Interactive drug-design: using advanced computing to evaluate the induced fit effect

A thesis submitted in partial fulfilment of the requirement for the degree of Doctor of Philosophy

Athanasios Anthopoulos

December 2013

Cardiff School of Pharmacy and Pharmaceutical Sciences

Abstract

This thesis describes the efforts made to provide protein flexibility in a molecular modelling software application, which prior to this work, was operating using rigid proteins and semi flexible ligands. Protein flexibility during molecular modelling simulations is a non-trivial task requiring a great number of floating point operations and it could not be accomplished without the help of supercomputing such as GPGPUs (or possibly Xeon Phi).

The thesis is structured as follows. It provides a background section, where the reader can find the necessary context and references in order to be able to understand this report. Next is a state of the art section, which describes what had been done in the fields of molecular dynamics and flexible haptic protein ligand docking prior to this work. An implementation section follows, which lists failed efforts that provided the necessary feedback in order to design efficient algorithms to accomplish this task.

Chapter 6 describes in detail an irregular – grid decomposition approach in order to provide fast non-bonded interaction computations for GPGPUs. This technique is also associated with algorithms that provide fast bonded interaction computations and exclusions handling for 1-4 bonded atoms during the non-bonded forces computation part. Performance benchmarks as well as accuracy tables for energy and force computations are provided to demonstrate the efficiency of the methodologies explained in this chapter.

Chapter 7 provides an overview of an evolutionary strategy used to overcome the problems associated with the limited capabilities of local search strategies such as steepest descents, which get trapped in the first local minima they find. Our proposed method is able to explore the potential energy landscape in such a way that it can pick competitive uphill solutions to escape local minima in the hope of finding deeper valleys. This methodology is also serving the purpose of providing a good number of conformational updates such that it is able to restore the areas of interaction between the protein and the ligand while searching for optimum global solutions.

Acknowledgements

I lovingly dedicate this thesis to my wife Catherine, who supported me each step of the way and never failed to remind me that “I’ve got to finish this at some point” and little Matilda who emerged half way through the course of this Ph.D. I would like to thank my supervisors Dr Andrea Brancale and Dr Ian Grimstead for their unlimited and valuable support in this three year period, which went well beyond my expectations. I also very much appreciated the help of Professor Chris Jones at the group of Geoinformatics for his help prior to finding this Ph.D. course. In addition, I would like to acknowledge, Professor John Patrick’s from the school of Psychology help and advice during the second year of this course. I would also like to thank fellow Ph.D. students who helped me on various aspects throughout this project and post-Doctoral fellow Antonio Ricci who spent a big amount of time and effort helping me during my first year. I also would like to acknowledge the efforts of Tom and Gaia, the two project students who used the code written during my Ph.D. for their thesis and gave me valuable feedback during this process. Finally I would like to thank my parents and especially my mother in law Helen, who kindly helped us a lot with Matilda during my third year, so that I could have the time needed to accomplish this work on time.

Table of Contents

Chapter 1

Introduction

In this chapter we present an introduction to the thesis and its outcomes. It gives a brief introduction to the terms GPGPU computing, conformational sampling, haptics and haptic protein ligand docking. Finally it provides the motivation for this project, the research hypothesis, aims and objectives. The terms introduced in this chapter are explained in more detail in Chapters 2, 3 and 4, with the appropriate referencing to other works in the literature.

1.1 Introduction

This work consists of a collection of high performance computing (HPC) algorithms targeted at NVIDIA's CUDA architecture. These algorithms were designed to solve the problem of real-time conformation sampling on a haptic protein ligand docking (HPLD) simulation software called Zodiac. The following chapters give an overview of the above terms and assist the reader in gaining a better understanding of the problem we are trying to solve.

