 6.9.a, there is a quadratic relationship between the objective function runtime and the number of processed atom pairs. This result is in accordance with that of the preceding experiment. However, the number of considered atom pairs differs between the CPU- and GPU-based objective functions. First, this is due to the modified cutoff computations for the GPU-based algorithm. Second, ligand sizes have a major impact as intramolecular interactions between protein atoms are ignored. Dummy atoms added to ligands on GPU hence boost the number of considered atom pairs.

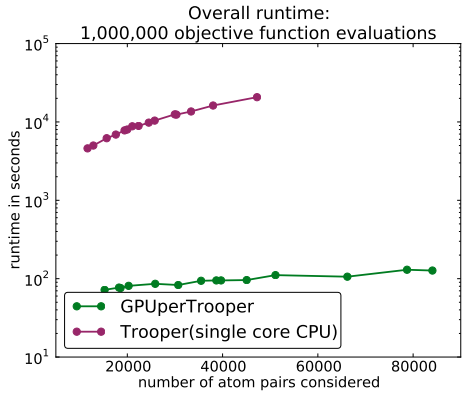


Figure 6.9.A: Runtime for computing the force field energy of binding pockets of protein-ligand complexes 1,000,000 times. The CPU- and GPU-based implementations of the SuperTrAmber force field were tested. Each method considered a differing number of atoms per complex, as specified by the labels of the x-axis.

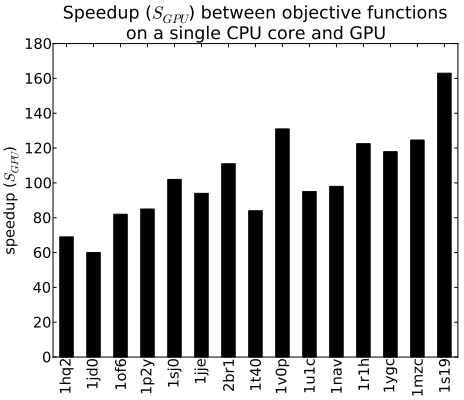


Figure 6.9.B: Speedup measured when energetically evaluating binding pockets of protein-ligand complexes with the CPU- and GPU-based implementations of the SuperTrAmber force field. The depicted values are averaged over 1,000,000 function calls. The labels on the x-axis specify the pdb code of the tested protein-ligand complexes.

FIGURE 6.9. Runtime and speedup for binding pocket scoring on GPU

Interestingly, speedups become larger as molecular system sizes increase (see Figure 6.9.b). This outcome can be linked to the constant kernel call overhead that can surpass the runtime of called GPU-based functions. Additionally, a relatively low number of thread blocks per multiprocessor results from smaller system sizes. This potentially entails insufficient workloads due to a certain number of thread blocks required per multiprocessor in order to hide global memory access latencies.

Furthermore, the observed speedups are on average about three times smaller than in the preceding experiment, which can be attributed to the less precise block-wise cutoff mechanism of the GPU implementation (see Section 4.3.2). It enhances the number of atom pairs processed on GPU in comparison to the CPU implementation. This also explains the smoother shape of the runtime curve for the CPU implementation in Figure 6.9.a. Furthermore, padding atoms added to ligands on GPU (see Section 4.3.1) necessitate additional computations.

Overall, this experiment reveals aspects of the GPU implementation that bear potential for further runtime reductions. At the same time, the observed speedups are sufficient for very fast optimizations. The next experiment evaluates whether the optimization algorithm efficiently maintains the speedup achieved up to this point.

6.3.3. Runtime of GPUperTrooper when optimizing ligands in binding sites. The runtime of GPUperTrooper was assessed with the experimental setup described in Section 6.1.8. Thus, the described set of protein-ligand complexes was optimized. Therefore, standard local parameters (see Section 4.8.1) were selected for GPUperTrooper. Furthermore, a cutoff of 12 $\AA$ was imposed on non-bonded interactions and protein atoms were fixed. The experiment furthermore

TABLE 6.5. Runtime of GPUperTrooper based minimizations for 200 cognate ligand poses in their respective binding pockets. The pdb codes of the tested protein-ligand complexes are specified, along with the number of atom pairs considered by GPUperTrooper. The last column specifies runtimes for minimization runs with activated early abort-mechanism. Speedup (S_{GPU}) and economic efficiency (E_{eco}) values for Trooper, MOE, and YASARA refer to runtime measurement described in Section 6.1.8.

pdb code	#pairs	runtime [s]	S_{GPU} Trooper	E_{eco} Trooper	S_{GPU} MOE	E_{eco} MOE	S_{GPU} YASARA	E_{eco} YASARA	runtime [s] w/ early abort
1jd0	18,256	0.26	38	4.0	426	45.4	618	65.8	0.09
1of6	18,576	0.26	55	5.9	1,961	208.8	3,820	406.8	0.16
1v0p	39,680	0.27	90	9.6	485	51.7	676	72.0	-
1u1c	45,056	0.27	89	9.5	1,553	165.4	1,795	191.2	0.20
1mzc	78,720	0.35	114	12.1	857	91.3	964	102.7	-
1s19	84,048	0.28	163	17.4	220	23.4	297	31.6	-

tested the early abort-mechanism (see Section 4.5.4) by switching it on in a second optimization run for complexes carbonic anhydrase XII (1jd0), DAHP synthase (1of6), and uridine phosphorylase (1u1c). The experiment was conducted on a single CPU core of test machine I (see Table 5.2) and on its NVIDIA Tesla C1060 GPU with 240 cores.

On average, an optimization run on GPU takes about 0.28 s per complex. This corresponds to a speedup of about 92 and an economic efficiency (see Section 3.1) of 9.7, with respect to the CPU-based implementation. Compared to the tested external software tools, the experiment yields speedup factors between 220 and 3,820. Economic efficiency factors range between 23.4 and 406.8.

Furthermore, optimization runs with activated early abort-mechanism yield an average runtime of 0.14 s per complex and thus speed up computations by about 50%. All further results are presented in detail in Table 6.5.

Discussion. In comparison to MOE and YASARA, GPUperTrooper speeds up calculations by at least two orders of magnitude and thus minimizes binding pockets of protein-ligand complexes in less than 0.3 s on average. A further runtime reduction by about 50% is achieved with the early abort-mechanism. This is very promising; it remains to be seen that the quality of the results is unaffected. This issue is tackled with the following experiment.

Recently, Anthopoulos et al. **[129]** published a GPU-based minimization approach that uses the MMFF94s force field (see Section 3.4.3). Their method takes about 1 s to perform 1,000 optimization steps. They use a NVIDIA GeForce GTX680 GPU, a processor that is nominally three times faster than the Tesla C1060 GPU which this research employs, as laid out in Section 5.4. Anthopoulos et al. optimized their algorithm for the Kepler architecture of the GeForce GTX680. For comparison, the runtimes of the early abort-accelerated approach are consequently

TABLE 6.6. Average over RMSDs between pre-optimized and optimized ligands and their respective crystal conformation. Two optimization protocols were applied for GPU-based minimizations: a standard and an accelerated one (early abort).

	pre-minimization	GPUperTrooper	GPUperTrooper (early abort)
RMSD (Å)	2.35	1.95	1.99
σ	-	0.029	0.026

multiplied with a scaling factor of 0.33. With this, GPUperTrooper is about 30 times faster than that of Anthopoulos et al.

Overall, the low runtimes enhance GPUperTrooper's usability in lead optimization applications and render downstream optimizations after large-scale docking runs more attractive. The fastest virtual screening methods known to the author, TrixX [**102**] and PhDOCK [**7**], take at least $0.1\,s$ to process a single conformation. When applying GPUperTrooper in a drug design pipeline in combination with these tools, runtimes remain in the same order of magnitude.

Further, applying GPUperTrooper bears economic advantages: it performs minimizations at least 4.0 times cheaper than any tested CPU-based method. Compared to MOE and YASARA, GPUperTrooper saves approximately more than 95% electric energy.

6.3.4. Quality of results of GPUperTrooper's ligand optimizations. To check the quality of results produced by the GPU-based optimization, it was applied for rerunning the minimization experiment described in Section 6.1.5. Thus, this experiment treated all protein atoms of complexes in data set ADS III (see Section 5.1.1) as rigid, imposed a cutoff of $12\,\mathring{A}$ around ligand atoms, and employed standard parameters for local optimizations (see Section 4.8.1). To obtain statistically sound results, 10 optimization runs were conducted. Further, 10 runs were started with the early abort-mechanism (see Section 4.5.4) switched on. For computing RMSD values, this experiment employed MOE and averaged over the 10 runs, respectively. Table 6.6 summarizes the results.

Discussion. In terms of quality, GPUperTrooper is an equivalent replacement of Trooper. This experiment's results show no significant aberrations between the two. The early abort-mechanism does not affect the RMSDs of poses produced by GPUperTrooper. At the same time, this mechanism significantly speeds up computations. The early abort-mechanism is therefore worth using.

6.4. Optimizing Amino Acid Side Chain Conformations

The following series of experiments elucidates areas of application and limitations of amino acid side chain optimizations without backbone flexibility. All experiments test Trooper's global

minimization mode, which uses the adaptive SA procedure described in Section 4.6. External comparisons apply the minimization methods provided by MOE and YASARA, which Sections 3.9.2 and 3.9.1 introduced.

6.4.1. Determining a parameterization range. The following experiment gauges the effects of modifying the maximum dihedral rotation parameter (Φ_{max}) of Trooper's adaptive SA procedure (see Section 4.6). Therefore, the amino acid side chains of binding pockets of data set NN I (see Section 5.2) were minimized. This data set comprises conformations of binding pockets superimposed onto sequentially identical template structures of the Astex Diverse Set [**3**]. The crystal conformation of template structure ligands were placed into the superimposed binding pockets. This entailed clashes between ligand and amino acid side chain atoms. In detail, these occurred for Lys99 in a human hemoglobin mutant (*1a0u*), Lys33, which is part of the binding pocket of human cyclin dependent protein kinase 2 (*1h0w*), Lys89 in themolysin (*1pe5*), Asn67, which is located in the binding region of protein kinase C (*1tb0*), Lys88, which belongs to p. falciparum pfpk5 (*1ob3*), and Met357 in human phosphodiesterase 4d (*1y2k*). Additionally, the side chain and backbone atoms of Tyr35 clash with the ligand in p38 (*1zzl*).

This research aimed at parameterizing Trooper such that it produces amino acid side chain conformations closer to the crystal structure and at the same time resolves clashes. To this end, this experiment tested all members of the set $\{\frac{\pi}{32}\,RAD,\ \frac{\pi}{16}\,RAD,\ \frac{\pi}{8}\,RAD, \frac{\pi}{4}\,RAD,\ \frac{\pi}{2}\,RAD,\ \pi\,RAD\}$ for parameter Φ_{max}. Furthermore, the maximum number of optimization steps (s_{max}) was set to 15,000, the starting temperature T was set to 30 and the SuperTrAmber force field (see Section 3.4.1) was employed as an objective function.

To assess the thus generated solutions, the author visually inspected them and categorized them as *resolved* if the optimizer removed the clashes between the ligand and the amino acids. The research furthermore analyzed whether conformations produced by Trooper were close to the crystal structure and discovered that this was the case for all settings of Φ_{max} but π. Table 6.7 summarizes the observations and Figures 6.10.a and 6.10.b present a pair of structures for which altering the Φ_{max} parameter leads to a resolved clash.

Discussion. Unresolved clashes occur when setting Φ_{max} to less than $\frac{\pi}{8}\,RAD$, which can be attributed to rather unlikely long sequences of energetically unfavorable transformations that have to be performed to attain clash-free conformations. Conversely, setting Φ_{max} to values above $\frac{\pi}{2}\,RAD$ drives conformations away from the crystal structure. Presumably, recovering the global minimum is unlikely after randomly accepted large dihedral rotations have resulted in significant structural changes. Based on these results, this work searches for a favorable setting of Φ_{max} in the range from $\frac{\pi}{8}\,RAD$ to $\frac{\pi}{2}\,RAD$.

6.4.2. Parameterizing Trooper's global minimization mode. This research determined a favorable parameterization for Trooper's global mode by minimizing the protein-ligand complexes of data set NN II (see Section 5.2) using all combinations of the parameter sets $\{\frac{\pi}{8}\,RAD, \frac{\pi}{4}\,RAD, \frac{\pi}{2}\,RAD\}$ for the maximum dihedral rotation parameter (Φ_{max}) and

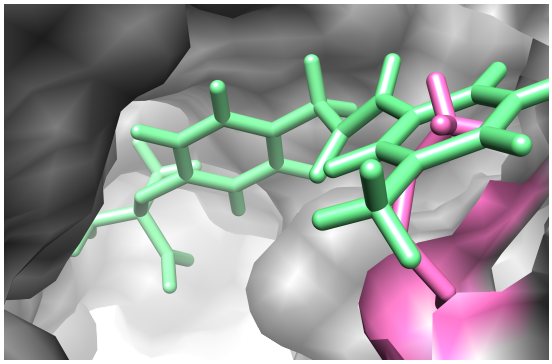


Figure 6.10.A: Binding pocket of a human hemoglobin mutant (1a0u) with crystal pose of ligand of deoxy hemoglobin (1g9v), which clashes with the side chain of Lys99.

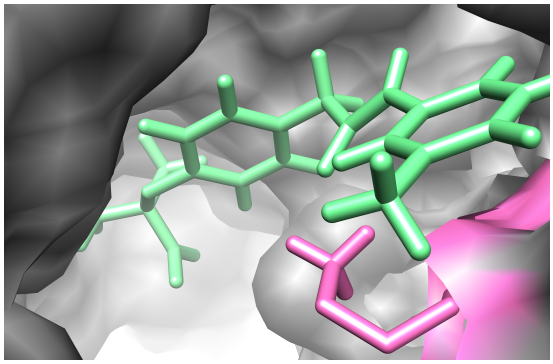


Figure 6.10.B: Optimized binding pocket of a human hemoglobin mutant (1a0u) with crystal pose of ligand of deoxy hemoglobin (1g9v) with resolved clash.

FIGURE 6.10. Clash resolution by appropriately parameterized side chain optimization.

TABLE 6.7. Depending on the choice of the maximum dihedral rotation parameter (Φ_{max}), Trooper manages to resolve clashes between a ligand and amino acid side chains. The tested structures are composed of a ligand and a binding pocket that originate from different protein-ligand complexes, whose pdb codes are specified.

		clash resolved if Φ_{max} is set to (RAD)					
Protein pdb code	Ligand pdb code	$\frac{\pi}{32}$	$\frac{\pi}{16}$	$\frac{\pi}{8}$	$\frac{\pi}{4}$	$\frac{\pi}{2}$	π
1a0u	1g9v	no	no	no	no	yes	yes
1h0w	1ke5	no	no	yes	yes	yes	yes
1pe5	1ke5	no	no	no	yes	yes	yes
1tb0	1oq5	no	no	no	yes	yes	yes
1ob3	1v0p	no	no	no	no	yes	yes
1y2k	1xoq	no	no	no	no	yes	yes
1zzl	1ywr	no	no	no	yes	yes	yes

$\{0.98, 0.96, 0.94, 0.92\}$ for the transformation downscaling factor ($rescale_{trans}$). Also, parameterizations entailing local optimizations were tested. Therefore, no transformation rescaling was performed and the temperature T was set to zero. For all other test runs, T was initially set to 30. Additionally, the number of optimization steps s_{max} was fixed to 15,000 and the SuperTrAmber force field served as an objective function for all test runs. Furthermore, all amino acid side chains with at least one atom closer than 4 $\mathring{A}$ to the ligand were treated as flexible and a cutoff of 8 $\mathring{A}$ was imposed on non-bonded interactions.

TABLE 6.8. Median potential energies of binding pocket conformations resulting from minimizations performed with Trooper utilizing the SuperTrAmber force field as an objective function. For parameterization purposes, different combinations of the maximum dihedral rotation angle (Φ_{max}) and the transformation downscaling factor ($rescale_{trans}$) as well as parameterizations for local optimizations were tested.
The last column specifies whether the median potential energy value in the respective row differs significantly from the one in the first row. The levels of significance are 0.1% (***), 1% (**), and 5% (*). The minus (-) symbol denotes no statistical significance.

$\Phi_{max}\,(RAD)$	$\Phi_{max}\,(DEG)$	$rescale_{trans}$	median $E\frac{kcal}{mol}$	significance
$\pi/2$	90.0	0.98	-182.17	-
$\pi/4$	45.0	0.98	-179.99	-
$\pi/2$	90.0	0.96	-179.44	-
$\pi/8$	22.5	0.98	-177.50	*
$\pi/4$	45.0	0.96	-177.11	-
$\pi/2$	90.0	0.94	-175.81	*
$\pi/2$	90.0	local	-175.80	*
$\pi/4$	45.0	0.94	-175.42	*
$\pi/8$	22.5	0.96	-174.06	**
$\pi/4$	45.0	0.92	-173.58	**
$\pi/4$	45.0	local	-172.75	**
$\pi/2$	90.0	0.92	-171.26	**
$\pi/8$	22.5	0.94	-169.41	***
$\pi/8$	22.5	local	-169.27	**
$\pi/8$	22.5	0.92	-165.76	***

The median force field energy was computed for the set of minimized conformations resulting from each of the tested parameter combinations. Table 6.8 offers a sorted overview of the results. Figure 6.11 shows the decline of Φ_{max} for the set of tested $rescale_{trans}$ factors.

Discussion. Analyzing the top-ranked parameter sets in Table 6.8, this work derives that using global parameters yields a better minimization performance. This discovery is additionally supported by a higher number of outliers produced by local parameterizations, which are not captured by the median energy value.

Furthermore, a gentle freezing of the system with the $rescale_{trans}$ parameter in the range from 0.96 up to 0.98 is evidently favorable. Figure 6.11 provides an explanation for this observation: for $rescale_{trans}$ values below 0.96, the optimization terminates prematurely. This happens as soon as the Φ_{max} parameter is reduced to a value close to zero. However, setting $rescale_{trans}$ to 0.98 would require performing 15,000 steps in every optimization. Otherwise, the cooling phase would be interrupted. As the results in Table 6.8 suggest that a $rescale_{trans}$ factor of 0.96 yields optimization results of equivalent quality, this work sets $rescale_{trans}$ to 0.97 in future experiments. This constitutes a compromise between speed and quality.

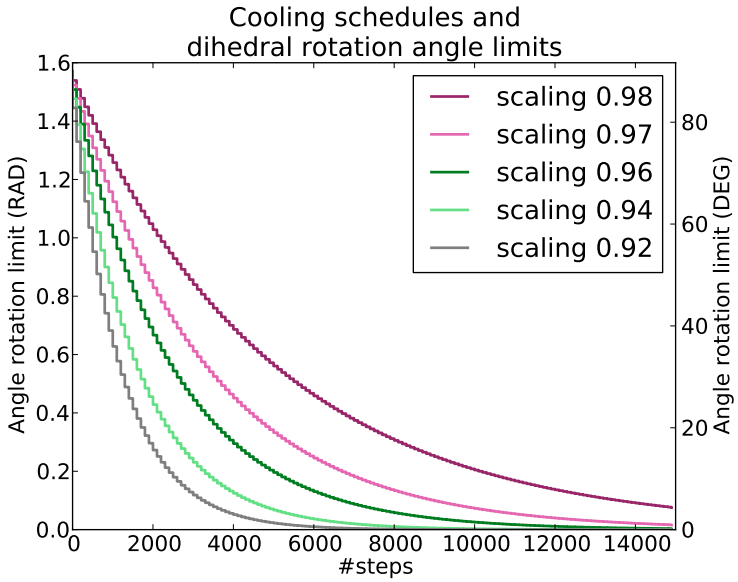


FIGURE 6.11. Effect of different transformation downscaling factors ($rescale_{trans}$) on the maximum dihedral rotation angle (Φ_{max}) observed for an optimization run with 15,000 steps in total, a cooling cycle length of 100 steps, and an initial setting of $\pi/2\,RAD$ for Φ_{max}.