1.2 High Performance Computing and GPGPUs

High performance computing has played a very important role in the evolution of molecular modelling applications. Multi-processor Clusters (a set of loosely coupled computers working together to solve a single problem) have enabled researchers to design parallel algorithms for simulating molecular conformations (accurate spatial atomic structures for molecular systems) for the past twenty years[1], [2], [3], [4]. Recent HPC advances have produced accelerators such as graphics cards, NVIDIA's CUDA (Compute Unified Device Architecture)

being an example. In this work, special attention has been paid to the CUDA architecture for its parallel processing capabilities and its extremely high floating point operations to Watts ratio (FLOPs/Watt) as illustrated in Figure 1-1. CUDA has been used in a wide variety of molecular modelling applications and the benefits offered from this architecture to the field of molecular modelling are many-fold. They span from enabling a workstation to achieve cluster level performance to boosting cluster capabilities in thousand-core clusters equipped with hundreds of GPGPU (General purpose graphics processing units often shortened to GPU) cards such as the Blue-Waters super-computer at the University of Illinois.

The advanced performance capabilities offered by these accelerators have been translated into the ability to perform faster conformational sampling in applications such as molecular dynamics (MD), docking or protein folding [5][6]. In addition, enhanced processing power has provided researchers with the potential to incorporate more accurate and computationally expensive force-fields for their modelling applications. Finally, the portability of the GPGPUs has provided the potential to design interactive modelling platforms where the user can see molecular interactions on a screen or even feel the forces applied on a particular molecule during interaction through the use of haptics [7], which we shall now discuss in more detail.

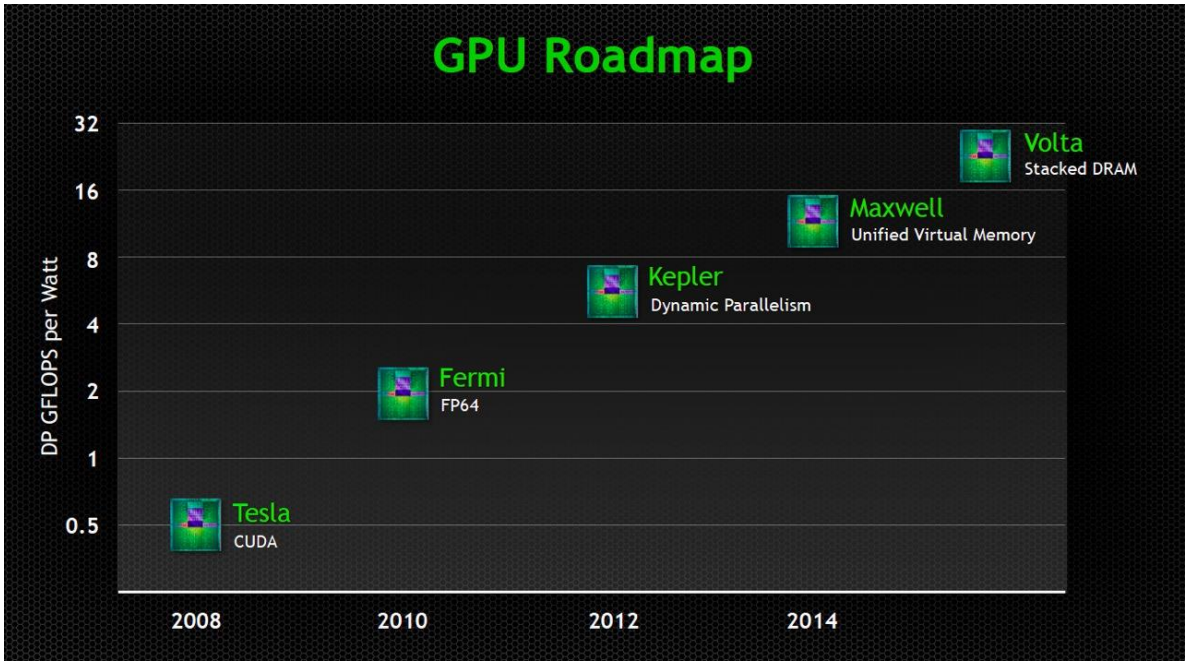


Figure 1-1: The GPGPU road-map. Image courtesy of NVIDIA as presented in 2013 GTC. Volta has now been changed to Pascal!

1.3 Haptic technology

Haptic technology has recently developed as a computer-interface technology that enables the mixing of real-world objects and computer-generated graphics. Haptics refers to the action of sensing and manipulation through touch. Since the early part of the 20th century, the term haptics has been used by psychologists for studies on the active touch of real objects by humans. Exploiting the sense of touch in computer simulation is a strategy already used with success in several other areas, ranging from sculpting to medical training.

With the aid of specific haptic devices, it is possible to control a selected object, such as a ligand in our case, around the translational (x, y, z) and the rotational (Euler angles or yaw, roll and pitch) in three dimensional (3D Cartesian) space. Moreover, these devices are able to convert the forces calculated in the computer simulation into a haptic feedback, which is then directly experienced by the user (such as a force expressed by an electric motor acting against the operator's movement). Several such devices are available to developers, from simple ones that work only in the translational space (three degrees of freedom [DOF]) to the most accurate six DOF, which can also cover rotational space [8]. The force feedback could be efficiently used in CADD (computer aided design capable of dynamic mathematical modelling) simulation experiments, allowing the researcher to feel the forces involved for example in ligand-protein interactions. Indeed, the application of haptic technology to molecular modelling is not a recent idea; one of the earliest successful implementations was reported in the late 1980s [9] .

1.4 The Zodiac molecular modelling platform

Zodiac is a software application able to provide real-time simulation of the interactions between a protein and a ligand. The ligand is controlled with the aid of a Phantom Omni haptic device, which provides six degrees of freedom for the translation and rotation of objects (a ligand molecule in our case) in three dimensional space and three DOFs of force feedback in the X, Y and Z directions of Cartesian space. Force feedback on the rotational DOFs, pitch roll and yaw, is available on haptic devices whose cost at the time of the submission of this thesis would exceed £14,000.

Zodiac has been designed primarily for HPLD simulations. That means that apart from the visual and force feedback that a user can experience during simulation, the user can save important docking results for further investigation. In other words, during a simulation the user can probe the ligand into a protein cavity and save a pose at will, together with the potential energy associated with this particular pose. The application's main advantage over non-human computer interactive applications is the fact that the user can choose the desired pose and save it at will. Humans are more capable of evaluating shapes than the available computer vision algorithms up to the time of submission of this thesis [10]. Zodiac, is taking advantage of the above fact, by providing visual interaction and force-feedback, in order to assist the user into generating the desired system configurations. The force feedback assists in this selection as the user can feel the areas that produce the desired quality of attractive or repulsive forces.

1.5 Zodiac limitations and motivation for this project

There were some limitations to Zodiac's abilities before this work. Specifically, Zodiac was only capable of performing simulations with rigid proteins and semi flexible ligands. In real life molecules are flexible and their atoms may move around in three dimensional space. We will refer to the simulation of this property in computers as conformational sampling. Assuming a system with N atoms, conformational sampling is a 3N dimensional problem. Proteins usually consist of several thousand atoms, whereas ligands are small molecules usually in the range of 30-50 atoms in total. Clearly, protein size determines the scale of the search space and the larger the system is, the higher amount of FLOPS needed during fully flexible conformation sampling and hence slow execution times. That because each attempt to sample a conformation in the 3N-dimensional space requires the computation of energy and force gradients using a forcefield, whose running time complexity originally scales quadraticaly with system size, as illustrated in Chapter 3. This is one reason why previous versions of Zodiac only implemented a rigid protein flexible ligand simulations.

Protein flexibility varies for different bio-molecules. This means that the flexibility of the

protein at its binding site is an important factor for evaluating the fit of a ligand against it and thus the structure of a protein-ligand system pose generated by the previous version of Zodiac can lead into inaccurate conclusions. Protein flexibility can assist or prevent a ligand from binding into its pocket and both of the above scenarios will be misinterpreted by the rigid version of the software. Finally, the potential energy scores are derived from the ligand's atoms and a small part of the protein cavity.

Conformational sampling and potential energy calculations are classified as N-Body problems, assuming N particle systems. This means that keeping the positions of the largest part of the N-body system fixed, will produce results of low accuracy. This problem provided the motivation for much of the work in this thesis. Zodiac along with other HPLD applications listed in Chapter 4, had been showing strong potential in molecular modelling research. However, these solutions lacked the level of accuracy required to make results derived from them reliable towards compound selection for the initial stages of drug discovery, as they were using rigid protein bodies for their simulations.