For the Φ_{max} parameter, results suggest that a maximum rotation angle of less than $\frac{\pi}{4}\,RAD$ is detrimental for the minimization performance. Furthermore, it is unclear whether setting Φ_{max} to $\frac{\pi}{2}\,RAD$ is superior to $\frac{\pi}{4}\,RAD$. Still, the experiment described in the previous section suggests that larger rotation angles entail a better clash resolution performance. Therefore, this work sets Φ_{max} to $\frac{\pi}{2}\,RAD$ and names the described settings along with the SA parameterization introduced above Trooper's global parameterization and summarize it in Section 4.8.2.

Trooper is thus fully parameterized for minimizations with the standard SuperTrAmber force field. The following experiments test test whether the early abort acceleration mechanism lowers the quality of the results. Furthermore, an alternative parameterization of the SuperTrAmber force field is assessed.

6.4.3. Emphasizing hydrogen bonds. Focusing on hydrogen bonds could enhance the quality of the optimization's results. To test this hypothesis, this research reran the parameterization experiment described in Section 6.4.2 with the `StateManager` component that implements the SuperTrAmber force field with emphasized hydrogen bond energies (see Sections 3.4.2 and 4.5.5). All parameters and preparatory steps were left unmodified and all combinations of the parameter sets $\Phi_{max} = \{\frac{\pi}{8}\,RAD, \frac{\pi}{4}\,RAD, \frac{\pi}{2}\,RAD\}$ and $rescale_{trans} = \{0.98, 0.96, 0.94, 0.92\}$ were tested.

An overview on the median complex energies resulting from the parameterized optimization runs is provided in Table 6.9. It is evident that the experiment's optimization runs, for which

TABLE 6.9. Median potential energies of binding pocket conformations resulting from minimizations with Trooper employing the SuperTrAmber force field with emphasized hydrogen bond energies as an objective function. For parameterization purposes, different combinations of the maximum dihedral rotation angle (Φ_{max}) and the transformation downscaling factor ($rescale_{trans}$) were tested. The last column specifies whether the median potential energy value of the respective row differs significantly from the one in the first row. The levels of significance are 0.1% (***), 1% (**), and 5% (*). The minus (-) symbol denotes no statistical significance.

$\Phi_{max}\,(RAD)$	$\Phi_{max}\,(DEG)$	$rescale_{trans}$	median $E\frac{kcal}{mol}$	significance
$\pi/4$	45.0	0.98	-209.07	-
$\pi/2$	90.0	0.98	-208.70	-
$\pi/4$	45.0	0.96	-204.79	*
$\pi/8$	22.5	0.98	-204.25	*
$\pi/8$	22.5	0.96	-201.69	**
$\pi/4$	45.0	0.94	-201.43	*
$\pi/2$	90.0	0.96	-200.81	*
$\pi/4$	45.0	0.92	-200.12	**
$\pi/2$	90.0	0.94	-199.76	*
$\pi/2$	90.0	0.92	-197.10	**
$\pi/8$	22.5	0.94	-195.56	***
$\pi/8$	22.5	0.92	-192.06	***

$rescale_{trans}$ was set to 0.96 or 0.98, produced complexes with lower median energies. Apart from that, no clear correlation between the setting of parameter Φ_{max} and the optimizer's energy reduction performance is apparent.

Discussion. Both parameterization experiments suggest that the value of the $rescale_{trans}$ parameter should be close to 0.98. Therefore, all the following experiments set this parameter to 0.97. This facilitates comparisons between the optimization procedures. Further reasons for this choice are discussed in the preceding parameterization experiment in Section 6.4.2.

For the Φ_{max} parameter, it is apparent that values of $\frac{\pi}{2}\,RAD$ and $\frac{\pi}{4}\,RAD$ yield equivalent results, under the pre-condition that the value of $rescale_{trans}$ is within the favorable value range. For the same reasons discussed in Section 6.4.2 , the experiment sets Φ_{max} to $\frac{\pi}{2}\,RAD$. The impact of this parameterization, in combination with the tested derivative of the SuperTrAmber force field, is assessed in Section 6.4.5.

6.4.4. Accelerations with the early abort-mechanism. To accelerate optimizations,this research parameterized the early abort-mechanism, which is described in Section 4.5.4. Therefore, this work reran the minimization experiment described in the previous section with the standard global parameterization (see Section 4.8.2). Energy curves were plotted and analyzed for all optimization runs. Typically, they initially exhibit a steep gradient. However, from the 8,000$^{\text{th}}$ step onwards, most minimization runs reduce their respective molecular system's energy

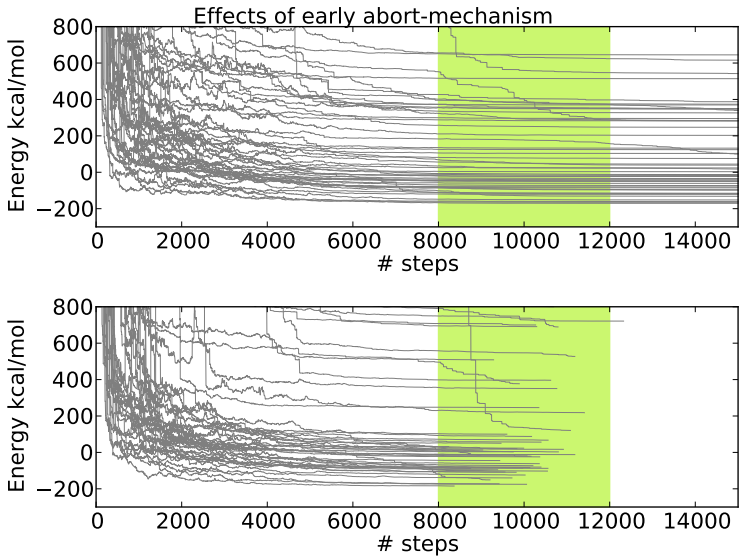


FIGURE 6.12. Force field energy of binding pocket conformations during optimization runs with Trooper. Its standard global minimization mode results in energy curves shown in the upper depiction. Switching the early abort-mechanism on produces energy curves shown in the lower depiction. In the highlighted regions, the gradient of most energy curves is close to zero. This triggers the early abort-mechanism.

by less than $5\,kcal/mol$, as highlighted in Figure 6.12. Consequently, the author activated the early abort-mechanism and set the maximum number of consecutively rejected states ($ea_{rej_{max}}$) to 200 and the minimal absolute energy reduction ($|ea_{red_{min}}|$) to 0.2 $kcal/mol$.

The experiment was rerun with these settings and the resulting energy curves were plotted (see Figure 6.12). These curves suggest that almost all minimization runs terminate in the region between the 8,000$^{\text{th}}$ and the 12,000$^{\text{th}}$ step. Furthermore, it is visible that the energy curves of most minimization runs exhibit virtually no slope before the optimization terminates. However, a few cases exist in which the energy values are still significantly reduced shortly before the termination.

Discussion. The energy curves in Figure 6.12 suggest that, in most runs, virtually no energy minimization takes place from the 9,000$^{\text{th}}$ step onwards. Thus, the optimization either detects the funnel of the energy function's global minimum, and thereafter stays in it, or is unable to overcome the energy barriers of a local minimum. Figure 6.11 provides one possible explanation for the latter case. In the late stage of the optimization, the maximum dihedral rotation parameter (Φ_{max}) is far below 10°. This value could be too low in some critical cases. This hypothesis could be verified by implementing and testing a dynamic reheating scheme.

In summary, the determined parameter settings shorten most optimization runs by approximately 33%. The following experiment assesses whether the optimization's quality is affected by these accelerations.

6.4.5. Optimizing wobbled flexible side chains. Trooper's performance was assessed and externally evaluated on the wobbled amino acid side chains in the binding pockets of data set ADS VIII (see Section 5.1.4). For this, this experiment employed Trooper's global minimization mode (*standard method*), its accelerated version featuring the early abort-mechanism (*fast method*), and its version that emphasizes hydrogen bonds (*strong H-bond method*). For external comparisons, MOE and YASARA were tested (see Sections 3.9.2 and 3.9.1). In all binding pockets, this experiment allowed dihedral bond rotations for all amino acid side chains that contain at least one atom that is closer than $4\,\mathring{A}$ to any ligand atom. The ligand itself remained rigid during the optimization. Furthermore, a cutoff of $8\,\mathring{A}$ was introduced for non-bonded interactions.

For performing minimizations with MOE, this experiment followed the specific standard procedure described in Section 3.9.2. The boundaries of the switching function interval were set to $8\,\mathring{A}$ and $10\,\mathring{A}$. This experiment followed the specific standard procedure described in Section 3.9.1 for YASARA. Here, a cutoff of $8\,\mathring{A}$ was imposed on non-bonded interactions.

The standard global parameterization, as specified in Section 4.8.2, was employed for Trooper. To assess the quality of the results, the RMSD (see Section 3.2) was computed between the optimized and crystal structures using the in-house tool. All atoms of amino acid residues with flexible side chains, including the respective backbone atoms, were considered in the calculations.

An overview on the experiment's results is provided in Figure 6.13. For drawing it, the wobbled binding pockets of data set ADS VIII were binned according to their initial RMSD to the respective crystal structure. Subsequent to the optimization procedures, the experiment computed the mean binding pocket RMSD of the minimized structures for the members of each bin and for each tested method. Furthermore, this experiment employed Welch's t-test (see Section 3.10) to assess, for each tested method, whether the mean RMSD values obtained with it differ significantly from those resulting from minimizations with Trooper's standard method.

Discussion. As apparent in Figure 6.13, Trooper's standard global mode guides wobbled side chains with an initial RMSD above $0.5\,\mathring{A}$ closer to the respective crystal structure conformation ($p \leq 0.01$). For the fast method and the strong H-bond method, this research observes no significantly different results for all but one bin. Thus, there is no point in employing the strong H-bond method, as there is no experimental support for the parameterization of its hydrogen bond potential. In contrast, it is recommendable to utilize the fast method, as it reduces the number of optimization steps by about 30% compared to the standard method.

It would be desirable to improve Trooper's performance on conformations with an initial RMSD below $0.5\,\mathring{A}$. Here, unfavorable results routinely occurred in binding pockets with erroneously protonated functional groups. For example, assigning protons to the δ and ε nitrogens

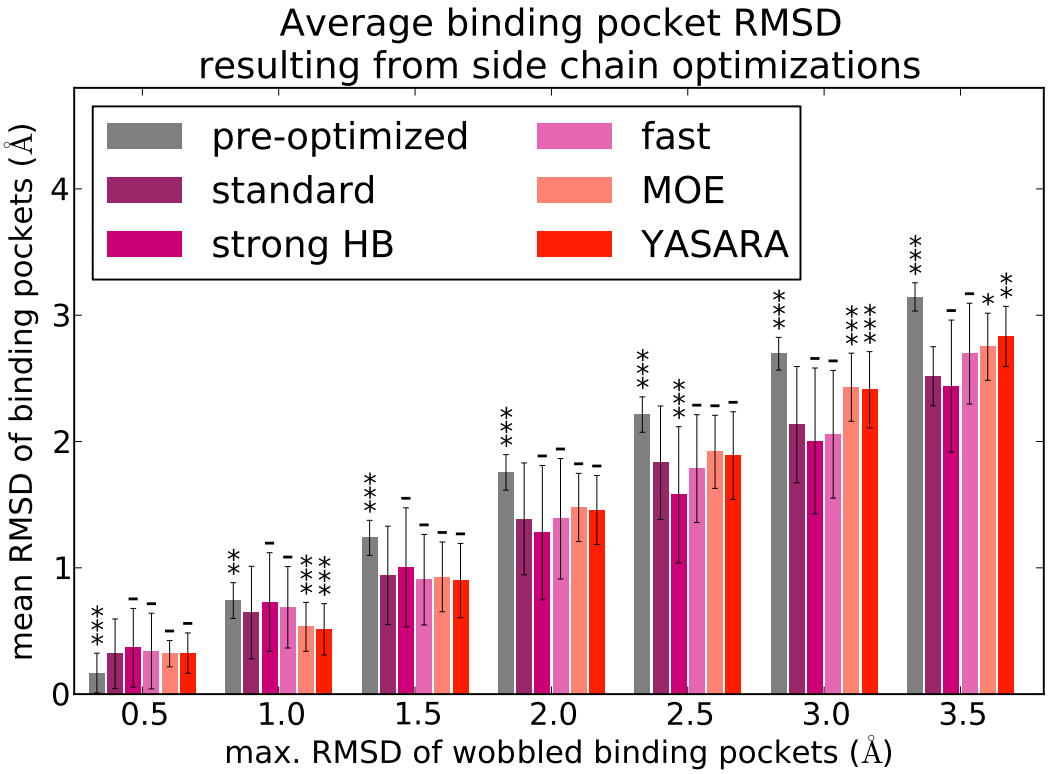


FIGURE 6.13. Average RMSD values of binding pocket conformations before and after optimization with MOE, YASARA, and Trooper. Randomly wobbling selected amino acid side chains produced the initial binding pocket conformations. Conformations are binned according to their RMSD to the respective crystal structure. An optimized conformation was associated to the bin of its initial conformations.
For Trooper, experiments with standard parameters for global optimizations (*standard*), activated early abort-mechanism (*fast*), and a SuperTrAmber version with emphasized hydrogen bond energies as an objective function (*strong HB*) were conducted.
Symbols above a method's error bar indicate whether the mean RMSD value of its produced conformations differs significantly from the one produced by Trooper's standard method. The levels of significance are 0.1% (***), 1% (**), and 5% (*). The minus (-) symbol denotes no statistical significance.

of histidines typically lead to outliers. Thus, great care must to be taken when generating protonation states. The SuperTrAmber force field itself is another reason deviations from crystal structures occur. As visible in Figure 6.14, these typically have lower potential energies than wobbled structures. However, often there exist accessible conformations with an even lower energy. Hence, the objective function has to be adapted rather than the optimization procedure.

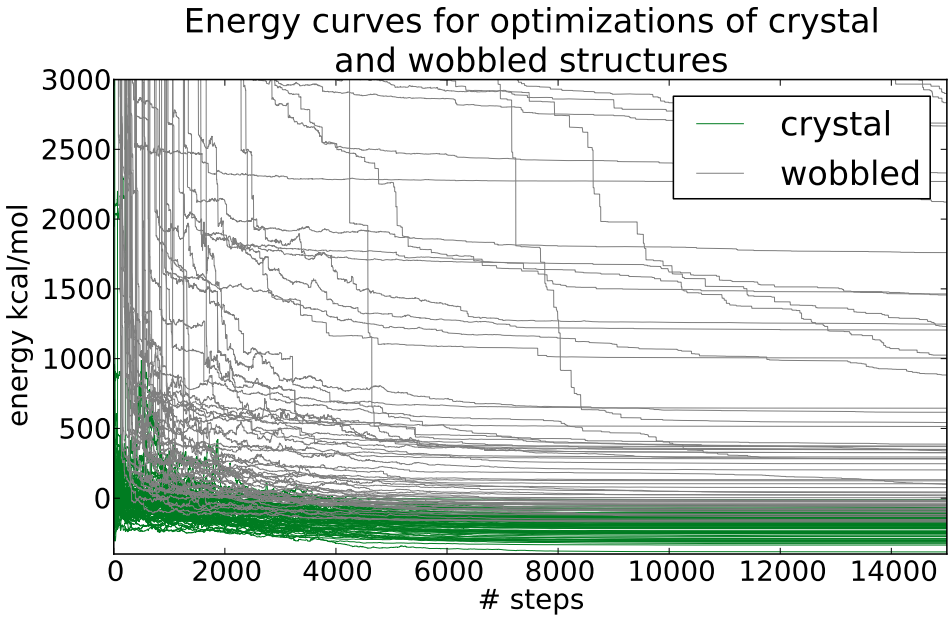


FIGURE 6.14. Force field energy of binding pocket conformations during optimization runs. Energy curves for runs with wobbled and crystal structures are shown in gray and green, respectively.

Alternatively, a parameterization for structures with low initial energy values could be developed. For this, the temperature could initially be set to zero and the movement parameters could be set to lower initial values.

Furthermore, for wobbled binding pockets with an initial RMSD between 0.5 $\AA$ and 1.0 $\AA$, Trooper's standard global mode is outperformed by YASARA and MOE ($p \leq 0.001$). Conversely, Trooper performs better than these external tools on binding sites with an initial RMSD above 2.5 $\AA$ ($p \leq 0.05$). For the rest of the tested structures, there are no significant differences between the tested methods. Thus, over the whole range of considered structures, all methods perform more or less on par.

Overall, this experiment provides evidence that all tested methods guide amino acid side chains towards their respective crystal structure conformation. For this, taking backbone flexibility into account proved to be unnecessary. Further experiments must be performed to determine whether this observation holds true for structures with varying backbone conformations and natural side chain conformations.

6.4.6. Force field comparison. To ascertain that selecting the MMFF94s force field (see Section 3.4.3) as an objective function has no major impact on the optimization results, the data set ADS VIII (see Section 5.1) was minimized with MOE. In this experiment, MMFF94s was

selected as the objective function instead of the AMBER94 force field. All other parameters and settings remained unchanged and can be looked up in Section 6.4.5.

The average RMSD to the crystal structure of structures resulting from minimizations with MOE and MMFF94s amounts to 1.039 $\AA$ with standard deviation $\sigma = 0.713$. The same experiment performed with AMBER94 yields an average RMSD of 1.078 $\AA$ with a standard deviation of $\sigma = 0.743$. According to Welch's t-test, the null hypothesis that these two results do not differ significantly cannot be refuted ($p \simeq 0.26$).

Discussion. There is no evidence that choosing MMFF94s instead of AMBER94 for the optimizations enhances the quality of the resulting structures. As minimization with SuperTrAmber performs at least as well as those carried out with MOE and AMBER94, there is no need to consider employing MMFF94s as an alternative to SuperTrAmber.

6.4.7. Optimizing transferred amino acid side chains. The following experiment assessed whether Trooper optimizes naturally occurring amino acid side chain conformations. At the same time, it tested whether considering solely dihedral rotation degrees of freedom suffices for this task. This research therefore evaluated the optimization performance of Trooper's global mode, MOE, and YASARA on amino acid side chain conformations occurring in protein crystal structures. To accomplish this, the experiment minimized the binding pockets of the 686 protein-ligand complexes of data set NN II. This set was created by transferring amino acid side chain conformations of binding pockets taken from the Astex Non-Native Set [**198**] to sequentially identical template structures of the Astex Diverse Set [**3**]. Section 5.2.1 provides a detailed description of this procedure.

For selecting flexible amino acid side chains, this experiment proceeded as the preceding one described in Section 6.4.5. Therein, Trooper's standard minimization with global parameterization and the experimental setups of the external tools is also described.

Prior to and after the minimization procedures, the RMSD of the flexible amino acids was measured against their respective template structures with the in-house tool. Figure 6.15 shows the results, which were binned as described in Section 6.4.5.

Discussion. Trooper drives naturally occurring amino acid side chain conformations towards their respective crystal conformation. This hypothesis holds true if the RMSD of the initial conformation to the crystal structure is larger than 0.5 $\AA$ and smaller than 1.5 $\AA$ ($p < 0.05$). For further assessing this observation, it would be interesting to enrich the tested set with structures that have binding site RMSD values above 1.5 $\AA$. Currently, the set contains only eight of these. However, to the knowledge of the author, the Astex Non-Native Set [**198**]—of which the current test set is formed—is the largest collection of sequentially identical protein binding sites publicly available.

Furthermore, the results demonstrate that Trooper's parameters are not overfitted as it yields similar results, irrespective of whether it is provided with natural or wobbled conformations, though it was trained on the latter. This observation also holds true for conformations with an

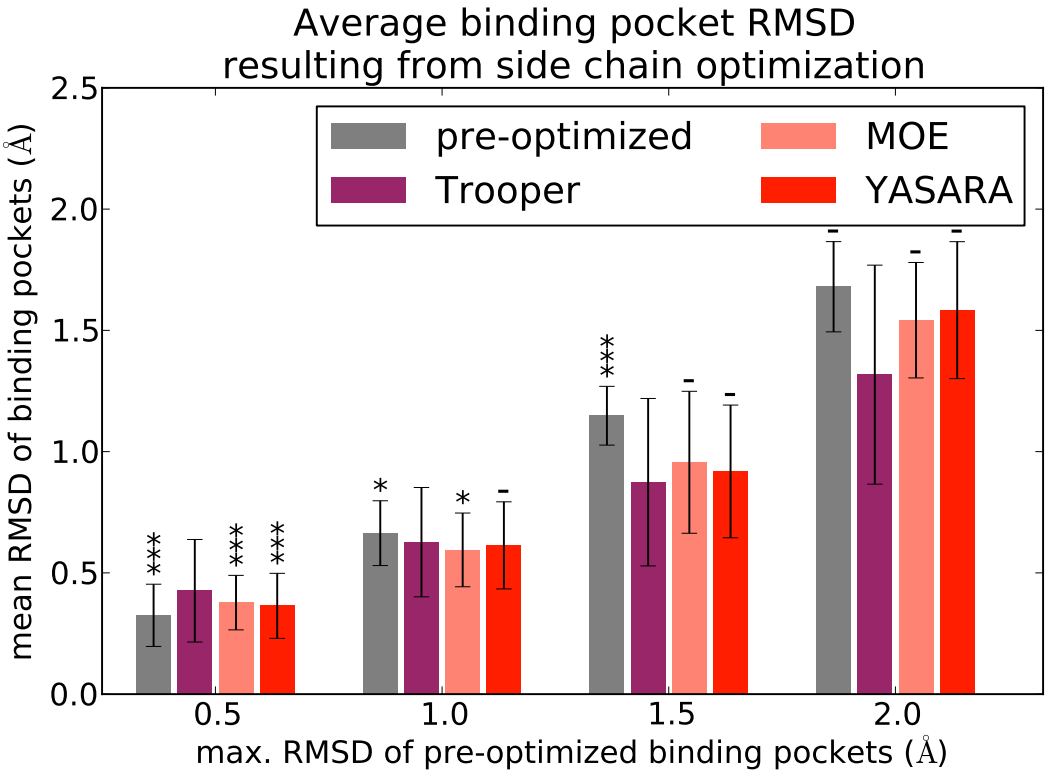


FIGURE 6.15. Average RMSD values of binding pocket conformations before and after optimization with Trooper, MOE, and YASARA.
Side chain conformation transfers from binding pockets to corresponding sequentially identical template structures produced the initial conformations, which were binned according to their RMSD to the respective template structure's original conformation. Optimized conformations were associated with the bins of their initial conformations.
Symbols above a method's error bar indicate whether the represented mean RMSD value differs significantly from the one produced by Trooper. The levels of significance are 0.1% (***), 1% (**), and 5% (*). The minus (-) symbol denotes no statistical significance.

initial RMSD to the crystal structure below 0.5 $\mathring{A}$. For these, a slight deterioration of RMSD values is again observed, which is discussed in detail in Section 6.4.5.

For large parts of the tested data set, Trooper evidently performs on par with MOE and YASARA. Interestingly, the optimized amino acid side chain conformation oftentimes slightly differ from crystal structure conformations in terms of binding angles. In contrast to this work's method, YASARA optimizes this degree of freedom, which is found to be apparently negligible.

Overall, this experiment indicates that amino acid side chain optimizations yield favorable results when provided with natural conformations with more than minor deviations from the

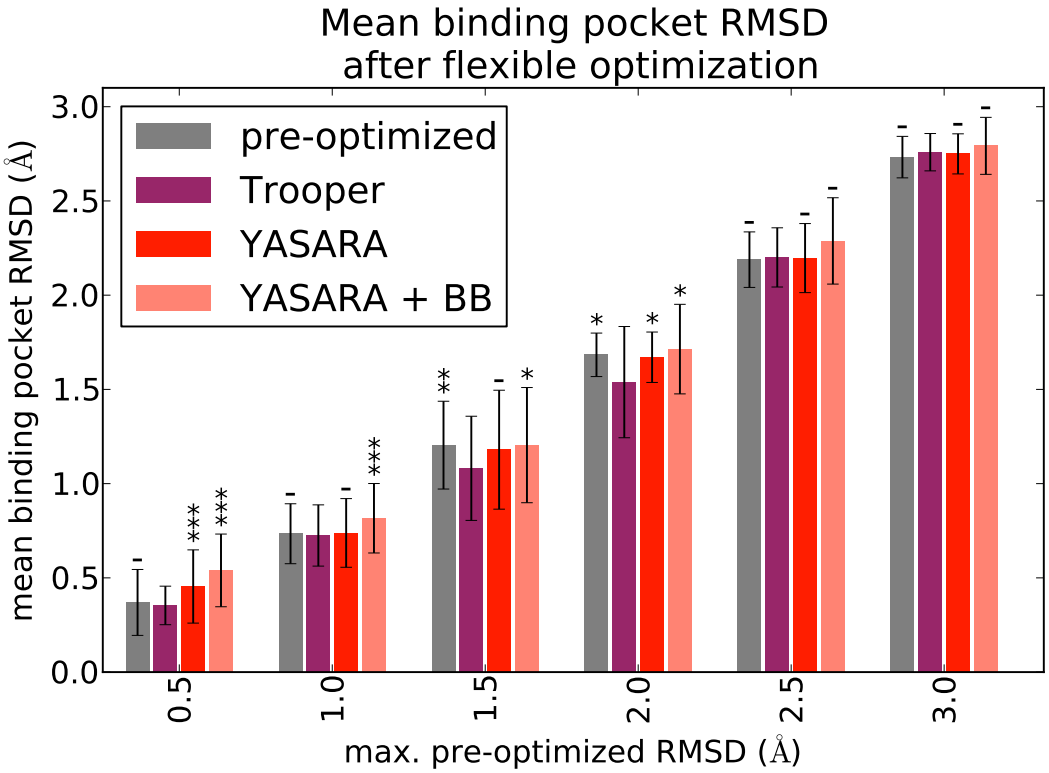


FIGURE 6.16. Average RMSD values of binding pocket conformations before and after optimization with Trooper and YASARA. For the latter, a second experiment was conducted that included backbone atoms in the minimization procedure (*YASARA + BB*).
Initial conformations were taken from the Astex Non-Native Set and binned according to their RMSD to the respective template structure. Optimized conformations were associated to the bins of their initial conformations.
Symbols above a method's error bar indicate whether the represented mean RMSD value differs significantly from the one produced by Trooper. The levels of significance are 0.1% (***), 1% (**), and 5% (*). The minus (-) symbol denotes no statistical significance.

respective crystal structure. Apparently, no additional degrees of freedom are required at this point. To further explore the limits of side chain optimization methods, aberrations in the backbone have to be introduced.

6.4.8. Optimizing the Non-Native Data Set. This work assessed whether side chain optimizations cope with varying backbone conformations. To achieve this, the research minimized the amino acid side chains of all binding pockets of the Astex Non-Native Set [**198**] with Trooper and YASARA and again proceeded as in the previous experiment, described in Section 6.4.5, for

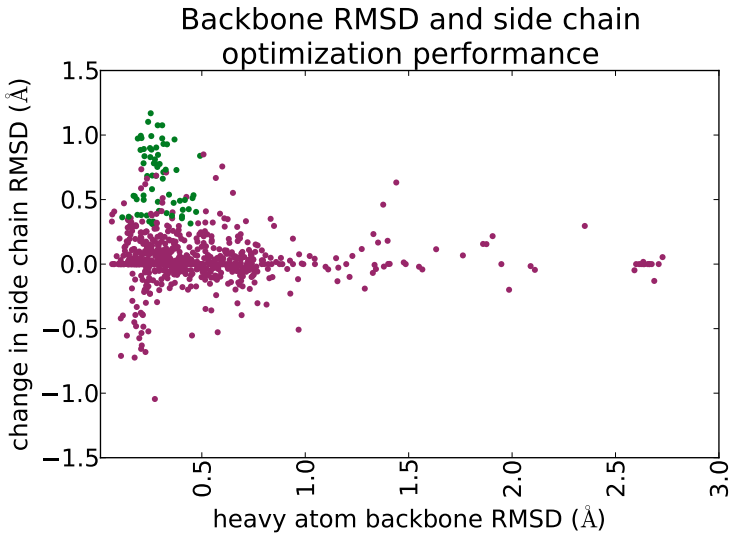


FIGURE 6.17. Initial binding pocket backbone RMSD plotted against the change in amino acid side chain RMSD relative to the respective crystal structure conformation. Green dots represent complexes with backbone variations that entail clashes between side chains and the ligand. Most often, only large side chain flips can resolve these clashes.

selecting flexible side chains. This section also specifies parameterizations and set-up procedures for the employed minimization methods. Additionally, the global parameterization of Trooper is listed in Section 4.8.2 and that of YASARA in Section 3.9.1.

The current experiment added a YASARA minimization run to test the effects of incorporating backbone flexibility. For this, the flexible regions of binding pockets were extended. They included all amino acids containing at least one atom with distance lower than 7 $\AA$ to any ligand atom. The backbone in this region was flexible during optimizations.

Following the minimizations, this work employed the in-house tool for measuring RMSDs of flexible amino acids against their respective template structures. A binning scheme, as described in Section 6.4.5, was applied for producing Figure 6.16, which shows the results. Furthermore, the author visually inspected complexes with an initial average binding pocket backbone RMSD of 0.5 $\AA$ and less for which the optimization drove side chain conformations further away from their respective crystal structure conformation. This inspection determined that slight backbone variations can entail clashes between side chains and the ligand. Figure 6.17 shows that this observation applies to the majority of the analyzed cases. Only large side chain flips could resolve these clashes.

Discussion. On average, none of the tested methods drive flexible amino acid side towards their respective crystal conformation. Small backbone variations often lead to conformations that

side chain optimizations cannot bring closer to their respective crystal structure. This is partly due to the rigid ligand. However, if the ligand was flexible, the optimization would most probably move it away from its crystal pose, which is undesired. From this, this work concludes that with the given degrees of freedom, aberrations in backbone conformations cannot be compensated for. This experiment thus defines the limit of applicability of Trooper. Furthermore, it contributes to elucidating the application range of amino acid side chain optimizations.

Incorporating backbone flexibility in the optimization procedure changes the results produced by YASARA for the worse. This observation suggests that either the objective function or the minimization method is inappropriate. The former hypothesis can be tested by performing optimization runs with YASARA and a wider range of force fields. Still, this work considers that the latter hypothesis is closer to the truth, as the force field comparison in Section 6.4.6 suggests.

As a next step, the author proposes to identify prerequisites for successfully optimizing structures with backbone variations. This could start with an analysis of structures that YASARA successfully optimizes. Considering multiple pre-determined backbone conformations would be another option. This way, potentially flexible segments of the protein backbone could be represented by an ensemble of pre-determined sound conformations. Also, softened potentials and cooling schedules that only moderately enhance the temperature parameter when clashes are present could be tested.

6.5. Accelerating Amino Acid Side Chain Conformation Optimizations

Runtimes are a critical aspect in application scenarios for amino acid side chain optimizations. As pointed out in Section 3.1.3, user productivity is enhanced if process response times are interactive. Conversely, slow response times demoralize users. This has to be avoided when lead optimizations are performed that test alternatives to certain amino acids in binding pockets. Furthermore, in large-scale optimization scenarios, energy efficient algorithms reduce the cost of scientific research.

The following sections therefore determine runtimes of different CPU-based amino acid side chain minimization procedures and assess the speedup that the utilization of GPUperTrooper engenders. GPUperTrooper is tested in its global mode. Thus, it uses the advanced GPU-based algorithm, which Section 4.4 described. The following sections employ GPU-specific terminology and therefore, reading Section 1.5 is recommended before continuing.

6.5.1. Runtime of amino acid side chain minimizations. This experiment assessed the runtime of Trooper, MOE, and YASARA for minimizing amino acid side chains in the binding pockets of the 748 protein-ligand complexes of data set ADS VIII (see Section 5.1). As in previous experiments, the crystal structure of the respective native ligand was placed in the binding pocket and remained rigid throughout the optimization. Flexible amino acid side chains were selected, as described in Section 6.4.5. That section also specifies the parameters and settings for all employed minimization procedures. In addition, the experiment tested the runtime of Trooper with activated early abort-mechanism (see Section 4.5.4). All runtimes were

TABLE 6.10. Median and mean runtime in seconds, along with standard deviation (σ) for optimizing binding pocket conformations with YASARA, MOE, and Trooper. For the latter, the mean runtime of minimizations conducted with the early abort-mechanism is specified as well.

tool	mean (s)	σ	median (s)
Trooper	142.72	38.68	140.94
Trooper w/ early abort	92.67	28.89	89.02
MOE	76.42	88.34	49.00
YASARA	160.09	666.11	60.00

measured on a single core of the x86 CPU of test machine III (see Section 5.4). This CPU ranked, at the time of writing, among the fastest available. The resulting average and median runtimes are listed in Table 6.10.

Discussion. Judging from the measured mean runtimes, the speed of all methods is in the same range. Compared to Trooper, YASARA consumes 173% runtime and MOE 83%. At the same time, YASARA and MOE produce a large amount of runtime outliers. This is indicated by differing mean and median runtime values, as well as high standard deviations. In contrast, the runtime of Trooper is stable.

As YASARA employs an adaptive SA algorithm with a variable number of steps, it is clear that the variance of runtimes is high. MOE employs a gradient-based approach, which terminates if the slope is below a given energy value. This can entail longer runtimes if a system does not converge well.

However, all procedures exhibit runtime behaviors that prohibit a productive usage in scenarios where responsiveness and interactivity (as defined in Section 3.1.3) are of importance. Also, processing larger data sets is overly time consuming. The following sections test GPUperTrooper, the accelerated version of Trooper.

6.5.2. Quality of GPU-based optimizations. The following experiment assures that the results reported in Section 6.4.5 also apply to GPUperTrooper. Therefore, this experiment compared the quality of results produced by Trooper's and GPUperTrooper's global mode (see Section 4.8.2) when optimizing amino acid side chains. To achieve this, this experiment minimized those 491 binding pockets of data set ADS VIII (see Section 5.1) that resulted from $0, 10{,}000, 20{,}000, 50{,}000, 100{,}000$, and 500,000 wobbling operations. Parameterization and flexible side chain selection procedures conform to those described in Section 6.4.5.

On average, the RMSD of optimized binding pockets to their respective crystal structure amounts to $1.13\,\mathring{A}$ with a standard deviation of $\sigma \simeq 0.73$ for Trooper and $1.12\,\mathring{A}$ with standard deviation $\sigma \simeq 0.72$ for GPUperTrooper. These figures strongly support the hypothesis that the results of the GPU- and CPU-based algorithms are equally distributed. Therefore, this work regards them as equal in terms of their optimization performance. The following compares their runtimes and assesses whether it is worthwhile to apply GPUperTrooper.

TABLE 6.11. Mean runtimes in seconds, along with standard deviations (σ) for optimizing binding pocket conformations. Figures for YASARA, MOE, Trooper, and GPUperTrooper with activated early abort-mechanism are given. Measurements on CPU were conducted on a single core. The last two columns specify the speedup (S_{GPU}) and the economic efficiency (E_{eco}) of GPUperTrooper relative to the method named in the respective row.

tool	runtime(s)	σ	S_{GPU}	E_{eco}
GPUperTrooper	0.79	0.18	-	-
Trooper	92.67	28.89	117.0	9.8
MOE	76.42	88.34	96.5	8.0
YASARA	160.09	666.11	202.1	16.8

6.5.3. Runtime of GPUperTrooper optimizations. This research measured runtimes of GPUperTrooper's global mode (see Section 4.8.2) for amino acid side chain minimizations. For this, GPUperTrooper optimized the binding pockets of data set ADS VIII (see Section 5.1) in the presence of the respective rigid ligand in its crystal conformation. Additional optimization runs were conducted with activated early abort-mechanism (see Section 4.5.4). Furthermore, this experiment tested the GPU-based minimization algorithm benchmarked in Section 6.3, which GPUperTrooper's local mode employs. The following refers to it as the *basic GPU algorithm* and refers to GPUperTrooper's global mode algorithm as the *advanced GPU algorithm*.

All parameters and settings corresponded to those employed in the runtime experiment for CPU-based methods described in Section 6.5.1. Runtimes of CPU-based methods are also taken from this section. The experiment used the GeForce GTX680 GPU of test machine II (see Section 5.4). All but the runtimes of the basic GPU algorithm are shown in Figure 6.18. For the basic GPU algorithm, the experiment measured an average runtime of 1.8 s per minimization run without early abort-mechanism. With the same setup, the advanced GPU algorithm consumes 1.1 s on average. Furthermore, Table 6.11 lists speedup factors and economic efficiencies with respect to the CPU-based methods. The two measures are introduced in Section 3.1.

Discussion. The results show that the advanced GPU-based algorithm is about 40% faster than its basic version. Compared to external tools, the advanced algorithm exhibits significantly less variance in the measured runtimes. Furthermore, it performs amino acid side chain minimizations about two orders of magnitude faster than all tested CPU-based methods. In addition, it saves at least 87.5% electric energy according to the economic efficiency.

When minimizing ligands only, the experiment measured higher speedup factors relative to the external CPU-based methods (see Section 6.3.3). The data suggests that the external tools treat intramolecular non-bonded interactions more efficiently in the current experiment's scenario.

In conclusion, GPUperTrooper's global mode minimizes amino acid side chains of binding pockets in less than 0.8 s on average. This allows for an interactive workflow (as defined in

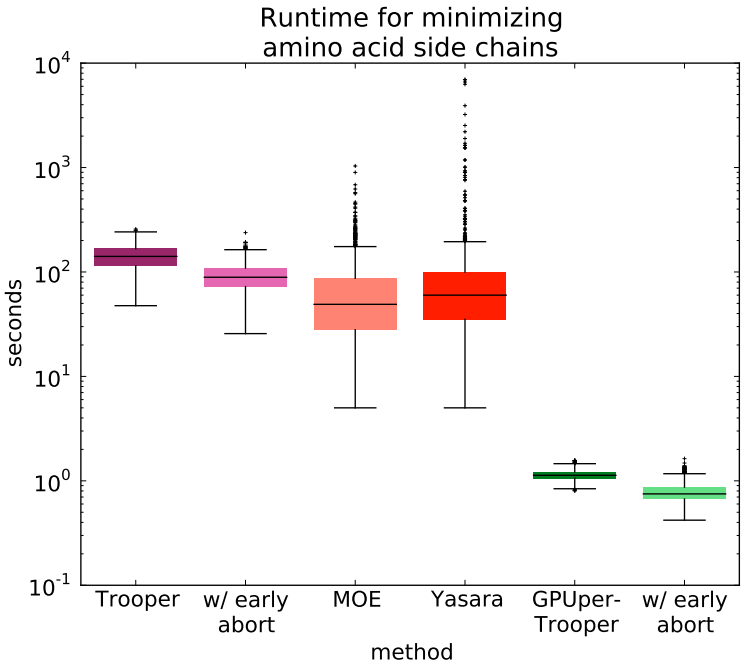


FIGURE 6.18. Median runtimes for optimizing binding pocket conformations with MOE, YASARA, Trooper, and GPUperTrooper. For the latter two, the standard versions and those accelerated with the early abort-mechanism (*w/ early abort*) were tested. Boxes span the 25^{th} to the 75^{th} percentile of the data range, while whiskers extend to data points within 1.5 times the data range represented by the boxes. Remaining values are outliers and depicted by the small (+) symbols.
The y-axis, which specifies the runtime in seconds, is logarithmically scaled.

Section 3.1.3) in drug development tools. Further, cost- and time-efficient optimizations of larger molecular libraries are now feasible. With 100 accelerator cards, GPUperTrooper minimizes 1,000,000 binding pockets in less than three hours. It can therefore be applied in combination with even the fastest docking methods, such as TrixX [**102**] and PhDOCK [**7**], without significantly enhancing their runtimes. Furthermore, GPUperTrooper could significantly speed up protein engineering. Bordner and Abagyan [**107**] applied ICM in this scenario and measured an average runtime of 720 s on a single core of a 1.3 GHz AMD Athlon CPU. Also, Miura et al. [**29**] applied the benchmarked minimization of MOE for predicting site-directed mutagenesis effects. The data showed that GPUperTrooper performs on par with MOE in terms of quality, yet it is two orders of magnitude faster.

For future large-scale cluster use, it would be enlightening to test whether GPUperTrooper scales well on different GPU generations. The next experiment is centered around this question.