1.6 Research hypothesis and challenges for this project

As we already explained, the protein flexibility in HPLD simulations is a computationally intense problem. Solutions for this problem had been provided in the literature in the field of molecular dynamics. However, these solutions would only produce the necessary performance for the HPLD problem in many-core clusters, using number of core processors in the range of 500-2000. Using a cluster for interactive visual simulation is not an option, for multiple reasons. The major one is the queuing system used in clusters for prioritizing scheduled jobs. In addition, the operating costs of clusters are high and this could be off-putting for the frequent use of the software. Finally, clusters are not widely available and designing algorithms for clusters would restrict the potential users of the application to those who have access to a cluster.

The introduction of GPGPUs provided the ability to get performance levels similar to a modest cluster in a single workstation. The cost of a good GPGPU card (top of the GeForce line) is relatively low (< 800 £). In addition, power consumption of these cards is low (250W for the

gtx780) and would not generate any significant costs for every consumption. GPGPUs can solve the problems mentioned above as they are portable, cheap to obtain and operate within a user's own workstation.

However, there are still challenges in using GPGPUs for conformational sampling for HPLD. One challenge is, a ligand is probed against a protein cavity at a frequency rate equal to that of the haptic's I/O (input output) set-up. In our case it is set at 33Hz, which is matching the visual output's rendering frequency. This means that between subsequent ligand poses against the protein target, there is a strict time limit to provide a stable conformation for the system. In the case of probing the ligand close to a protein's pocket, high forces are induced on both molecules and this means that the system needs to map all the transient reactions to the molecules up until a stable conformation is found. This constitutes an additional challenge in visualizing the system's interactions as the above mapping should ideally be performed before the new ligand position is probed by the user.

A second challenge faced in this project was the force-field choice. As we explain in detail in Chapter 3, the way to sample conformations is by moving atoms in Euclidean space such that one minimizes an objective function consisting of a series of equations for bonded and non-bonded interactions. This objective function is the force-field. Our force-field of choice, MMFF94s, requires a substantially higher amount of FLOPS to compute compared to other popular force-fields used by other modelling applications as we explain in more detail in Chapter 3.

During the planning phase of this work, the available literature in HPLD was reviewed and a summary of it was contributed in Ricci *et.al* [7], a perspective article for the haptics present use and future potential in molecular modelling applications. However, no application at the time was making any attempts towards flexible HPLD. We then performed an extensive literature review in the areas of flexible (non-human-interactive) docking and molecular dynamics. The former were applying techniques to approximate flexibility in the binding pocket area, whereas the latter were using GPGPUs to provide accurate conformational sampling.

Early in this work we faced a dilemma, which was whether to follow the MD approach or less computationally intensive ideas of approximating flexibility in the pocket provided in the flexible molecular docking literature. The factors that helped us formulate our decision were the fact that our problem was not facing other challenging requirements that docking software do, such as virtual screening and the automatic generation of protein ligand poses. The latter is a multidimensional problem on its own with high processing power requirements, which in our case is solved by the user's ability to evaluate shapes. On the other side, our requirement to provide accurate system conformations at subsequent protein-ligand poses using a computationally demanding force-field was an antagonistic factor towards the MD approach.

The choice was in favour of the MD approach. That because, the MMFF94s choice of force-field requirement of this project would have no rational on an approach approximating cavity configurations in space. In other words, there would be no point of modelling configurations with an accurate and computationally expensive forcefield, while introducing high levels of approximation, by focusing on computations and flexibility around the binding pocket.

On these grounds our research hypothesis was formulated as follows:

"Is it possible to design algorithms for the latest CUDA architecture that would perform such that an HPLD simulation could run in Zodiac with accurate conformational updates modelled with the MMFF94s force-field for systems up to 30 000 atoms, matching the visual rendering rate of 33 Hz?".