TABLE 6.12. Mean runtime in seconds for minimizing binding pocket conformations with GPUperTrooper on different NVIDIA GPUs. The table specifies the respective processor's number of CUDA cores (*#cores*), the clock frequency of the multiprocessors (*core clock*), and the nominal performance in teraFLOPs (*TFLOPS*) along with the host test machines.

test machine	GPU (chipset)	#cores	core clock	TFLOPS	runtime (s)
I	Tesla C1060 (Tesla)	240	1,296 MHz	0.936	2.48
V	Quadro 4000 (Fermi)	256	475 MHz	0.486	3.17
II	GeForce GTX680 (Kepler)	1,536	1,058 MHz	3.090	1.13
IV	Tesla K20 (Kepler)	2,496	706 MHz	3.520	1.41

6.5.4. Scaling behavior of GPUperTrooper. This experiment tested the scaling behavior of GPUperTrooper's global mode by rerunning the runtime evaluation experiment described in the preceding section. Therefore, test machines I, II, IV, and V (see Section 5.4) were employed, as they host GPU devices with chipsets of the Tesla, Fermi, and Kepler generations.

To produce comparable results, the early abort-mechanism was switched off. This work lists the measured runtimes, along with relevant technical specifications of the test machines, in Table 6.12.

Discussion. GPUperTrooper runs properly on all NVIDIA GPU generations produced since 2007 up to the writing of this work. This research measures the lowest runtimes on GPUs of the recent Kepler generation. The performance on the most recent GPU, the Tesla K20, which is lower than that of the GeForce GTX680, is surprising. Neither the nominal performance in teraFLOPs, nor the number of cores in combination with the core clock explains this behavior.

For elucidating it, the author proposes to design a benchmarking protocol to assess the speed of the test machines. This would capture the performance differences entailed by their heterogeneous hardware setups. Based on these findings, a more reliable predictor of the runtime behavior could be established.

Further insights could arise from in-depth profiling of GPUperTrooper; specifically the occupancy of the GPU multiprocessors would be of interest. This occupancy could be low, as computing the force field energy of binding pockets with GPUperTrooper requires a comparatively small number of arithmetic operations. This could prove to slow larger GPUs down as their architecture requires a large pool of operations in order to hide memory latencies.

6.6. Optimizing Entire Binding Pockets

The previous sections were centered on parameterizing Trooper for the exclusive optimization of either amino acid side chains or ligands. In contrast, the following experiments concentrate on minimizations that take ligand and amino acid side chain flexibility into account at the same time. Additionally, constraints for the ligand are introduced. This permits a coarse definition of binding pockets.

TABLE 6.13. Median and mean potential energies of protein-ligand complex binding pockets resulting from minimizations with Trooper employing the standard SuperTrAmber force field as an objective function. Both amino acid side chains and the ligand were flexible during the minimization. For parameterization purposes, different combinations of the maximum molecular rotations angle (ρ_{max}) and the maximum length of the translation vector [$max(|\vec{t}|)$] were tested.
The last column specifies whether the median potential energy value in the respective row differs significantly from the one in the first row. One-tailed Mann-Whitney U tests were performed. The minus (-) symbol denotes no statistical significance.

$max(\|\vec{t}\|)\,(\mathring{A})$	$\rho_{max}\,(RAD)$	mean $E\frac{kcal}{mol}$	median $E\frac{kcal}{mol}$	significance
0.4	0.39	-116.73	-196.18	
0.2	0.09	147.37	-194.15	-
0.2	0.39	233.08	-193.66	-
0.4	0.19	-125.63	-193.56	-
0.1	0.39	-100.22	-193.26	-
0.2	0.19	158.83	-193.13	-
0.1	0.19	-117.92	-192.09	-
0.4	0.09	-113.72	-191.59	-
0.1	0.09	311.40	-189.24	-

6.6.1. Parameterization. The first experiment determines a favorable parameterization for Trooper's global mode for entire binding pockets. For this, it minimized the 340 wobbled binding pockets of data set ADS IX (see Section 5.1). Parameters were taken from the set $\{0.1\,\mathring{A}, 0.2\,\mathring{A}, 0.4\,\mathring{A}\}$ for the maximum length of the translation vector [$max(|\vec{t}|)$] and $\{\frac{\pi}{8}\,RAD, \frac{\pi}{16}\,RAD, \frac{\pi}{32}\,RAD\}$ for the maximum molecular rotations angle (ρ_{max}). This experiment treated all amino acid side chains containing atoms closer than $4\,\mathring{A}$ to any ligand atom as flexible and imposed a cutoff of $8\,\mathring{A}$ around the flexible region. Additionally, it constrained ligand atoms to a maximum deviation of $6.5\,\mathring{A}$ from their initial location. For all other parameters, this experiment employed the standard settings for global optimizations specified in Section 4.8.2. In this way, all protein-ligand complexes were processed. Table 6.13 lists the resulting mean and median force field energies, along with significance levels. For computing these, one-tailed Mann-Whitney U tests (see Section 3.10) were performed.

Discussion. The large deviations between the mean and median energy values of optimized complexes signify a non-normal distribution of the results. Apparently, a certain proportion of structures remain on a comparatively high energetic level depending on the selected parameter combination. The mean energy values thus incorporate information on how severely the force field energies of outliers deviate from the bulk of produced results. Therefore the effect of the parameter choice is judged based on the resulting mean energy values. Setting $max(|\vec{t}|)$ to $0.4\,\mathring{A}$ is thus a sound choice. For the selection of parameter ρ_{max}, the results provide only

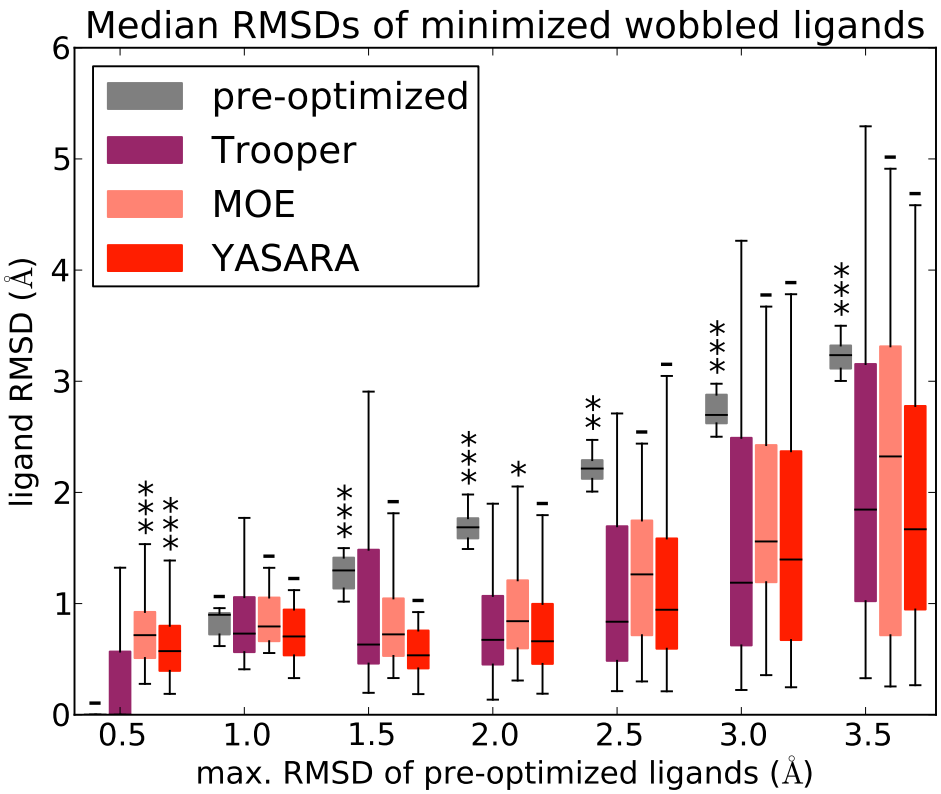


FIGURE 6.19. Median RMSD values of ligands optimized in binding pockets with flexible amino acid side chains. The results are binned according to the RMSD values of the pre-optimized ligand structures. Boxes span the 25^{th} to the 75^{th} percentile of the data range while whiskers extend to data points within 1.5 times the data range, represented by the boxes. Symbols above the whiskers indicate whether the median of the RMSD values produced by Trooper and the respective whisker's method differ significantly. The levels of significance are 0.1% (***), 1% (**), and 5% (*). The minus (-) symbol denotes no statistical significance.

little help. The parameter was set to $0.39\,RAD$, as the exponential downscaling scheme would otherwise limit rotations to comparatively small angles after a few thousand steps. The following experiment evaluates whether Trooper's thus parameterized global mode successfully optimizes ligands in the presence of flexible amino acid side chains.

6.6.2. Optimizations. To evaluate Trooper's global mode, this experiment compared its performance to that of MOE and YASARA. To evaluate the ligands and binding pockets of data set ADS IX (see Section 5.1) were minimized. Trooper was parameterized, as specified in the preceding section. Further parameters correspond to the standard settings for the global mode specified in Section 4.8.2. For MOE and YASARA, the experiment defined a flexible region that

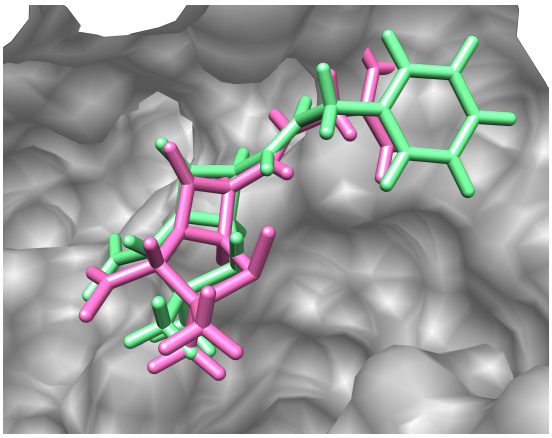


Figure 6.20.A: Wobbled binding pocket of penicillin G acylase (*1gm8*). The fuchsia colored wobbled ligand clashes with the binding pocket and is still close to the original crystal structure pose, which is colored in mint.

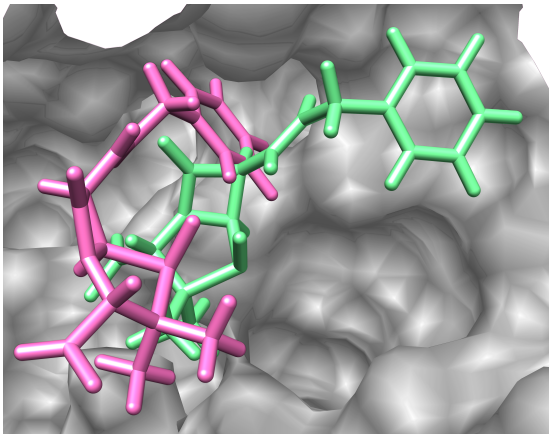


Figure 6.20.B: Optimized binding pocket of penicillin G acylase (*1gm8*). Optimizing the wobbled binding pocket depicted to the left drives the ligand (colored in fuchsia) further away from the crystal pose, which is colored in mint.

FIGURE 6.20. Ligand in binding pocket before and after optimization.

included the ligand and all amino acid side chains containing atoms closer than 4 $\AA$ to any ligand atom. YASARA's cutoff for non-bonded interactions was set to 8 $\AA$ and the limits of MOE's switching function were set to 8 $\AA$ and 10 $\AA$. All further details on the optimization procedures for the external tools are given in Sections 3.9.2 and 3.9.1.

This experiment measured RMSDs of minimized ligand poses to their respective crystal structure with the in-house tool for Trooper and YASARA. For results produced by MOE, this experiment employed its own RMSD calculation method. Significance levels were computed using two-tailed Mann Whitney U and Mood tests, which is introduced in Section 3.10. Figure 6.19 provides an overview on the obtained results.

Discussion. On average, Trooper drives ligands with an initial RMSD above 0.5 $\AA$ significantly towards their respective crystal structure ($p < 0.01$). All other poses are not significantly impaired by Trooper. In contrast, YASARA and MOE significantly deteriorate ligand poses with an initial RMSD of less than 0.5 $\AA$ to their respective crystal structure ($p < 0.001$). Overall, Trooper performs at least as well as the tested external tools at minimizing ligand poses in binding pockets with flexible amino acid side chains.

At the same time, results produced by Trooper exhibit a comparatively high level of variance. This research analyzed outliers with ligand RMSD values above 5 $\AA$ after the minimization procedure and discovered a common pattern. When optimizing a ligand pose that clashes with amino acid atoms (see Figure 6.20.a), Trooper's adaptive cooling scheme causes the system to heat up (see Figure 6.21). This allows the ligand to cross larger energy barriers and thus transcend parts of the binding pocket. This resolves the initial clash that then, in turn, results in

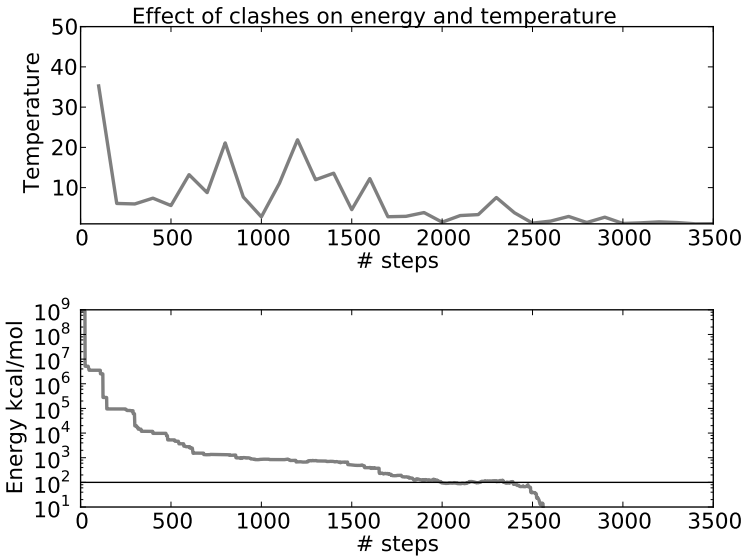


FIGURE 6.21. System temperature (upper depiction) and potential force field energy (lower depiction) while optimizing the wobbled binding pocket of penicillin G acylase (*1gm8*). An initial clash between ligand and binding pocket entails high potential energy, which causes the system to heat up.

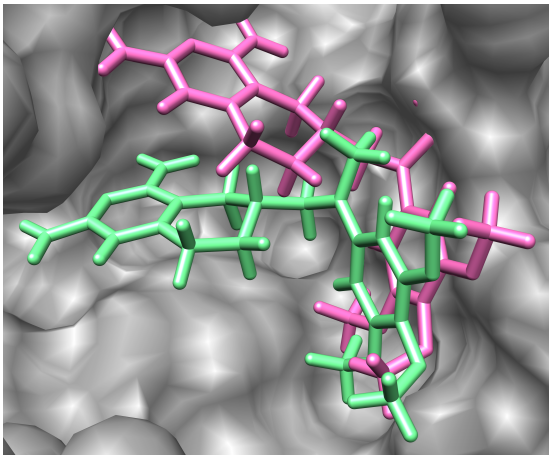


Figure 6.22.A: Wobbled binding pocket of dihydrofolate reductase (*1s3v*). A methyl group of the fuchsia colored wobbled ligand clashes with the binding pocket. The original crystal structure pose is colored in mint.

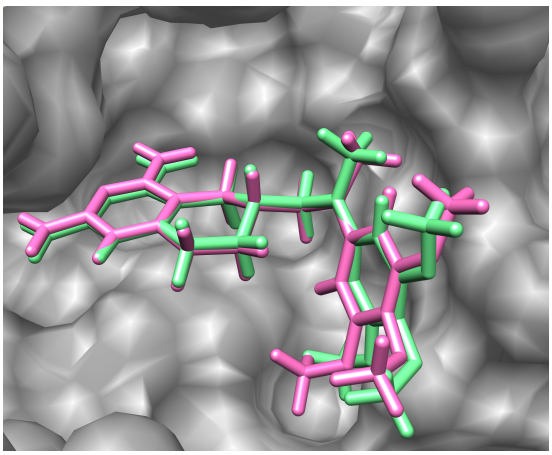


Figure 6.22.B: Optimized binding pocket of dihydrofolate reductase (*1s3v*). All clashes in the pre-optimized binding pocket depicted to the left are resolved. The minimized ligand (colored in fuchsia) almost matches the crystal pose, which is colored in mint.

FIGURE 6.22. Ligand in binding pocket before and after optimization.

a cooling of the system. In some cases, the path of the ligand pose to the crystal pose is now blocked by amino acids of the binding pocket (see Figure 6.20.b).

This pattern only occurs in rare cases. Most initial clashes do not lead to outliers, but are well resolved (see Figures 6.22.a and 6.22.b). However, to further improve Trooper, the data suggests an initial local minimization if a high force field energy is measured. This could circumvent the heating up of the system. Furthermore, it is likely that a ligand that is already close to its crystal pose would be further driven towards it. Additionally, it should be considered to allow the user to mark certain protein-ligand interactions as vital, which would add to the already available spatial constraints on atoms. Incorporating available knowledge into the optimization procedure could help produce results desired by the user.

6.7. Constrained Optimization of Molecular Conformations

All of the preceding experiments are centered around optimizing structures in binding pockets of protein-ligand complexes. However, the scope of application for Trooper is larger. As part of CONFECT, a conformation generator introduced in Section 3.8.5, Trooper minimizes small molecules while observing constraints on dihedral angle rotations. The following experiments test the efficacy of these downstream optimizations.

6.7.1. Parameterization. In the first experiment, Trooper was parameterized for downstream optimizations of small molecules. This experiment minimized the 4,997 conformations contained in the CONFECT test data set, which is introduced in Section 5.3.

The parameterization run tested all values of the set $\{5°, 10°, 15°, 20°\}$ for the maximum dihedral rotation parameter $[\Phi_{max}\,(DEG)]$ of Trooper. As only single molecules were minimized, dihedral rotations were the only allowed transformation operation. The initial temperature T was set to 0, the maximum number of steps (s_{max}) to 1,000, and the maximum number of failed steps to 100. The early abort-mechanism (see Section 4.5.4) was switched on. Its maximum number of consecutively rejected states ($ea_{rej_{max}}$) and the minimal absolute energy reduction ($|ea_{red_{min}}|$) parameters were set to 100 and 0.1 $kcal/mol$, respectively. Furthermore, the experiment constrained the dihedral rotations according to CONFECT's level II thresholds, as described in Section 4.7. After the minimization runs, the experiment computed the mean potential energy of the resulting conformations for each tested parameterization. To test whether mean energies differ significantly, the standard t-test (see Section 3.10) was applied. Table 6.14 provides an overview of the result.

Discussion. This experiment sets the value 15° for the Φ_{max} parameter. As visible in Table 6.23, this parameterization results in low energy conformations. Choosing smaller maximum dihedral angles clearly leads to unfavorable higher energies ($p < 0.001$). Setting Φ_{max} to 20° produces conformations with lower potential energies ($p < 0.05$). Still, larger dihedral rotations entail more constraint violations and therefore an enhanced runtime.

TABLE 6.14. Mean potential energy of conformations optimized with Trooper and different settings for the maximum dihedral rotation parameter (Φ_{max}). The last column specifies whether the respective measured mean value differs significantly from that for parameter setting $\Phi_{max} = 15°$. The levels of significance are 0.1% (***) and 5% (*) and were computed with the standard t-test.

$\Phi_{max}\,(DEG)$	mean $E\frac{kcal}{mol}$	significance
5°	23.01	***
10°	20.04	***
15°	13.10	-
20°	12.09	*

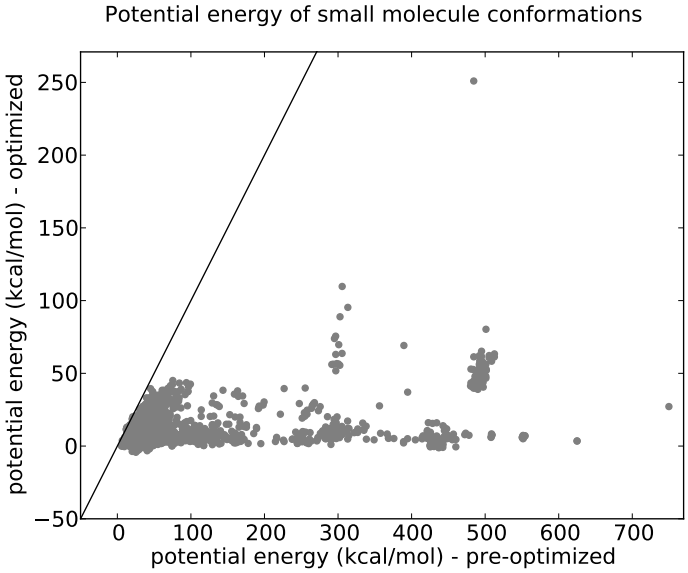


FIGURE 6.23. Potential force field energy of conformations before and after minimization with Trooper and parameter setting $\Phi_{max} = 15°$

6.7.2. Optimization quality and runtime. The parameterization selected, described in the preceding section, was evaluated. This research analyzed whether downstream optimizations enhanced the quality of conformations generated by CONFECT. A key performance figure is the number of constraint violations in the CONFECT test data (see Section 5.3) set before and after minimization runs. The experiment found that 485 out of an initial number of 4,997 conformations exhibit constraint violations after the optimization. The success rate is thus about 90%.

Furthermore, this research measured CONFECT's runtime with and without downstream optimizations. In the former case, generating conformations takes 20 s on test machine I. In the latter case, the procedure consumes 120 s. When utilizing eight CPU cores, optimizing a single conformation takes 0.02 s. In comparison, SZYBKI (see Section 2.1) consumes 0.5 s to minimize

a small molecule on an unknown machine and DG-AMMOS [**115**] consumes 0.61 s on two cores of an Intel Xeon CPU. Furthermore, ConfGen [**112**] generates conformations for one ligand in 4.2 s to 45.8 s on a single core of an Intel Core2 Q6600 processor. Therefore, Trooper is at least as fast as commercially available tools.

Overall, Trooper reduces the number of CONFECT's constraint violations by one order of magnitude. However, this enhanced quality comes at the cost of speed. Further reparameterizations could allow for reduction of small molecules with fewer optimization steps. Still, the runtime issue should only be tackled if speed proves to be vital for the usability of CONFECT.

CHAPTER 7

Conclusion and Outlook

This work presents a novel method that substantially accelerates force field-based optimizations of small molecules outside of, and especially within, protein binding sites. This chapter recapitulates the building blocks, features, and application scenarios of this method. Furthermore, it summarizes experiments performed for its parameterization, validation, and evaluation. Key findings of these experiments are highlighted. Finally, limits of application for the method are discussed and an outlook on topics worth investigating in future research is given.

7.1. Original Work: Trooper and GPUperTrooper

This work is centered around Trooper, a force field-based downstream optimization method. It is specifically designed to energetically minimize ligand molecules outside and inside protein binding sites. Therein, amino acid side chain conformations are also optimized. Trooper is a refinement method and intended to be applied within a pipeline of tools. Thus, initial structures for it must be generated previously by structure prediction methods. Examples of these are ligand conformation generators, molecular docking and lead optimization tools, as well as computational mutagenesis methods for protein design. Trooper refines the structures that these methods produce. In this process, it removes clashes and drives molecules closer to their experimentally determined conformations and poses.

Trooper targets systems that typically contain less than 1,500 atoms. The degrees of freedom it considers are rotations and translations of entire molecules. Furthermore, rotations about single dihedral bonds are performed. This operation optimizes amino acid side chain conformations. Consequently, small systems of which large parts stay rigid are minimized. Trooper exploits this by avoiding calculations of non-bonded forces between and within rigid regions.

The SuperTrAmber force field is Trooper's default objective function. Alternatively, the MMFF94s force field can be used. Two distinct operation modes minimize these objective functions. Both employ SA algorithms. The first one is a standard approach with a predefined exponential cooling scheme, which is applied for local optimizations that target ligand molecules and rigid protein binding sites. Consequently, it is called the local minimization mode. In contrast, the second mode dynamically adapts its parameterization to energies measured during the optimization process, which facilitates the removal of severe clashes between parts of molecules. Furthermore, energetic barriers can be crossed. At the same time, transformations that are energetically overly unfavorable are avoided. This second SA protocol is used whenever amino acid side chains are included in the optimization procedure. It is called the global optimization mode.

GPUperTrooper is an accelerated version of Trooper. Its purpose is to sustain the low computational cost of the aforementioned upstream methods that generate molecule structures. It is designed for optimizing ligand molecules in protein binding sites and takes amino acid side chain flexibility into account. It operates on modern SIMD graphics processors. Consequently, it builds upon data structures and algorithms that are designed for this processor and memory architecture. With these algorithms, GPUperTrooper performs SuperTrAmber force field energy computations and atom coordinate transformations in parallel. At the same time, GPUperTrooper seamlessly integrates with the same optimization framework that Trooper uses. This way, the SA procedure operates on the CPU of a host system, while time-demanding computations are executed on GPU. Consequently, GPUperTrooper uses the parameterizations of Trooper's local and global minimization modes. In the latter mode, GPUperTrooper additionally employs a refined algorithm that handles short-range intramolecular interactions more efficiently. GPUperTrooper is carefully tuned to avoid overheads. These commonly occur when accessing GPU memory and in the intercommunication between host and GPU devices.

7.2. Experimental Findings

7.2.1. Optimizing ligands in rigid protein binding pockets. The first series of experiments of this work spans Sections 6.1.1 to 6.1.5. These are centered around optimizing ligands in rigid protein binding pockets. This scenario is relevant when refining structures resulting from molecular docking runs. For parameterizing and validating Trooper, the docking tool TrixX (see Section 3.8.3) was used to produce ligand poses. Of these, 125 were selected to form a training set and another 67 were selected to form a test set. The RMSDs of docking poses to their respective crystal structure ranges from 1.5 $\mathring{A}$ to 3.0 $\mathring{A}$ in both sets. Trooper was parameterized on the training set and subsequently successfully validated on the test set. With the same set, Trooper's minimization results were compared to those of the commercially available tools MOE and YASARA, which Sections 3.9.2 and 3.9.1 introduced. In this evaluation experiment, Trooper reduced the average RMSD of docking poses from 2.35 $\mathring{A}$ to 1.95 $\mathring{A}$. It performed better than MOE, which reduced the average RMSD of the poses to 2.07 $\mathring{A}$, however, YASARA yielded the best results as it reduced the average RMSD to 1.67 $\mathring{A}$.

In Sections 6.1.6 and 6.1.7, Trooper was evaluated in cognate docking experiments. For the first one, TrixX docked the ligands of the protein targets of the Astex Diverse Set [**3**]. This way, TrixX produced excellent and correct poses (as defined in Section 3.2) for 22 and 61 protein targets, respectively. YASARA and Trooper optimized all docking poses resulting from this docking run. In this refinement stage, Trooper enhanced the number of protein targets with excellent and correct poses to 44 and 68, respectively. YASARA's optimization performed on par and yielded 50 and 66 protein targets with excellent and correct poses, respectively.

For a further cognate docking experiment, a pipeline formed of TrixX, Trooper, and the scoring function HYDE (see Section 3.8.1) was constructed. This experiment aimed at assessing three key questions: first, whether Trooper enhances the quality of TrixX's results in cognate

docking runs. Second, whether applying Trooper before HYDE helps this tool to identify correctly placed ligand poses. Third, whether the constructed pipeline performs on par with other state-of-the-art docking approaches. The experiment was conducted on a revised version of the Astex Diverse Set. This data set was compiled for the Docking and Scoring Symposium that took place during the 241st ACS National Meeting. The goal of this symposium was to evaluate the performance of current docking and scoring methods. The most relevant research groups in the field of docking participated in this symposium. The experiment determined that using Trooper to optimize results produced by TrixX drives docking poses closer to their respective experimentally determined structure: the number of binding pockets with excellent docking poses increases from 59 to 85. Furthermore, Trooper enhances HYDE's rescoring performance. The number of excellent ligand poses ranked among the top 32 increases by 24 when post-optimizations with Trooper precede the rescoring step. Finally, it was found that the whole pipeline performs on par with the state-of-the-art docking tools FlexX, SurFlex, Lead Finder, GOLD, DOCK, and MOE (these tools are introduced in Chapter 2) for producing correct docking poses.

GPUperTrooper was validated for ligand optimizations in rigid protein binding pockets in Sections 6.3.4 and 6.3.2. First, minimization results produced by Trooper and GPUperTrooper were compared in terms of quality. For this, GPUperTrooper optimized the previously introduced test set that comprises 67 docking poses. No significant aberrations between the average RMSDs of structures minimized with Trooper and GPUperTrooper were measured. In the second step, the runtimes of potential energy calculations were assessed for Trooper and GPUperTrooper. For this assessment, 15 binding pockets containing one random docking pose, respectively, were selected from the Astex Diverse Set. The total number of considered atoms in these binding pockets ranged from 871 to 1,821. Relevant SuperTrAmber potentials were computed 1,000,000 times with Trooper on CPU and GPUperTrooper on GPU. Measured speedup values (S_{GPU}, defined in Section 3.1) vary with the number of considered atom pairs in the binding pockets and range from 60 to 160.

Section 6.3.3 describes the evaluation of GPUperTrooper. It was performed on six protein binding sites. In each of these, 200 docked ligand poses were minimized with Trooper, GPUperTrooper, MOE, and YASARA. Depending on the selected parameterization, GPUperTrooper minimizes the ligands of the data set in $0.09\,s$ to $0.35\,s$ on average. Compared to the external methods, this implies a speedup of at least 100. Furthermore, computations with GPUperTrooper are cheaper by a factor of at least 23.4, according to the economic efficiency measure introduced in Section 3.1. Compared to Trooper, applying GPUperTrooper is at least four times cheaper and 38 times faster.

7.2.2. Optimizing amino acid side chains. Section 6.4 describes the validation and evaluation of amino acid side chain optimizations with Trooper's global minimization mode. The first experiment establishes a parameterization for this adaptive SA procedure. The following stage successfully validates Trooper. This validation was carried out on a data set comprising 748 protein-ligand complexes. Their binding sites contained randomly wobbled amino acid side

chain conformations and the crystal structure pose of the respective ligand. However, only the side chain conformations were optimized.

In the same experiment, Trooper's performance is compared to those of MOE and YASARA. Two further evaluation experiments probe the limits of application of side chain-only optimizations. The first experiment is conducted on 686 protein-ligand complexes. Naturally occurring amino acid side chain conformation variations were introduced in their binding pockets. These were taken from the Astex Non-Native Set [**198**]. At the same time, the backbone conformations remained unchanged. The second experiment attempts an optimization of the protein-ligand complexes of the Astex Non-Native Set. These contain backbone conformation variations. Trooper performed side chain-only optimizations, while YASARA was tested with and without additional degrees of freedom for backbone flexibility.

The experiments determined that on average, Trooper, MOE, and YASARA drive amino acid side chains closer to their respective experimentally determined conformation. All tested methods perform on par at this task. These observations hold if the average RMSD of the initial side chain conformations to their respective crystal structure is larger than $0.5\,\mathring{A}$. Furthermore, the backbone conformation must be correct. If this is not the case, neither Trooper nor YASARA minimizations are effective. Also, including backbone flexibility in YASARA's optimization procedure is shown to be useless. Thus, simply adding backbone flexibility appears to be an insufficient measure to cope with backbone variations in force field-based protein binding site optimizations.

Further experiments, documented in Section 6.5, assess the global minimization mode of GPUperTrooper. First, a validation run compares the performances of Trooper and GPUperTrooper. For this, the previously introduced data set comprised of 748 protein-ligand complexes was used. This experiment demonstrates that there are no significant differences between results produced by Trooper and GPUperTrooper. In the same data set, GPUperTrooper's runtime is compared to those of Trooper, MOE, and YASARA. It is determined that applying GPUperTrooper yields speedup values ranging from 96.5 to 202.1. Furthermore, computations with GPUperTrooper are 8.0 to 16.8 times cheaper than with the other tested methods. These figures were established with the economic efficiency measure. A final experiment shows that GPUperTrooper runs efficiently on the Tesla, Fermi, and Kepler generations of NVIDIA GPUs. Thus, GPUperTrooper is suitable for all NVIDIA GPUs released since 2007.

7.2.3. Optimizing entire binding pockets. The experiments in Section 6.6 test Trooper's global mode for optimizing ligands and amino acid side chains in protein binding sites. For this, the parameters for molecule translations and rotations are retrained. With this new parameterization, Trooper is compared to MOE and YASARA on a data set comprised of 330 protein-ligand complexes. Their binding sites contain randomly wobbled amino acid side chains and ligands. Trooper drives these wobbled ligands closer to their respective crystal structure poses. If the initial RMSD of a ligand is below $0.5\,\mathring{A}$, Trooper, on average, performs better than MOE and YASARA. For all other cases, all tested tools perform on par.

7.2.4. Constrained optimization of molecular conformations. Section 6.7 describes how Trooper's local mode is tested for optimizing small molecule conformations. These are generated by CONFECT, which uses a sophisticated set of dihedral constraints, introduced in Section 3.8.5. Trooper incorporates these constraints. In the first experimental step, a parameterization run determines a favorable dihedral angle rotation limit. With this, Trooper minimizes 4,997 conformations generated by CONFECT. Initially, all of these violate CONFECT's constraints. After the optimization run, 485 constraint violations are detected. Thus, Trooper's success rate is approximately 90%. Its average runtime for optimizing one conformation is $0.02\,s$. This figure is in the same regime as those reported for state-of-the-art tools such as SZYBKI (see Section 2.1) and DG-AMMOS [**115**].

7.3. Limits of Application

Trooper is a post-optimization method for refining small molecules inside and outside of protein binding sites. As such, it is not meant to generate entirely new molecule poses. Therefore, it should be applied in a pipeline of tools. In this, upstream methods must produce reasonable initial molecule poses and conformations. This task includes the generation of ring conformations, bond angles, and lengths. Also, a correct protonation state must be assigned in advance. Trooper was trained on structures with average deviations from their respective crystal structure of $3.0\,\mathring{A}$ and less. Its performance on structures with higher deviations is unknown. In protein binding sites, Trooper solely modifies amino acid side chain conformations. Evaluations show that deviating backbone conformations may have negative impacts on optimization results. Finally, Trooper is a stochastic energy minimization method. Therefore, it is not suitable for studying MD and minimization trajectories.

GPUperTrooper accelerates optimizations of protein-ligand binding pockets. It is possible to include water molecules in this procedure. However, all molecules are padded with dummy atoms so that they contain at least 32 atoms. Consequently, the handling of water molecules may impair efficiency. Therefore, these molecules should be added with care. Furthermore, GPUperTrooper performs best on systems comprised of 1,500 atoms or less and it is likely to be inefficient for larger systems. It has been neither designed nor benchmarked for small molecule optimizations.

7.4. Substantial Contributions

The results of this work show that Trooper and GPUperTrooper distinguish themselves from previous work by the following properties:

- GPUperTrooper is the only algorithm for graphics processors and, thus for SIMD architectures, which is specifically designed for quick force field-based minimizations of molecules in protein binding sites.
- GPUperTrooper saves time, energy, and money in large-scale application scenarios. With the economic efficiency this observation can be quantified. According to this

measure, optimizations with GPUperTrooper are at least eight times less energy consuming than those with the commercially available tools MOE and YASARA.

- GPUperTrooper makes interactive force field-based minimizations in desktop usage scenarios (e.g., lead optimizations) possible. This is due to the speedup of approximately two orders of magnitude that GPUperTrooper offers, compared to methods that operate on a single CPU core.
- Coupling GPUperTrooper with the fastest available docking tools like TrixX and PhDOCK [**7**] only negligibly affects their overall runtime. These tools produce docking poses in approximately $0.1\,s$. GPUperTrooper operates in the same runtime regime when optimizing ligands in binding pockets.
- There are only a few downstream optimization methods whose efficacy in conjunction with docking tools has been shown. This work contributes to existing studies by demonstrating that Trooper drives docking poses generated by TrixX closer to experimentally determined structures.
- It has been disputed that force field-based downstream optimizations are beneficial when applied prior to a rescoring stage that uses another objective functions [**4**, **5**]. This hypothesis is refuted in this work. The experiment in Section 6.1.7 clearly shows that Trooper improves the rescoring efficacy of HYDE.
- Trooper incorporates the dihedral angle constraints of CONFECT. In this fashion, Trooper supplements force field potentials with knowledge-based constraints derived from co-crystallized conformations by Schärfer [**6**]. With these, a highly specific and transparent association between dihedral angle constraints and torsion patterns is established.

7.5. Outlook

Experimental results of this work suggest several topics worth investigating in future research. Furthermore, there are application scenarios for downstream optimizations that are neither thoroughly analyzed in this nor in other studies.

- As discovered in Section 6.4.5, the minima of the SuperTrAmber force field often do not correspond to experimentally determined crystal structures of amino acid side chain conformations. However, energies of randomly wobbled structures are on average considerably higher than those of crystal structures. Therefore, parameters should be identified that prevent Trooper from driving conformations away from their crystal structure. With these, an alternative optimization mode could be established. A low initial energy could serve as an indication for using it.
- The experiment in Section 6.4.8 shows that neither Trooper nor YASARA drive amino acid side chains towards their respective crystal structure conformation in the presence of protein backbone variations. Including backbone flexibility in YASARA's optimization procedure yields the same results. Thus, prerequisites for successfully optimizing

structures with backbone variations should be searched for. This could start with an analysis of structures that YASARA successfully optimizes. Considering multiple predetermined backbone conformations would be another option. This way, potentially flexible segments of the protein backbone could be represented by an ensemble of predetermined sound conformations.

- Trooper is sensitive to falsely protonated molecules. Before applying it, a reasonable protonation state must thus be determined. A yet unpublished version of ProToss (an older version is introduced in Section 3.8.2) reliably assigns sound protonation states to protein binding site atoms and ligand molecules therein. Including this version of ProToss in the default preparatory steps of Trooper could prove to be beneficial.
- Trooper could benefit from including more pre-existing chemical knowledge, which could be mapped to further constraints. This would also permit a more fine-grained spatial and pattern-based definition of flexible molecular regions. Furthermore, vital interactions, e.g., certain hydrogen bonds, could be marked and thus protected from being broken during the optimization. These measures would enhance Trooper's usefulness in lead optimization scenarios. In virtual screenings, an upstream docking method could pass its pharmacophore-like constraints to Trooper.
- Including experimentally observed water molecules in binding site optimization could enhance the quality of results. This assumption should be validated. In a next step, GPUperTrooper could be adapted to efficiently handle water molecules. A possible approach would treat them all as a single molecule and its individual water molecules could be treated as components. Applying transformations to these would require minor adaptations to the `Transformation Executor` component of GPUperTrooper.
- The current Intel Haswell CPU generation features 16 SIMD registers, each 256 bits wide [**213**]. Future Intel Xeon processors will include more and wider SIMD registers [**214**]. The fundamental algorithms and data structures of GPUperTrooper are designed for the SIMD architecture. Therefore, it should be analyzed whether it is possible and sensible to implement them for x86 CPUs. For this, the usage of the OpenCL [**215**] standard should be considered.
- Perola et al. [**19**] demonstrated for two protein targets that force field-based post-optimizations improve enrichments in virtual screening. They claim that this observation only holds for tight binding pockets and provide one anecdotal experiment to prove this hypothesis. Further investigations on the interplay between downstream optimizations and rescoring functions in virtual screening should be conducted.