It makes sense that there are some fuzzy factors in the above hypothesis. The choice of GPGPU card can have a major impact on performance. For reasons explained in Chapters 6-8, we will use the GTX 680 and 780 of the Kepler line of GPGPU cards as the cards of choice to answer the above research question. These cards retail at 300-450£ at the point of submission of this thesis. The second fuzzy parameter is the size of systems we target for these simulations. The aforementioned solution choice is system size dependent and there will be a point that the time-limit for protein-ligand interactions will not be sufficient for the force-field calculations that are expected to scale linearly with system size. The motivation of this project was to

simulate the induced fit effect on protein-ligand systems for the purposes of research in the pharmaceutical sciences. The induced fit effect is the impact caused by the ligand positioned inside a protein cavity. In some extreme cases the protein cavity can only be formed during interactions with specific ligands. Most of the systems we intended to simulate range from 2K – 30K atoms. Therefore we will set as a requirement a ceiling of 30,000 atom systems including hydrogens, which covers a wide range of protein sizes in drug design research.

Regarding the force feedback on the haptic device, it makes sense that it would be impossible to provide feedback at a 1,000Hz update rate, which is the rate required by the haptic device to provide smooth force feedback. However, aiming to generate force updates at the rate of 33Hz could be sufficient for an acceptable force feedback and feasible to provide with the new version of our code. The feedback granularity could then potentially be improved with the appropriate interpolation technique, providing the desired 1,000Hz update rate.

1.7 Aims and Objectives

Drawing upon our research hypothesis and the brief introduction we provided for our research problem, we will formulate two objectives we need to accomplish in order to be able to solve our research problem and provide a positive answer to our research hypothesis for the targeted system sizes and GPGPU chipsets.

1. Design fast and accurate algorithms for the computation of the MMFF94s force-field, able to complete a series of energy and force calculation functions within the 33 ms time-limit.
2. Design efficient energy minimization algorithms able to generate accurate conformations for systems up to 30000 atoms within the 33Hz requirement.

For the rest of this thesis we will refer to the above objectives as the first objective and second objective respectively, matching the listing above.

1.8 Contributions

The list of contributions of this thesis is listed below:

1. A cell-list approach for the computation of non-bonded interactions consisting of an

adjustable irregular grid decomposition for neighbour cell search, whose cells aspect ratio can be adjusted by the user prior to the simulation

2. A new meta-heuristics method for fast potential energy surface exploration for biomolecules designed to solve our HPLD problem.
3. The first application to provide fully – flexible HPLD for system sizes up to 30, 000 atoms

The above contributions are explained in detail in Chapter 6 and Chapter 7.

1.9 Thesis overview

The rest of this thesis is organised as follows. In Chapter 2, we provide the necessary background for the CUDA architecture and its associated programing language. The background is designed such that it provides the necessary knowledge to follow the implementation and outcomes of this thesis. Chapter 3 provides insight and references to our force-field of choice MMFF94s. Chapter 4 contains a critical survey of the most up to date related work (state of the art) of methodologies designed to solve problems similar to the ones formulated in our two main objectives and HPLD in general. Chapter 5 describes initial attempts towards the two objectives of this thesis. Chapter 6 provides the implementation details and results of the final solution provided to the first objective of this thesis, where contributions one to three are explained in detail. Chapter 7 provides the implementation details, results and discussion for the second objective of this thesis as well as detailed information of the fourth contribution. Chapter 8, provides a discussion regarding the usability and limitations of this thesis' outcomes, as well as perspectives and future research recommendations. Chapter 9 provides the concluding remarks and re-states the contributions of this thesis.

GPGPUs and the CUDA Architecture

The work presented in this thesis draws on the evolution of graphics processing units into general purpose parallel computation devices, used to perform energy and force calculations for the MMFF94s force-field. For this reason, this chapter provides an overview of the CUDA architecture as well as information regarding coding practices for the above architecture using the CUDA C programming language.

CUDA has been rapidly evolving during the past 4-5 years. This evolution has produced a continuously improved line of GPGPU chipsets. The findings of this thesis are based on the GK104 / GK110 chips of the Kepler line. As a result of this, the present chapter highlights the new programing features available in these chipsets. Most of this chapter's content is derived from "CUDA C programming guide" and "CUDA C best practices guide" available from NVIDIA [12-13].