Bibliography

[1] E. Fischer, "Einfluss der Configuration auf die Wirkung der Enzyme," *Berichte der deutschen chemischen Gesellschaft*, vol. 27, no. 3, pp. 2985–2993, 1894. 1

[2] P. Csermely, R. Palotai, and R. Nussinov, "Induced fit, conformational selection and independent dynamic segments: an extended view of binding events," *Trends in Biochemical Sciences*, vol. 35, no. 10, pp. 539–546, 2010. 1

[3] M. Hartshorn, M. Verdonk, G. Chessari, S. Brewerton, W. Mooij, P. Mortenson, and C. Murray, "Diverse, high-quality test set for the validation of protein-ligand docking performance," *Journal of Medicinal Chemistry*, vol. 50, no. 4, pp. 726–741, 2007. 4, 79, 104, 113, 130

[4] J. C. Cole, C. W. Murray, J. W. M. Nissink, R. D. Taylor, and R. Taylor, "Comparing protein-ligand docking programs is difficult," *Proteins: Structure, Function, and Bioinformatics*, vol. 60, no. 3, pp. 325–332, 2005. 4, 14, 94, 134

[5] N. M. O'Boyle, J. W. Liebeschuetz, and J. C. Cole, "Testing assumptions and hypotheses for rescoring success in protein-ligand docking," *Journal of Chemical Information and Modeling*, vol. 49, pp. 1871–1878, Aug. 2009. 4, 14, 94, 134

[6] C. Schärfer, *Wissensbasierte Analyse von Konformationen in kleinen Molekülen*. Dissertation, Universität Hamburg, 2013. 4, 23, 24, 134

[7] D. Joseph-McCarthy, B. E. Thomas, M. Belmarsh, D. Moustakas, and J. C. Alvarez, "Pharmacophore-based molecular docking to account for ligand flexibility," *Proteins: Structure, Function, and Bioinformatics*, vol. 51, no. 2, pp. 172–188, 2003. 4, 17, 20, 103, 120, 134

[8] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, pp. 149–154, Jan. 1964. 6, 18, 23, 43

[9] S. S. Petrova and A. D. Solov'ev, "The origin of the method of steepest descent," *Historia Mathematica*, vol. 24, pp. 361–375, Nov. 1997. 6, 43

[10] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, pp. 308–313, Jan. 1965. 6, 15, 16, 17, 18

[11] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with CUDA," *Queue*, vol. 6, no. 2, pp. 40–53, 2008. 7

[12] NVIDIA corporation, "CUDA C programming guide," manual. 7, 8

[13] NVIDIA corporation, "NVIDIA's next generation CUDA compute architecture: Fermi," whitepaper, 2010. 7, 8

[14] NVIDIA corporation, "NVIDIA's next generation CUDA compute architecture: Kepler GK110," whitepaper, 2012. 7, 8

[15] LAVA Lab, University of Virginia, "CUDA support: CUDA Kernel Overhead." `http://www.cs.virginia.edu/~mwb7w/cuda_support/kernel_overhead.html`. [Online; accessed 15-October-2013]. 9

[16] T. Pencheva, D. Lagorce, I. Pajeva, B. Villoutreix, and M. Miteva, "Ammos: Automated molecular mechanics optimization tool for in silico screening," *BMC Bioinformatics*, vol. 9, no. 1, p. 438, 2008. 11

[17] "Ammp." `http://www.cs.gsu.edu/~cscrwh/ammp/ammp.html`. [Online, accessed 18-October-2013]. 11

[18] F. Mohamadi, N. G. J. Richards, W. C. Guida, R. Liskamp, M. Lipton, C. Caufield, G. Chang, T. Hendrickson, and W. C. Still, "Macromodel—an integrated software system for modeling organic and bioorganic molecules using molecular mechanics," *Journal of Computational Chemistry*, vol. 11, no. 4, pp. 440–467, 1990. 12

[19] E. Perola, W. P. Walters, and P. S. Charifson, "A detailed comparison of current docking and scoring methods on systems of pharmaceutical relevance," *Proteins: Structure, Function, and Bioinformatics*, vol. 56, pp. 235–249, Apr. 2004. 12, 19, 90, 135

[20] G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor, "Development and validation of a genetic algorithm for flexible docking," *Journal of Molecular Biology*, vol. 267, pp. 727–748, Apr. 1997. 12, 16, 20

[21] G. Jones, P. Willett, and R. C. Glen, "Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation," *Journal of Molecular Biology*, vol. 245, no. 1, pp. 43–53, 1995. 12, 16, 20

[22] 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," *Journal of Computational Chemistry*, vol. 15, no. 5, pp. 488–506, 1994. 12, 19, 20, 21, 22, 40

[23] M. Totrov and R. Abagyan, "Flexible protein-ligand docking by global energy optimization in internal coordinates," *Proteins*, vol. Suppl 1, pp. 215–220, 1997. 12, 19, 20, 21, 22, 40

[24] R. A. Friesner, J. L. Banks, R. B. Murphy, T. A. Halgren, J. J. Klicic, D. T. Mainz, M. P. Repasky, E. H. Knoll, M. Shelley, J. K. Perry, D. E. Shaw, P. Francis, and P. S. Shenkin, "Glide: A new approach for rapid, accurate docking and scoring. 1. Method and assessment of docking accuracy," *Journal of Medicinal Chemistry*, vol. 47, pp. 1739–1749, Mar. 2004. 12, 16, 20, 40

[25] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives, "Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids," *Journal of the American Chemical Society*, vol. 118, no. 45, pp. 11225–11236, 1996. 12, 16, 19, 21, 23, 51, 90

[26] G. A. Kaminski, R. A. Friesner, J. Tirado-Rives, and W. L. Jorgensen, "Evaluation and reparametrization of the OPLS-AA force field for proteins via comparison with accurate quantum chemical calculations on peptides," *The Journal of Physical Chemistry B*, vol. 105, no. 28, pp. 6474–6487, 2001. 12, 16, 19, 21, 51, 90

[27] M. L. Verdonk, J. C. Cole, M. J. Hartshorn, C. W. Murray, and R. D. Taylor, "Improved protein-ligand docking using GOLD," *Proteins: Structure, Function, and Bioinformatics*, vol. 52, no. 4, pp. 609–623, 2003. 12, 16

[28] Molecular Operating Environment (MOE), Versions 2010.10 and 2012.10, "Chemical Computing Group (CCG) Inc., 1010 Sherbooke St. West, Suite #910, Montreal, QC, Canada, H3A 2R7," 2012. 12, 22, 43

[29] S. Miura, S. Ferri, W. Tsugawa, S. Kim, and K. Sode, "Development of fructosyl amine oxidase specific to fructosyl valine by site-directed mutagenesis," *Protein Engineering Design and Selection*, vol. 21, pp. 233–239, Apr. 2008. 12, 22, 120

[30] A. Klenner, M. Weisel, F. Reisen, E. Proschak, and G. Schneider, "Automated docking of flexible molecules into receptor binding sites by ligand self-organization in situ," *Molecular Informatics*, vol. 29, no. 3, pp. 189–193, 2010. 12, 16, 20

[31] R. Das and D. Baker, "Macromolecular modeling with Rosetta," *Annual Review of Biochemistry*, vol. 77, no. 1, pp. 363–382, 2008. 13

[32] K. W. Kaufmann, G. H. Lemmon, S. L. DeLuca, J. H. Sheehan, and J. Meiler, "Practically useful: What the Rosetta protein modeling suite can do for you," *Biochemistry*, vol. 49, pp. 2987–2998, Apr. 2010. 13

[33] A. Leaver-Fay, M. Tyka, S. M. Lewis, O. F. Lange, J. Thompson, R. Jacak, K. Kaufman, P. D. Renfrew, C. A. Smith, and W. Sheffler, "ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules," *Methods Enzymol*, vol. 487, pp. 545–574, 2011. 13

[34] B. Kuhlman and D. Baker, "Native protein sequences are close to optimal for their structures," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 97, pp. 10383–10388, Sept. 2000. 13

[35] K. R. Khar, L. Goldschmidt, and J. Karanicolas, "Fast docking on graphics processing units via ray-casting," *PLoS ONE*, vol. 8, p. e70661, Aug. 2013. 13, 14, 20

[36] J. Meiler and D. Baker, "ROSETTALIGAND: protein-small molecule docking with full side-chain flexibility," *Proteins: Structure, Function, and Bioinformatics*, vol. 65, no. 3, pp. 538–548, 2006. 13, 17, 20

[37] I. W. Davis and D. Baker, "RosettaLigand docking with full ligand and receptor flexibility," *Journal of Molecular Biology*, vol. 385, pp. 381–392, Jan. 2009. 13, 17, 20

[38] OpenEyes Scientific Software, Santa Fe, New Mexico, USA, "SZYBKI." `http://www.eyesopen.com/szybki`. [Online; accessed 17-October-2013]. 13

[39] Daylight Chemical Information Systems, Inc., "Daylight Theory: SMARTS - A Language for Describing Molecular Patterns." `http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html`. [Online; accessed 13-November-2013]. 13

[40] Yet Another Scientific Artificial Reality Application (YASARA), Version 11.9.18. `http://www.yasara.org`. 13, 43

[41] S. Nabuurs, M. Wagener, and J. De Vlieg, "A flexible approach to induced fit docking," *Journal of Medicinal Chemistry*, vol. 50, no. 26, pp. 6507–6518, 2007. 13, 15, 20, 85, 96

[42] M. Mizutani, Y. Takamatsu, T. Ichinose, K. Nakamura, and A. Itai, "Effective handling of induced-fit motion in flexible docking," *Proteins*, vol. 63, no. 4, pp. 878–891, 2006. 14, 20

[43] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989. 14, 19

[44] A. C. Cabrera, J. Klett, H. G. Dos Santos, A. Perona, R. Gil-Redondo, S. M. Francis, E. M. Priego, F. Gago, and A. Morreale, "CRDOCK: an ultrafast multipurpose protein-ligand docking tool," *Journal of Chemical Information and Modeling*, vol. 52, pp. 2300–2309, Aug. 2012. 14, 20

[45] C. Pérez and A. R. Ortiz, "Evaluation of docking functions for protein-ligand docking," *Journal of Medicinal Chemistry*, vol. 44, pp. 3768–3785, Nov. 2001. 14

[46] H. Claußen, C. Buning, M. Rarey, and T. Lengauer, "FlexE: efficient molecular docking considering protein structure variations," *Journal of Molecular Biology*, vol. 308, pp. 377–395, Apr. 2001. 15

[47] E. Krieger, T. Darden, S. B. Nabuurs, A. Finkelstein, and G. Vriend, "Making optimal use of empirical energy functions: force-field parameterization in crystal space," *Proteins*, vol. 57, pp. 678–683, Dec. 2004. 15

[48] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe, "A fast flexible docking method using an incremental construction algorithm," *Journal of Molecular Biology*, vol. 261, no. 3, pp. 470–489, 1996. 15, 21, 22, 42

[49] D. K. Gehlhaar, G. M. Verkhivker, P. A. Rejto, C. J. Sherman, D. B. 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," *Chemistry & Biology*, vol. 2, pp. 317–324, May 1995. 15

[50] M. D. Miller, S. K. Kearsley, D. J. Underwood, and R. P. Sheridan, "FLOG: a system to select quasi-flexible ligands complementary to a receptor of known three-dimensional structure," *Journal of Computer-Aided Molecular Design*, vol. 8, pp. 153–174, Apr. 1994. 15, 20

[51] M. McGann, "FRED pose prediction and virtual screening accuracy," *Journal of Chemical Information and Modeling*, vol. 51, pp. 578–596, Mar. 2011. 15, 20

[52] R. A. Friesner, R. B. Murphy, M. P. Repasky, L. L. Frye, J. R. Greenwood, T. A. Halgren, P. C. Sanschagrin, and D. T. Mainz, "Extra precision Glide: docking and scoring incorporating a model of hydrophobic enclosure for protein-ligand complexes," *Journal of Medicinal Chemistry*, vol. 49, no. 21, pp. 6177–6196, 2006. 16, 20

[53] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic Acids Research*, vol. 28, pp. 235–242, Jan. 2000. 16, 81, 83

[54] M. D. Eldridge, C. W. Murray, T. R. Auton, G. V. Paolini, and R. P. Mee, "Empirical scoring functions: I. The development of a fast empirical scoring function to estimate the binding affinity of ligands in receptor complexes," *Journal of Computer-Aided Molecular Design*, vol. 11, pp. 425–445, Sept. 1997. 16, 79

[55] O. V. Stroganov, F. N. Novikov, V. S. Stroylov, V. Kulkov, and G. G. Chilov, "Lead finder: An approach to improve accuracy of protein-ligand docking, binding energy estimation, and virtual screening," *Journal of Chemical Information and Modeling*, vol. 48, pp. 2371–2385, Dec. 2008. 16, 20

[56] E. Neria, S. Fischer, and M. Karplus, "Simulation of activation free energies in molecular systems," *The Journal of Chemical Physics*, vol. 105, pp. 1902–1921, Aug. 1996. 16

[57] G. Wu, D. H. Robertson, C. L. Brooks, and M. Vieth, "Detailed analysis of grid-based molecular docking: A case study of CDOCKER—A CHARMm-based MD docking algorithm," *Journal of Computational Chemistry*, vol. 24, no. 13, pp. 1549–1562, 2003. 17, 21, 22, 97

[58] A. N. Jain, "Surflex: Fully automatic flexible molecular docking using a molecular similarity-based search engine," *Journal of Medicinal Chemistry*, vol. 46, pp. 499–511, Feb. 2003. 17, 20

[59] A. N. Jain, "Surflex-dock 2.1: Robust performance from ligand energetic modeling, ring flexibility, and knowledge-based search," *Journal of Computer-Aided Molecular Design*, vol. 21, pp. 281–306, May 2007. 17, 20

[60] A. N. Jain, "Effects of protein conformation in docking: improved pose prediction through protein pocket adaptation," *Journal of Computer-Aided Molecular Design*, vol. 23, pp. 355–374, June 2009. 17, 20

[61] S. L. Mayo, B. D. Olafson, and W. A. Goddard, "DREIDING: a generic force field for molecular simulations," *Journal of Physical Chemistry*, vol. 94, no. 26, pp. 8897–8909, 1990. 18

[62] I. D. Kuntz, J. M. Blaney, S. J. Oatley, R. Langridge, and T. E. Ferrin, "A geometric approach to macromolecule-ligand interactions," *Journal of Molecular Biology*, vol. 161, pp. 269–288, Oct. 1982. 18, 20, 40

[63] E. C. Meng, B. K. Shoichet, and I. D. Kuntz, "Automated docking with grid-based energy evaluation," *Journal of Computational Chemistry*, vol. 13, no. 4, pp. 505–524, 1992. 18, 20, 40

[64] T. J. A. Ewing and I. D. Kuntz, "Critical evaluation of search algorithms for automated molecular docking and database screening," *Journal of Computational Chemistry*, vol. 18, no. 9, pp. 1175–1189, 1997. 18, 20, 40

[65] T. J. A. Ewing, S. Makino, A. G. Skillman, and I. D. Kuntz, "DOCK 4.0: Search strategies for automated molecular docking of flexible molecule databases," *Journal of Computer-Aided Molecular Design*, vol. 15, pp. 411–428, May 2001. 18, 20, 40

[66] D. T. Moustakas, P. T. Lang, S. Pegg, E. Pettersen, I. D. Kuntz, N. Brooijmans, and R. C. Rizzo, "Development and validation of a modular, extensible docking program: DOCK 5," *Journal of Computer-Aided Molecular Design*, vol. 20, pp. 601–619, Oct. 2006. 18, 20, 40

[67] P. T. Lang, S. R. Brozell, S. Mukherjee, E. F. Pettersen, E. C. Meng, V. Thomas, R. C. Rizzo, D. A. Case, T. L. James, and I. D. Kuntz, "DOCK 6: Combining techniques to model RNA-small molecule complexes," *RNA*, vol. 15, pp. 1219–1230, June 2009. 18, 20, 40

[68] S. Weiner, P. Kollman, D. Case, U. Singh, C. Ghio, G. Alagona, S. Profeta, and P. Weiner, "A new force field for molecular mechanical simulation of nucleic acids and proteins," *Journal of the American Chemical Society*, vol. 106, no. 3, pp. 765–784, 1984. 18, 19, 35, 36, 38, 40

[69] S. J. Weiner, P. A. Kollman, D. T. Nguyen, and D. A. Case, "An all atom force field for simulations of proteins and nucleic acids," *Journal of Computational Chemistry*, vol. 7, no. 2, pp. 230–252, 1986. 18, 19, 35, 38

[70] D. A. Gschwend and I. D. Kuntz, "Orientational sampling and rigid-body minimization in molecular docking revisited: on-the-fly optimization and degeneracy removal," *Journal of Computer-Aided Molecular Design*, vol. 10, pp. 123–132, Apr. 1996. 18, 20, 40

[71] E. C. Meng, D. A. Gschwend, J. M. Blaney, and I. D. Kuntz, "Orientational sampling and rigid-body minimization in molecular docking," *Proteins*, vol. 17, pp. 266–278, Nov. 1993. 18, 20, 40

[72] H. Yang, B. Li, Y. Wang, Z. Luan, D. Qian, and T. Chu, "Accelerating DOCK6 Amber scoring with graphic processing unit," in *Algorithms and Architectures for Parallel Processing* (C.-H. Hsu, L. T. Yang, J. H. Park, and S.-S. Yeo, eds.), no. 6081 in Lecture Notes in Computer Science, pp. 404–415, Springer Berlin Heidelberg, Jan. 2010. 18

[73] C. R. Corbeil, P. Englebienne, and N. Moitessier, "Docking ligands into flexible and solvated macromolecules. 1. Development and validation of FITTED 1.0," *Journal of Chemical Information and Modeling*, vol. 47, pp. 435–449, Mar. 2007. 18, 20

[74] C. R. Corbeil, P. Englebienne, C. G. Yannopoulos, L. Chan, S. K. Das, D. Bilimoria, L. L'Heureux, and N. Moitessier, "Docking ligands into flexible and solvated macromolecules. 2. Development and application of FITTED 1.5 to the virtual screening of potential HCV polymerase inhibitors," *Journal of Chemical Information and Modeling*, vol. 48, pp. 902–909, Apr. 2008. 18, 20

[75] C. R. Corbeil and N. Moitessier, "Docking ligands into flexible and solvated macromolecules. 3. Impact of input ligand conformation, protein flexibility, and water molecules on the accuracy of docking programs," *Journal of Chemical Information and Modeling*, vol. 49, pp. 997–1009, Apr. 2009. 18, 20

[76] J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman, and D. A. Case, "Development and testing of a general Amber force field," *Journal of Computational Chemistry*, vol. 25, no. 9, pp. 1157–1174, 2004. 18, 19

[77] W. Sherman, T. Day, M. P. Jacobson, R. A. Friesner, and R. Farid, "Novel procedure for modeling Ligand/Receptor induced fit effects," *Journal of Medicinal Chemistry*, vol. 49, pp. 534–553, Jan. 2006. 18, 20

[78] M. P. Jacobson, R. A. Friesner, Z. Xiang, and B. Honig, "On the role of the crystal environment in determining protein side-chain conformations," *Journal of Molecular Biology*, vol. 320, pp. 597–608, July 2002. 18, 19

[79] M. P. Jacobson, G. A. Kaminski, R. A. Friesner, and C. S. Rapp, "Force field validation using protein side chain prediction," *The Journal of Physical Chemistry B*, vol. 106, pp. 11673–11680, Nov. 2002. 18

[80] M. P. Jacobson, D. L. Pincus, C. S. Rapp, T. J. Day, B. Honig, D. E. Shaw, and R. A. Friesner, "A hierarchical approach to all-atom protein loop prediction," *Proteins: Structure, Function, and Bioinformatics*, vol. 55, no. 2, pp. 351–367, 2004. 18

[81] F. A. Momany, R. F. McGuire, A. W. Burgess, and H. A. Scheraga, "Energy parameters in polypeptides. VII. geometric parameters, partial atomic charges, nonbonded interactions, hydrogen bond interactions, and intrinsic torsional potentials for the naturally occurring amino acids," *The Journal of Physical Chemistry*, vol. 79, pp. 2361–2381, Oct. 1975. 19, 22

[82] G. Nemethy, M. S. Pottle, and H. A. Scheraga, "Energy parameters in polypeptides. 9. updating of geometrical parameters, nonbonded interactions, and hydrogen bond interactions for the naturally occurring amino acids," *The Journal of Physical Chemistry*, vol. 87, pp. 1883–1887, May 1983. 19, 22

[83] G. Nemethy, K. D. Gibson, K. A. Palmer, C. N. Yoon, G. Paterlini, A. Zagari, S. Rumsey, and H. A. Scheraga, "Energy parameters in polypeptides. 10. Improved geometrical parameters and nonbonded interactions for use in the ECEPP/3 algorithm, with application to proline-containing peptides," *The Journal of Physical Chemistry*, vol. 96, pp. 6472–6484, July 1992. 19, 22

[84] M. Zacharias, "Rapid protein-ligand docking using soft modes from molecular dynamics simulations to account for protein deformability: Binding of FK506 to FKBP," *Proteins: Structure, Function, and Bioinformatics*, vol. 54, no. 4, pp. 759–767, 2004. 19, 20