2.1 The CUDA architecture.

GPGPUs are frequently defined as shared memory architectures with the ability of multithreaded execution. A brief diagram of the recent GK104 chip is shown in Figure 2-1. The GK104 chip contains 8 streaming multiprocessors (SMX), where each multiprocessor contains 192 CUDA cores. The CUDA execution model is performed in blocks of threads, where each block can contain up to a fixed maximum amount of threads (1,024 for the Kepler line).

Figure 2-1: The GK104 (Kepler line) chip architecture. Image courtesy of NVIDIA

2.2 The CUDA execution model (SIMT)

The execution unit of the GPGPUs is 32 parallel threads, termed a "warp". CUDA executions are triggered from the CPU using a kernel call specifying the number of blocks and threads per block that the kernel is programmed to use. Threads in each block are grouped into warps and dispatched via warp schedulers, where each warp schedule can process warps concurrently. Fermi has 2 warp schedulers and Kepler architectures have 4. When a multiprocessor is given one or more of these thread blocks to execute from the kernel, it partitions them into warps and each of these warps gets scheduled by a *warp scheduler* for execution. Warps are allocated a finite amount of registers each from the register file and threads in a warp are executed in a SIMT manner (single instruction multiple threads). Each core is assigned with the sequential execution of a single thread of a particular warp. Instructions are executed one at a time, for each thread in a warp. When instructions within a warp diverge, such as when a conditional statement causes some threads to execute different code sections to other threads, then we have the divergent warp phenomenon. In

order to exemplify this, we assume a code block with a conditional branch where both paths of the conditional branch are executed by at least one thread. In this case, due to the SIMT nature, threads processing the “else” statement of the conditional are simply ignoring the “if” part of the instructions, but paying the same penalty as if they have executed the code. This phenomenon deteriorates performance as each thread may pay the performance penalty of executing both paths of a branch [12].

2.3 The CUDA memory hierarchy

The CUDA memory hierarchy synopsis is provided in Table 2-1. Global memory is visible to all threads in all blocks and can be cached in the GPU’s L1 and L2. If cached in L2 only it is serviced by 32 byte memory transactions, whereas if cached in both L1 and L2 global memory can be serviced in 128 byte memory transactions. The most important characteristic of this memory type is that it has a relatively low bandwidth when randomly accessed. If it is accessed in sequential order by consecutive threads then the resulting high rates of L2 cache hits improve its bandwidth significantly. Global memory accesses have to be performed in a coalesced manner in 4, 8, 16 or 32, 64 or 128 bytes otherwise transactions are partitioned in two memory fetch orders and performance deteriorates.

2.3.1 Shared Memory

The shared memory resides in L1 cache on-chip and is configurable either as a 16KB L1 cache and 48Kb of shared memory or vice versa. In Kepler it can also be conFigured as 32KB of shared memory and 32Kb L1 cache. Shared memory is visible by all threads of a single block and its read/write bandwidth is very fast. There is no need to be accessed in sequential order as it is cached. The shared memory bandwidth is 32 bits per clock cycle. If two or more threads are accessing words from the same shared memory bank, then there is a bank conflict and access to the requested word is serialized. However, if all threads in a warp access the same bank, the word is broadcast to all threads and there is no bank conflict.

2.3.2 Constant memory

There is a total of 64KB of read-only memory cached on the device. The requirement for

achieving the high access bandwidth of this memory is that all threads are reading the same memory location. In Kepler and Fermi architectures the “load uniform” instruction is also supported which means that when two or more threads of a warp are accessing the same global address the fetch is broadcast amongst them.

2.3.3 Registers

Registers are the fastest type of memory and they are only visible by their host thread. For Fermi and Kepler there are 63 registers available per thread. If a kernel attempts to use more, the compiler spills the excess data to local memory whose access is at the rate of global memory. For Kepler architectures and above, register values can be visible and exchanged with other registers of threads belonging to the same warp. This is done by the newly introduced shuffle operation whose bandwidth is that of a register operation.

Memory	Location on/off chip	Cached	Access	Scope	Lifetime
Register	On	n/a	R/W	1 thread	Thread
Local	Off	†	R/W	1 thread	Thread
Shared	On	n/a	R/W	All threads in block	Block
Global	Off	†	R/W	All threads + host	Host allocation
Constant	Off	Yes	R	All threads + host	Host allocation
Texture	Off	Yes	R	All threads + host	Host allocation
† Cached only on devices of compute capability 2.x.					

Table 2-1: CUDA memory hierarchy. Source: CUDA C programming guide (NVIDIA).

2.4 Occupancy

Occupancy is the ratio of the number of active warps per multiprocessor to the maximum number of possible active warps. Thread instructions are processed sequentially and in order within a warp, but if an operand is not ready the warp will stall. Context switching to another warp for processing is the way that CUDA hides latencies, such as from memory reads/writes. High occupancy ratios assist this approach. Occupancy ratios up to 50% are known to help realise the full performance potential of a GPU, while occupancies higher than 50% have not

been shown to add further performance gains. However, low occupancy rates known to result in performance loss and should be avoided if possible.

One factor that affects occupancy levels is register usage within a kernel. The register file is a limited resource that all blocks resident in a multiprocessor have to share. Registers are allocated in a thread block all at once. This means that the more registers a kernel use, the fewer thread blocks can be resident in a multiprocessor at a given time, lowering the occupancy levels. The size of the thread block is another factor affecting occupancy as registers are allocated per thread, so the total amount of threads used per block is directly proportional to the number of threads contained in it. Similarly shared memory usage is a factor affecting occupancy levels as the L1 cache is also a limited resource per SMX.

Latencies can also be hidden using instruction level parallelism (ILP) by using more on-chip memory as long as it is within the hardware limits. This constitutes an alternative to achieving higher occupancy levels by limiting the amount of on-chip memory usage. It is difficult to argue at this point which approach is superior, as it seems that it depends largely on the nature of the problem and the algorithmic solution applied. NVIDIA recommends higher occupancies for memory bound kernels, while higher occupancies can benefit computationally bound kernels too, but ILP can also be considered [13].

2.5 CUDA streams

The recent generation of Kepler cards (GK110) has introduced new features in GPGPU computing and one of them is the ability to run up to 32 kernels asynchronously, achieving execution overlaps very close to that of a parallel execution. This feature introduces an additional dimension of parallelism in CUDA and initiates the need of new optimization techniques in CUDA-parallel programming. To be more specific, considering this new feature, an algorithm designer should not treat each kernel individually any more, but as part of a wider context able to utilize the graphics card more efficiently.

However, there are some rules and constraints on the way that GPGPUs parallelize kernel execution [12-13]. At the moment, Nvidia GPGPUs are able to run 16 scheduled blocks per streaming multiprocessor (SM). In the case of a K20 card, that gives a total of 208 blocks able

to run concurrently at any time in its 13 SMs. From a resources perspective, the shared memory limits are bound to 48kB. This means that the total amount of shared memory used by the asynchronously scheduled kernels in each stream should not exceed the above limit. If it does, then the parallel execution is restricted to the number of streams whose total shared memory usage is below the 48kB limit. Regarding register usage, similar restrictions apply and if the total number of registers used per thread exceeds the card's limit (255 for GK110s) then some register overspill might take place and the slower performing local memory might be used. Finally overlapping streams can only write to independent memory addresses, but this is not a significant constraint as arrays can be copied into multiple buffers for each stream to operate individually upon them.

2.6 Summary

This chapter provided the fundamentals of the CUDA architecture and its associated programming model. In particular we referred to the CUDA execution model, explaining the terms threads, blocks and computational kernels. In addition we provided information regarding the memory hierarchy available in CUDA, along with its performance characteristics. Finally, the terms stream execution and multiprocessor occupancy were also explained. In Chapters 5-7 we list a number of algorithms and methodologies targeted for the CUDA architecture. The algorithms' implementation description is based on CUDA's properties as explained in this chapter. Therefore this chapter can act as an initial guide to the reader, in order to be able to follow the outcomes of Chapters 5-7.

The MMFF94s Force-field

This chapter gives an overview of the MMFF94s forcefield, which is used as a modelling tool for our software and gives an overview of bonded and non-bonded interactions. It also provides an indication to the level of complexity of this force-field's non-bonded interactions due to its Van Der Waals interactions modelling approach.