[85] A. May and M. Zacharias, "Protein-Ligand docking accounting for receptor side chain and global flexibility in normal modes: Evaluation on kinase inhibitor cross docking," *Journal of Medicinal Chemistry*, vol. 51, pp. 3499–3506, June 2008. 19, 20

[86] U. Schulze-Gahmen, J. Brandsen, H. D. Jones, D. O. Morgan, L. Meijer, J. Vesely, and S. H. Kim, "Multiple modes of ligand recognition: crystal structures of cyclin-dependent protein kinase 2 in complex with ATP and two inhibitors, olomoucine and isopentenyladenine," *Proteins*, vol. 22, pp. 378–391, Aug. 1995. 19

[87] S. Leis and M. Zacharias, "Efficient inclusion of receptor flexibility in grid-based protein-ligand docking," *Journal of Computational Chemistry*, vol. 32, no. 16, pp. 3433–3439, 2011. 19

[88] D. S. Goodsell and A. J. Olson, "Automated docking of substrates to proteins by simulated annealing," *Proteins*, vol. 8, no. 3, pp. 195–202, 1990. 21, 40

[89] G. M. Morris, D. S. Goodsell, R. Huey, and A. J. Olson, "Distributed automated docking of flexible ligands to proteins: parallel applications of AutoDock 2.4," *Journal of Computer-Aided Molecular Design*, vol. 10, pp. 293–304, Aug. 1996. 21, 40

[90] 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," *Journal of Computational Chemistry*, vol. 19, no. 14, pp. 1639–1662, 1998. 21, 40

[91] G. M. Morris, R. Huey, W. Lindstrom, M. F. Sanner, R. K. Belew, D. S. Goodsell, and A. J. Olson, "AutoDock4 and AutoDockTools4: automated docking with selective receptor flexibility," *Journal of Computational Chemistry*, vol. 30, pp. 2785–2791, Dec. 2009. 21, 40

[92] R. Huey, G. M. Morris, A. J. Olson, and D. S. Goodsell, "A semiempirical free energy force field with charge-based desolvation," *Journal of Computational Chemistry*, vol. 28, no. 6, pp. 1145–1152, 2007. 21, 37

[93] O. Trott and A. J. Olson, "AutoDock vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading," *Journal of Computational Chemistry*, vol. 31, pp. 455–461, Jan. 2010. 21

[94] R. Wang, L. Lai, and S. Wang, "Further development and validation of empirical scoring functions for structure-based binding affinity prediction," *Journal of Computer-Aided Molecular Design*, vol. 16, no. 1, pp. 11–26, 2002. 21

[95] M. Vieth, J. D. Hirst, A. Kolinski, and C. L. Brooks, "Assessing energy functions for flexible docking," *Journal of Computational Chemistry*, vol. 19, no. 14, pp. 1612–1622, 1998. 21, 40

[96] M. Vieth, J. D. Hirst, B. N. Dominy, H. Daigler, and C. L. Brooks, "Assessing search strategies for flexible docking," *Journal of Computational Chemistry*, vol. 19, no. 14, pp. 1623–1631, 1998. 21, 40

[97] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus, "CHARMM: a program for macromolecular energy, minimization, and dynamics calculations," *Journal of Computational Chemistry*, vol. 4, no. 2, pp. 187–217, 1983. 21, 23, 51

[98] W. Wenzel and K. Hamacher, "Stochastic tunneling approach for global minimization of complex potential energy landscapes," *Physical Review Letters*, vol. 82, pp. 3003–3007, Apr. 1999. 21

[99] H. Merlitz, B. Burghardt, and W. Wenzel, "Application of the stochastic tunneling method to high throughput database screening," *Chemical Physics Letters*, vol. 370, pp. 68–73, Mar. 2003. 21

[100] H. Merlitz and W. Wenzel, "Comparison of stochastic optimization methods for receptor-ligand docking," *Chemical Physics Letters*, vol. 362, pp. 271–277, Aug. 2002. 21

[101] N. Schneider, S. Hindle, G. Lange, R. Klein, J. Albrecht, H. Briem, K. Beyer, H. Claussen, M. Gastreich, C. Lemmen, and M. Rarey, "Substantial improvements in large-scale redocking and screening using the novel HYDE scoring function," *Journal of Computer-Aided Molecular Design*, vol. 26, pp. 701–723, June 2012. 21, 42, 81, 95

[102] J. Schlosser and M. Rarey, "Beyond the virtual screening paradigm: Structure-based searching for new lead compounds," *Journal of Chemical Information and Modeling*, vol. 49, no. 4, pp. 800–809, 2009. 21, 22, 42, 103, 120

[103] Y. Zhao and M. F. Sanner, "FLIPDock: docking flexible ligands into flexible receptors," *Proteins: Structure, Function, and Bioinformatics*, vol. 68, no. 3, pp. 726–737, 2007. 21

[104] V. Schnecke and L. A. Kuhn, "Virtual screening with solvation and ligand-induced complementarity," *Perspectives in Drug Discovery and Design*, vol. 20, pp. 171–190, Dec. 2000. 22

[105] V. Schnecke and L. A. Kuhn, "Database screening for HIV protease ligands: the influence of binding-site conformation and representation on ligand selectivity," *Proceedings of the International Conference on Intelligent Systems for Molecular Biology ; ISMB. International Conference on Intelligent Systems for Molecular Biology*, pp. 242–251, 1999. 22

[106] I. Schellhammer and M. Rarey, "TrixX: structure-based molecule indexing for large-scale virtual screening in sublinear time," *Journal of Computer-Aided Molecular Design*, vol. 21, pp. 223–238, Feb. 2007. 22, 42

[107] A. J. Bordner and R. A. Abagyan, "Large-scale prediction of protein geometry and stability changes for arbitrary single point mutations," *Proteins: Structure, Function, and Bioinformatics*, vol. 57, pp. 400–413, June 2004. 22, 120

[108] R. Abagyan and M. Totrov, "Biased probability monte carlo conformational searches and electrostatic calculations for peptides and proteins," *Journal of Molecular Biology*, vol. 235, pp. 983–1002, Jan. 1994. 22

[109] Z. Zhang, L. Wang, Y. Gao, J. Zhang, M. Zhenirovskyy, and E. Alexov, "Predicting folding free energy changes upon single point mutations," *Bioinformatics*, vol. 28, pp. 664–671, Mar. 2012. 23

[110] TINKER-Software Tools for Molecular Design, "Washington University, St. Luis," 2012. 23

[111] Catalyst, "Accelrys Software Inc., San Diego, California, USA ." 23

[112] K. S. Watts, P. Dalal, R. B. Murphy, W. Sherman, R. A. Friesner, and J. C. Shelley, "ConfGen: a conformational search method for efficient generation of bioactive conformers," *Journal of Chemical Information and Modeling*, vol. 50, pp. 534–546, Apr. 2010. 23, 128

[113] W. L. Jorgensen and J. Tirado-Rives, "The OPLS [optimized potentials for liquid simulations] potential functions for proteins, energy minimizations for crystals of cyclic peptides and crambin," *Journal of the American Chemical Society*, vol. 110, no. 6, pp. 1657–1666, 1988. 23

[114] J. L. Banks, H. S. Beard, Y. Cao, A. E. Cho, W. Damm, R. Farid, A. K. Felts, T. A. Halgren, D. T. Mainz, J. R. Maple, R. Murphy, D. M. Philipp, M. P. Repasky, L. Y. Zhang, B. J. Berne, R. A. Friesner, E. Gallicchio, and R. M. Levy, "Integrated modeling program, applied chemical theory (IMPACT)," *Journal of Computational Chemistry*, vol. 26, no. 16, pp. 1752–1780, 2005. 23

[115] D. Lagorce, T. Pencheva, B. O. Villoutreix, and M. A. Miteva, "DG-AMMOS: a new tool to generate 3D conformation of small molecules using distance geometry and automated molecular mechanics optimization for in silico screening," *BMC Chemical Biology*, vol. 9, p. 6, Nov. 2009. 23, 128, 133

[116] I. T. Weber and R. W. Harrison, "Molecular mechanics calculations on rous sarcoma virus protease with peptide substrates.," *Protein Science : A Publication of the Protein Society*, vol. 6, pp. 2365–2374, Nov. 1997. 23

[117] P. C. D. Hawkins, A. G. Skillman, G. L. Warren, B. A. Ellingson, and M. T. Stahl, "Conformer generation with OMEGA: algorithm and validation using high quality structures from the protein databank and cambridge structural database," *Journal of Chemical Information and Modeling*, vol. 50, pp. 572–584, Apr. 2010. 24

[118] J. Stone, J. Phillips, P. Freddolino, D. Hardy, L. Trabuco, and K. Schulten, "Accelerating molecular modeling applications with graphics processors," *Journal of Computational Chemistry*, vol. 28, no. 16, pp. 2618–2640, 2007. 24, 99

[119] J. A. Anderson, C. D. Lorenz, and A. Travesset, "General purpose molecular dynamics simulations fully implemented on graphics processing units," *Journal of Computational Physics*, vol. 227, no. 10, pp. 5342–5359, 2008. 24

[120] J. van Meel, A. Arnold, D. Frenkel, S. Zwart, and R. Belleman, "Harvesting graphics power for MD simulations," *Molecular Simulations*, vol. 34, no. 3, pp. 259–266, 2008. 24, 99

[121] B. S. Weiguo Liu, "Accelerating molecular dynamics simulations using graphics processing units with CUDA," *Computer Physics Communications*, vol. 9, pp. 634–641, 2008. 24

[122] Y. Roh, J. Lee, S. Park, and J.-I. Kim, "A molecular docking system using CUDA," in *Proceedings of the 2009 International Conference on Hybrid Information Technology*, pp. 28–33, 2009. 24

[123] M. Friedrichs, P. Eastman, V. Vaidyanathan, M. Houston, S. Legrand, A. Beberg, D. Ensign, C. Bruns, and V. Pande, "Accelerating molecular dynamic simulation on graphics processing units," *Journal of Computational Chemistry*, vol. 30, no. 6, pp. 864–872, 2009. 24, 25, 26, 27, 100

[124] N. Schmid, M. Botschi, and W. Van Gunsteren, "A GPU solvent-solvent interaction calculation accelerator for biomolecular simulations using the GROMOS software," *Journal of Computational Chemistry*, vol. 31, no. 8, pp. 1636–1643, 2010. 24, 100

[125] P. Eastman and V. Pande, "Efficient nonbonded interactions for molecular dynamics on a graphics processing unit," *Journal of Computational Chemistry*, vol. 31, no. 6, pp. 1268–1272, 2010. 24, 25, 26, 27

[126] A. P. Ruymgaart, A. E. Cardenas, and R. Elber, "MOIL-opt: energy-conserving molecular dynamics on a GPU/CPU system," *Journal of Chemical Theory and Computation*, vol. 7, pp. 3072–3082, Oct. 2011. 24

[127] A. P. Ruymgaart and R. Elber, "Revisiting molecular dynamics on a CPU/GPU system: Water kernel and SHAKE parallelization," *Journal of Chemical Theory and Computation*, vol. 8, pp. 4624–4636, Nov. 2012. 24

[128] A. W. Götz, M. J. Williamson, D. Xu, D. Poole, S. Le Grand, and R. C. Walker, "Routine microsecond molecular dynamics simulations with AMBER on GPUs. 1. Generalized born," *Journal of Chemical Theory and Computation*, vol. 8, no. 5, pp. 1542–1555, 2012. 24, 26, 27, 100

[129] A. Anthopoulos, I. Grimstead, and A. Brancale, "GPU-accelerated molecular mechanics computations," *Journal of Computational Chemistry*, vol. 34, no. 26, pp. 2249–2260, 2013. 24, 26, 27, 102

[130] NVIDIA corporation, "NVIDIA GeForce GTX280 specifications." `http://www.nvidia.de/object/geforce_gtx_280_de.html`. [Online; accessed 12-November-2013]. 25

[131] W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. J. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman, "A second generation force field for the simulation of proteins, nucleic acids, and organic molecules," *Journal of the American Chemical Society*, vol. 117, pp. 5179–5197, 1995. 26, 34, 38

[132] J. L. Gustafson, "Reevaluating Amdahl's Law," *Communications of the ACM*, vol. 31, pp. 532–533, 1988. 29

[133] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, spring joint computer conference*, AFIPS '67 (Spring), (New York, NY, USA), pp. 483–485, ACM, 1967. 29

[134] V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, and P. Hammarlund, "Debunking the 100X GPU vs. CPU myth: An evaluation of throughput computing on CPU and GPU," in *ACM SIGARCH Computer Architecture News*, vol. 38, pp. 451–460, 2010. 29

[135] C. Gregg and K. Hazelwood, "Where is the data? Why you cannot debate CPU vs. GPU performance without the answer," in *Performance Analysis of Systems and Software (ISPASS), 2011 IEEE International Symposium on*, pp. 134–144, 2011. 30

[136] D. H. Woo and H.-H. Lee, "Extending Amdahl's Law for energy-efficient computing in the many-core era," *Computer*, vol. 41, no. 12, pp. 24–31, 2008. 30

[137] J. Brady, "A theory of productivity in the creative process," *IEEE Computer Graphics and Applications*, vol. 6, no. 5, pp. 25–34, 1986. 30

[138] A. J. Thadhani, "Interactive user productivity," *IBM Systems Journal*, vol. 20, pp. 407–423, Dec. 1981. 30

[139] R. B. Miller, "Response time in man-computer conversational transactions," in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, AFIPS '68 (Fall, part I), (New York, NY, USA), pp. 267–277, ACM, 1968. 30, 31

[140] N. Tolia, D. Andersen, and M. Satyanarayanan, "Quantifying interactive user experience on thin clients," *IEEE Computer*, vol. 39, pp. 46–52, March 2006. 30, 31

[141] S. K. Card, G. G. Robertson, and J. D. Mackinlay, "The information visualizer, an information workspace," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '91, (New York, NY, USA), pp. 181–186, ACM, 1991. 31

[142] The International Bureau of Weights and Measures, *The International System of Units (SI)*. US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 2001. 31

[143] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983. 31, 32

[144] V. Černý, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, pp. 41–51, Jan. 1985. 31

[145] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, p. 1087, 1953. 31, 32, 66

[146] B. Hajek, "Cooling schedules for optimal annealing," *Mathematics of Operations Research*, vol. 13, no. 2, pp. 311–329, 1988. 31

[147] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statistical Science*, vol. 8, no. 1, pp. 10–15, 1993. 31

[148] V. Granville, M. Krivanek, and J.-P. Rasson, "Simulated annealing: a proof of convergence," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 652–656, 1994. 31

[149] M. E. Johnson, *Simulated Annealing (SA) & Optimization: Modern Algorithms with VLSI, Optical Design, & Missle Defense Applications*. Syracuse, N.Y.: American Sciences Press, 1988. 31

[150] P. J. v. Laarhoven and E. H. Aarts, *Simulated Annealing: Theory and Applications*. Springer, June 1987. 31

[151] S. Lakshmanan and H. Derin, "Simultaneous parameter estimation and segmentation of gibbs random fields using simulated annealing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 8, pp. 799–813, 1989. 31

[152] M. Locatelli, "Convergence properties of simulated annealing for continuous global optimization," *Journal of Applied Probability*, vol. 33, pp. 1127–1140, 1996. 31

[153] M. Locatelli, "Simulated annealing algorithms for continuous global optimization: convergence conditions," *Journal of Optimization Theory and Applications*, vol. 104, no. 1, pp. 121–133, 2000. 31

[154] U. Faigle and W. Kern, "Note on the convergence of simulated annealing algorithms," *SIAM Journal on Control and Optimization*, vol. 29, no. 1, pp. 153–159, 1991. 31

[155] A. Johnson and S. Jacobson, "On the convergence of generalized hill climbing algorithms," *Discrete Applied Mathematics*, vol. 119, no. 1-2, pp. 37–57, 2002. 31

[156] M. Lundy and A. Mees, "Convergence of an annealing algorithm," *Mathematical Programming*, vol. 34, no. 1, pp. 111–124, 1986. 31

[157] D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli, "Convergence and finite-time behavior of simulated annealing," *Advances in Applied Probability*, vol. 18, pp. 747–771, Sept. 1986. 31

[158] Y. Rossier, M. Troyon, and T. M. Liebling, "Probabilistic exchange algorithms and euclidean traveling salesman problems," *Operations-Research-Spektrum*, vol. 8, no. 3, pp. 151–164, 1986.

[159] Y. Nourani and B. Andresen, "A comparison of simulated annealing cooling strategies," *Journal of Physics A: Mathematical and General*, vol. 31, no. 41, p. 8373, 1998. 32

[160] M. Harris, "Optimizing parallel reductions in CUDA." `http://developer.download.nvidia.com/compute/cuda/1.1-Beta/x86_website/projects/reduction/doc/reduction.pdf`. [online; accessed 19-november-2013]. 33

[161] W. Kahan, "Pracniques: further remarks on reducing truncation errors," *Communications of the ACM*, vol. 8, p. 40, Jan. 1965. 33

[162] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C (2nd ed.): The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 1992. 34

[163] R. Klein, D. Hauck, and S. Heinz, "unpublished results." 34, 35

[164] T. A. Halgren, "MMFF VI. MMFF94s option for energy minimization studies," *Journal of Computational Chemistry*, vol. 20, no. 7, pp. 720–729, 1999. 34, 37

[165] Y. Duan, C. Wu, S. Chowdhury, M. Lee, G. Xiong, W. Zhang, R. Yang, P. Cieplak, R. Luo, T. Lee, J. Caldwell, J. Wang, and P. Kollman, "A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations," *Journal of Computational Chemistry*, vol. 24, no. 16, pp. 1999–2012, 2003. 34, 38

[166] M. Clark, R. Cramer, and N. Van Opdenbosch, "Validation of the general purpose TRIPOS 5.2 force field," *Journal of Computational Chemistry*, vol. 10, no. 8, pp. 982–1012, 1989. 35, 36, 38

[167] PDB format. `http://www.wwpdb.org/documentation/format32/v3.2.html`. [Online; accessed 27-March-2012]. 35, 76

[168] TRIPOS International, "MOL2 format." `http://tripos.com/index.php`. [Online; accessed 27-March-2012]. 35, 76

[169] M. Beachy, D. Chasman, R. Murphy, T. A. Halgren, and R. Friesner, "Accurate ab initio quantum chemical determination of the relative energetics of peptide conformations and assessment of empirical force fields," *Journal of the American Chemical Society*, vol. 119, no. 25, pp. 5908–5920, 1997. 35

[170] G. M. Keseru and I. Kolossvary, *Molecular Mechanics and Conformational Analysis in Drug Design*. Blackwell, London, UK, 1999. 35

[171] J. Gasteiger and M. Marsili, "Iterative partial equalization of orbital electronegativity—a rapid access to atomic charges," *Tetrahedron*, vol. 36, no. 22, pp. 3219–3228, 1980. 36, 39

[172] T. A. Halgren, "Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94," *Journal of Computational Chemistry*, vol. 17, no. 5-6, pp. 490–519, 1996. 37, 39

[173] T. A. Halgren, "Merck molecular force field. II. MMFF94 van der waals and electrostatic parameters for intermolecular interactions," *Journal of Computational Chemistry*, vol. 17, no. 5-6, pp. 520–552, 1996. 37

[174] T. A. Halgren, "Merck molecular force field. III. Molecular geometries and vibrational frequencies for MMFF94," *Journal of Computational Chemistry*, vol. 17, no. 5-6, pp. 553–586, 1996. 37

[175] T. A. Halgren, "Merck molecular force field. V. Extension of MMFF94 using experimental data, additional computational data, and empirical rules," *Journal of Computational Chemistry*, vol. 17, no. 5-6, pp. 616–641, 1996. 37

[176] T. A. Halgren and R. Nachbar, "Merck molecular force field. IV. Conformational energies and geometries for MMFF94," *Journal of Computational Chemistry*, vol. 17, no. 5-6, pp. 587–615, 1996. 37

[177] S. Urbaczek, A. Kolodzik, J. Fischer, T. Lippert, S. Heuser, I. Groth, T. Schulz-Gasch, and M. Rarey, "Naomi: On the almost trivial task of reading molecules from different file formats," *Journal of Chemical Information and Modeling*, vol. 51, no. 12, pp. 3199–3207, 2011. 39

[178] I. Reulecke, G. Lange, J. Albrecht, R. Klein, and M. Rarey, "Towards an integrated description of hydrogen bonding and dehydration: Decreasing false positives in virtual screening with the HYDE scoring function," *ChemMedChem*, vol. 3, no. 6, pp. 885–897, 2008. 42

[179] N. Schneider, G. Lange, S. Hindle, R. Klein, and M. Rarey, "A consistent description of HYdrogen bond and DEhydration energies in protein-ligand complexes: methods behind the HYDE scoring function," *Journal of Computer-Aided Molecular Design*, vol. 27, pp. 15–29, Dec. 2012. 42

[180] T. Lippert and M. Rarey, "Fast automated placement of polar hydrogen atoms in protein-ligand complexes," *Journal of Cheminformatics*, vol. 1, no. 1, p. 13, 2009. 42

[181] A. Griewel, O. Kayser, J. Schlosser, and M. Rarey, "Conformational sampling for large-scale virtual screening: Accuracy versus ensemble size," *Journal of Chemical Information and Modeling*, vol. 49, pp. 2303–2311, Oct. 2009. 42

[182] C. Schärfer, T. Schulz-Gasch, J. Hert, L. Heinzerling, B. Schulz, T. Inhester, M. Stahl, and M. Rarey, "CONFECT: conformations from an expert collection of torsion patterns," *ChemMedChem*, vol. 8, no. 10, pp. 1690–1700, 2013. 42

[183] C. Schärfer, T. Schulz-Gasch, H.-C. Ehrlich, W. Guba, M. Rarey, and M. Stahl, "Torsion angle preferences in druglike chemical space: A comprehensive guide," *Journal of Medicinal Chemistry*, vol. 56, pp. 2016–2028, Mar. 2013. 42

[184] F. H. Allen, "The cambridge structural database: a quarter of a million crystal structures and rising," *Acta Crystallographica Section B Structural Science*, vol. 58, pp. 380–388, May 2002. 42, 83

[185] H. Levene, "Robust tests for equality of variances," in *Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling* (I. Olkin, H. Hotelling, *et al.*, eds.), pp. 278–292, Stanford: Stanford University Press, 1960. 44

[186] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python," 2001–. 44, 45

[187] T. E. Oliphant, "Python for scientific computing," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10–20, 2007. 44, 45

[188] R. A. Fisher, "Applications of "student's" distribution," *Metron*, vol. 5, pp. 90–104, 1925. 44

[189] B. L. Welch, "The generalization of "student's" problem when several different population variances are involved," *Biometrika*, vol. 34, no. 1-2, pp. 28–35, 1947. 45

[190] F. E. Satterthwaite and B. L. Welch, "An approximate distribution of estimates of variance components," *Biometrics Bulletin*, vol. 2, no. 6, pp. 110–114, 1946. 45

[191] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 1947. 45

[192] A. McFarlane Mood and F. A. Graybill, *Introduction to the Theory of Statistics*. New York, New York, USA: McGraw-Hill Book Co., 1950. 45

[193] M. J. Hwang, T. P. Stockfisch, and A. T. Hagler, "Derivation of class II force fields. 2. derivation and characterization of a class II force field, CFF93, for the alkyl functional group and alkane molecules," *Journal of the American Chemical Society*, vol. 116, pp. 2515–2525, Mar. 1994. 51

[194] E. Gamma, R. Johnson, J. Vlissides, and R. Helm, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995. 52, 66

[195] M. Gent, "Objektorientierte Implementierung von klassischen Kraftfeldern am Beispiel von MMFF94s," master's thesis, University of Hamburg, 05 2011. 56, 90

[196] T. A. Halgren, "MMFF94 validation suite." `http://www.ccl.net/cca/data/MMFF94/ver.98.05.22/index.html`. [Online; accessed 16-September-2013]. 56

[197] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C (3rd ed.): The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 2007. 62

[198] M. L. Verdonk, P. N. Mortenson, R. J. Hall, M. J. Hartshorn, and C. W. Murray, "Protein-ligand docking against non-native protein conformers," *Journal of Chemical Information and Modeling*, vol. 48, pp. 2214–2225, Nov. 2008. 81, 113, 115, 132

[199] Intel corporation, "Intel Xeon E5420 specifications." `http://ark.intel.com/products/33927`. [Online; accessed 14-October-2013]. 84

[200] Intel corporation, "Intel Xeon E5-2609 specifications." `http://ark.intel.com/products/64588`. [Online; accessed 14-October-2013]. 84

[201] Intel corporation, "Intel Xeon E5-2680 specifications." `http://ark.intel.com/products/64583`. [Online; accessed 14-October-2013]. 84

[202] Intel corporation, "Intel Xeon E5630 specifications." `http://ark.intel.com/products/47924`. [Online; accessed 14-October-2013]. 84

[203] gpuzoo, "NVIDIA Tesla C1060 specifications." `http://www.nvidia.de/object/geforce_gtx_280_de.html`. [online; accessed 14-november-2013]. 84

[204] NVIDIA corporation, "NVIDIA GeForce GTX680 specifications." http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-680/specifications. [Online; accessed 14-November-2013]. 84

[205] Fujitsu corporation, "NVIDIA Tesla K20 specifications." http://globalsp.ts.fujitsu.com/dmsp/Publications/public/ps-py-nvidia-k20-k20x-en.pdf. [Online; accessed 14-November-2013]. 84

[206] NVIDIA corporation, "NVIDIA Quadro 4000 specifications." http://www.nvidia.com/object/product-quadro-4000-us.html. [Online; accessed 14-November-2013]. 84

[207] M. Neves, M. Totrov, and R. Abagyan, "Docking and scoring with ICM: the benchmarking results and strategies for improvement," *Journal of Computer-Aided Molecular Design*, vol. 26, no. 6, pp. 675–686, 2012. 95

[208] R. Spitzer and A. Jain, "Surflex-Dock: Docking benchmarks and real-world application," *Journal of Computer-Aided Molecular Design*, vol. 26, no. 6, pp. 687–699, 2012. 95

[209] F. Novikov, V. Stroylov, A. Zeifman, O. Stroganov, V. Kulkov, and G. Chilov, "Lead finder docking and virtual screening evaluation with Astex and DUD test sets," *Journal of Computer-Aided Molecular Design*, vol. 26, no. 6, pp. 725–735, 2012. 95

[210] J. W. Liebeschuetz, J. C. Cole, and O. Korb, "Pose prediction and virtual screening performance of GOLD scoring functions in a standardized test," *Journal of Computer-Aided Molecular Design*, vol. 26, no. 6, pp. 737–748, 2012. 95

[211] S. Brozell, S. Mukherjee, T. Balius, D. Roe, D. Case, and R. Rizzo, "Evaluation of DOCK 6 as a pose generation and database enrichment tool," *Journal of Computer-Aided Molecular Design*, vol. 26, no. 6, pp. 749–773, 2012. 95

[212] C. Corbeil, C. Williams, and P. Labute, "Variability in docking success rates due to dataset preparation," *Journal of Computer-Aided Molecular Design*, vol. 26, no. 6, pp. 775–786, 2012. 95

[213] Intel corporation, "Details of Intel Advanced Vector Extensions Intrinsics." http://software.intel.com/sites/products/documentation/doclib/iss/2013/compiler/cpp-lin/GUID-A9C3B12F-7A9A-4C8D-A6CD-9974ABC570E9.htm. [Online; accessed 09-December-2013]. 135

[214] Intel corporation, "AVX-512 instructions." http://software.intel.com/en-us/blogs/2013/avx-512-instructions. [Online; accessed 09-December-2013]. 135

[215] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: a parallel programming standard for heterogeneous computing systems," *IEEE Design and Test of Computers*, vol. 12, pp. 66–73, May 2010. 135

APPENDIX A

Appendix

A.1. Software Packages

The methods described in this work were implemented in C++ and are grouped into three main packages. These are, as Figure A.1 shows, the `Forcefield`, `ForcefieldCUDA`, and `Optimization` packages.

The `Forcefield` package comprises five subpackages:

`Core` implements the force field framework that Section 4.2 introduces. Figure 4.3 captures the architecture of the force field implementation. Furthermore, the `Core` package contains the `Scorer` class that represents the component with the same name introduced in Section 4.2.4.

`Molinfo` implements the molecule representation and the component tree introduced in Section 4.1. Figure 4.1 reflects the class design of this subpackage.

`ChemModels` contains partial charge and atom type assigning classes. As Figure A.2 shows, assigners for AMBER types, SYBYL types, and MMFF94 types are available. The corresponding atom type models are introduced in Section 3.5. Implemented partial

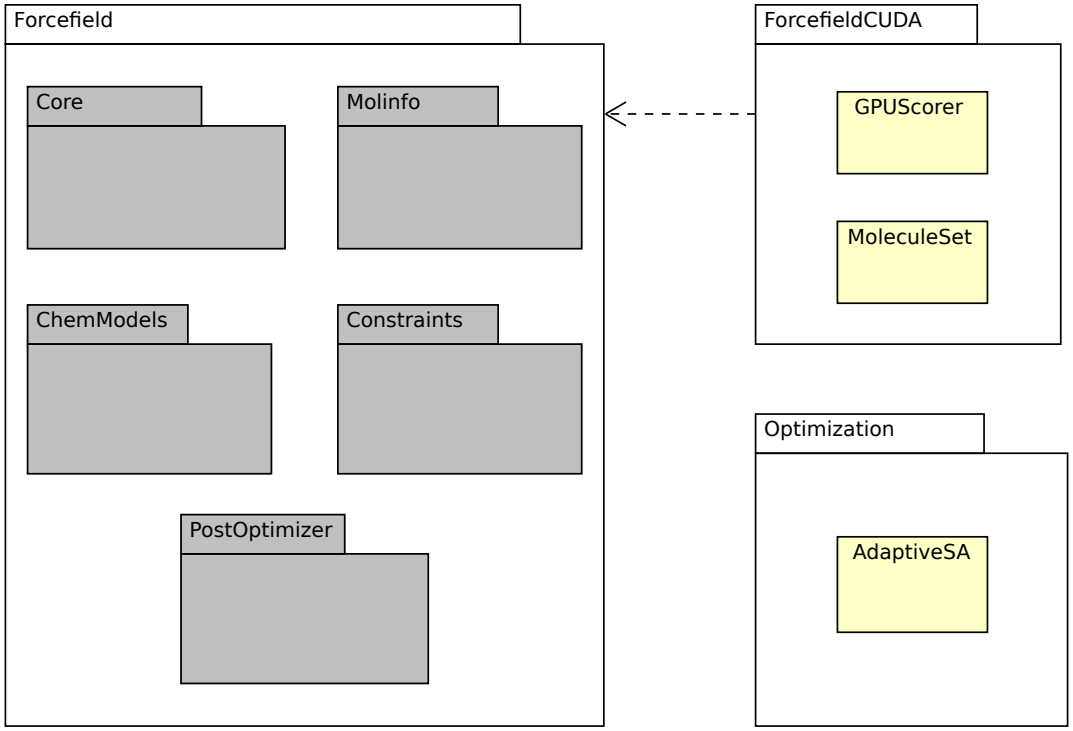


FIGURE A.1. Main packages of this work. Subpackages are shown in gray and the most relevant classes in yellow.

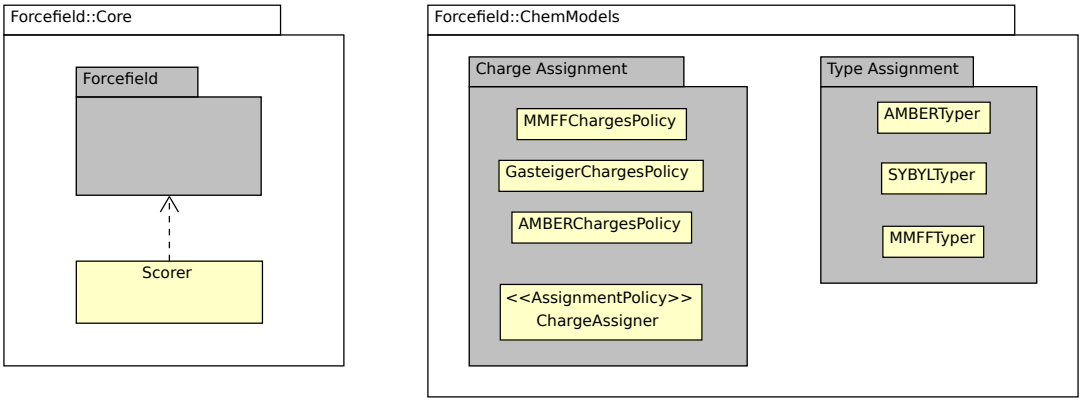


FIGURE A.2. Detail on subpackages of the Forcefield package. Nested subpackages are shown in gray and the most relevant classes in yellow.

charge models, namely Gasteiger-Marsili, MMFF94, and AMBER, are described in Section 3.6.

`PostOptimizer` comprises the CPU-based components of the optimization algorithm that Section 4.5.5 describes. Next to classes that compute molecular transformations, this package implements all CPU-based `StateManager` components shown in Figure 4.13.

`Constraints` contains data structures and functions that permit to constrain an optimization, as described in Section 4.7 .

The `ForcefieldCUDA` package contains the GPU portation of the SuperTrAmber force field. Its main classes are `MoleculeSet` and `GPUScorer`. The former implements the data structures described in Sections 4.3.1 and 4.4, while the latter offers the scoring functionality described in Sections 4.3.3 and 4.6.7.

Finally, the `Optimization` package implements the generic SA framework introduced in Sections 4.5 and 4.6, including the policies that guide the optimization procedure.

A.2. Tool Manual

A.2.1. Trooper and GPUperTrooper. The executable `postOptimizer` provides access to Trooper and GPUperTrooper, the two main methods of this work introduced in Section 4.8. This executable is called via the command line using the following parameters:

`--help` display help
`--gpu` GPUperTrooper mode, SuperTrAmber force field, CUDA required
`--complex-flexible` flexible protein-ligand complex mode (default)
`--complex-ligRigid` flexible protein-rigid ligand complex mode
`--complex-protRigid` rigid protein-flexible ligand complex mode
`--local` local parameterization (default)
`--global` global parameterization
`--tramber` SuperTrAMBER force field (default)
`--mmff94s` MMFF94s force field—no GPU support
`--strongHBonds` switch on SuperTrAmber's strong H-Bond mode
`--fast` switch on early-abort mechanism
`-l [ --ligand ] arg` path to input ligand molecule file
`-p [ --protein ] arg` path to input protein molecule file
`--proteinDB arg` path to input protein database
`--ligand_out arg` path to ligand output file
`--protein_out arg` path to protein output file
`--proteinDB_out arg` path to output protein database
`--log_out arg` path to log output file
`--steps arg` number of optimization steps
`--temperature arg` set initial temperature
`--max-dihedral arg` set maximal dihedral rotation
`--max-rotation arg` set maximal rotation
`--max-translation arg` set maximal translation
`--rescaling-factor arg` set transformation rescaling factor
`--flexible-side-chains arg` list of flexible amino acids. Format: TYR981 ARG771
`--constrainLigand arg` radial constraint ($\AA$) for ligand movements (CPU only)

A.2.2. ActiveSiteWobbler. The executable `ActiveSiteWobbler` can manipulate binding pockets of proteins by applying random transformations to molecules in binding sites, as described in Section 4.9. For this purpose, this program performs global molecule rotations and translations, as well as dihedral rotations. For amino acid side chains, only the latter transformation type is available. Thus, the `ActiveSiteWobbler` operates on the same degrees of freedom as Trooper and GPUperTrooper. Additionally, the `ActiveSiteWobbler` permits transferring side chain conformations between binding pockets. The executable is called via the command line using the following parameters:

`--help` display help

`-p [ --protein ] arg` protein structure to manipulate (pdb format)

`-l [ --refLig ] arg` reference ligand defining the active site

`-d [ --radius ] arg` radius (Å) around reference ligand defining the active site

`-n [ --nofSteps ] arg` number of random transformations to perform

`--includeLigand` switch for including ligand in wobbling operations

`-o [ --outProt ] arg` path for saving transformed protein structure (pdb format)

`--outDB arg` path to database for saving transformed complex (internal database format)

`--outLigand arg` path for transformed ligand structure

`--transferConformation arg` source complex for transferring conformations of binding site amino acid side chains (pdb format)

A.3. Acknowledgements

Ich danke Prof. Dr. Matthias Rarey für Leitung und Anleitung und für richtige und wichtige Fragen zur rechten Zeit. Vor allen Dingen möchte ich mich für die letzten fünf Jahre bedanken, während derer ich in der fröhlichen und inspirierenden Athmosphäre der AMD-Gruppe arbeiten und dazu lernen durfte und außerdem die weite Welt der Wissenschaft bei Konferenzen und Tagungen erkunden konnte. Ich werde diese Zeit sicher in guter Erinnerung behalten.

Ich danke Prof. Dr. Stephan Olbrich für das Vermitteln seiner Sichtweise auf das Hochleistungsrechnen und das Begutachten dieser Arbeit.

Dr. Robert Klein danke ich für seine umfassende Beratung rund um Kraftfelder, für die Zusammenarbeit bei unserer gemeinsamen Publikation und die Überlassung des FORTRAN-Codes des SuperTrAmber-Kraftfeldes.

Weiter geht ein Dank nach Santa Clara: I gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K20 GPU used for this research.

Mein besonderer Dank gilt meinen Kollegen aus der AMD-Gruppe. Freude an der täglichen Arbeit, das weiß ich nach dem Niederschreiben der Dissertation sicher, ist ohne nette Mitarbeiter undenkbar. Im speziellen danke ich Dr. Martin Kowalski, der trotz seiner Dreifachbelastung durch Frau, Kinder und Haus mir in den letzten Monaten selbstlos seinen Arbeitsplatz überließ. Ich danke Matthias Hilbig für spannende Diskussionen von Fragestellungen rund um Programmierung und Algorithmen und für gemeinsame Abende, ob diese nun im Kino, bei Konzerten oder beim Falafel-Imbiss verbracht wurden. Therese Inhester danke ich für die gute Laune, Spaß bei der Begutachtung esoterischer Publikationen vom Biomarkt und gewonnene Tischfußballpartien, Dr. Tim Harder für das Korrekturlesen dieser Arbeit und schweißtreibende Badminton-Matches. Ebenso für dies beides und zudem für eine mutige Anreise über die Elbe nach Wilhelmsburg danke ich Florian Lauck. Melanie Geringhoff danke ich für aufmunternde Worte und ihre Gelassenheit, Stefan Bietz für Korrekturhilfen und Einsichten rund ums Protein. Auch bei Dr. Nadine Schneider und Agnes Meyder bedanke ich mich für Korrekturen und Vorschläge zu dieser Arbeit. Dr. Christin Schärfer danke ich für die gelungene Zusammenarbeit für den CONFECT-Optimierer und noch einmal für das Aufhängen meines Posters bei der GCC 2012. Angela Henzler möchte ich für die gemeinsame, spannende Konferenzreise nach Philadelphia 2012 und die Gespräche unter Dissertationsschreibenden danken. Jörn Adomeit und Christian Rhein möchte ich für ihre Hilfsbereitschaft beim Lösen technischer Probleme aller Art, insbesondere bei der Installation von Grafikprozessoren danken. Allen nicht genannten ehemaligen und aktuellen Mitarbeitern danke ich für die schöne Zeit bei der Arbeit, beim Essen und beim Pausieren in der Kaffeeküche.

A.4. List of Publications

Schärfer, C; Schulz-Gasch, T; Hert, J; Heinzerling, L; Schulz, B; Inhester, T; Stahl, M; Rarey, M
CONFECT: Conformations from an Expert Collection of Torsion Patterns.
ChemMedChem, 2013, 8(10):1690–1700

Heinzerling, L; Klein, R; Rarey, M
Fast force field-based optimization of protein-ligand complexes with graphics processor.
Journal of Computational Chemistry, 2012, 33(2):2554-2565.

A.5. Declaration on Oath

I hereby declare, on oath, that I have written the present dissertation by my own and have not used other than the acknowledged resources and aids.

Hamburg, **Lennart Erik Heinzerling**

A.6. Copyright Notice