3.1 Molecular force-fields

Force-field methods assist in molecular mechanics computations by providing the necessary equations to calculate the total energy of the system as a function of its nuclear positions. They are used extensively by many software applications in the field of molecular docking and molecular dynamics. There are many variations of the force-field equation, whose general form is (Equation 3-1):

$$\sum E_{total} = \sum E_{Bonded} + \sum E_{Non-Bonded}$$

Equation 3-1: The general force-field equation for the total of bonded and non-bonded terms

3.2 The MMFF94s Force-field

Zodiac uses the MMFF94s force-field for modelling molecular conformations, as it is recommended for use in pharmaceutical research. This forcefield is explained in detail in a series of publications [15 - 21]. The present chapter highlights the equations and other

important modelling properties listed in the above mentioned publication series. For MMFF94s the general force-field Equation 3-1 is unrolled as follows:

$$\sum E_{Bonded} = \sum EB_{ij} + \sum EA_{ijk} + \sum EBA_{ijk} + \sum EOOP_{ijkl} + \sum ET_{ijkl}$$

Equation 3-2: The bonded energy potential

$$\sum E_{Non-Bonded} = \sum EVdW_{ij} + \sum EQ_{ij}$$

Equation 3-3: The non-bonded energy potential

The terms of Equation 3-2 and Equation 3-3 are explained in the following sections. The following equations have been taken from MMFF94s publication series [15-21]. In these equations some constants are represented by two letter combinations, which might confuse someone from a mathematical background. However, this is the best way to describe them as it is the industry standard for the specific forcefield.

3.3 Bonded Interactions

The bonded interactions are the trivial part of each force-field and can be derived at linear time O(N), for an N-atom system. This is since bond information is provided and the interactions consist of bonded atom pairs for bonded forces, triplet groups for the angle and stretch bend forces and 4-atom structures for the dihedral and out of plane potentials.

3.3.1 Stretching Energy $\sum \boldsymbol{EB_{ij}}$

The first term of Equation 3-2 is the bond stretching energy between two bonded atoms *i,j* at an inter-atomic distance r between them as shown in Figure 3-1.

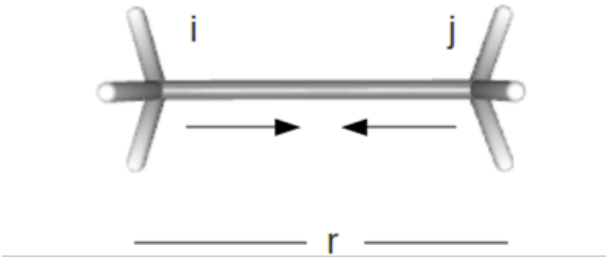


Figure 3-1: The bond stretch interaction between 1-2 bonded atoms i and j.

In Equation 3-4 (below), Δr_{ij} denotes the deviation of $r_{ij}$from a reference bond length r_0. $kb_{ij}$is a constant whose value depends on the type of atoms *ij,* and *cs* (cubic strength) is a constant whose value is -2Å.

$$\sum EB_{ij} = 143.9325 \frac{kb_{ij}}{2} \Delta r_{ij}^2 (1 + cs\Delta r_{ij}) + \frac{7}{12} cs^2 \Delta r_{ij}^2$$

Equation 3-4: the bonded energy potential between two bonded atoms

3.3.2 Bending Energy $\sum EA_{ijk}$

In Equation 3-2, the term $\sum EA_{ijk}$refers to the energy related with the binding angle between two bonds formed by three 1-3 bonded atoms ijk at an angle θ_{ijk} as shown in

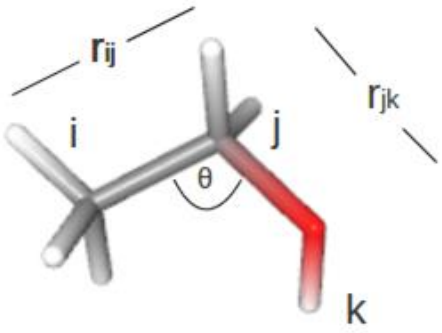


Figure 3-2 below.

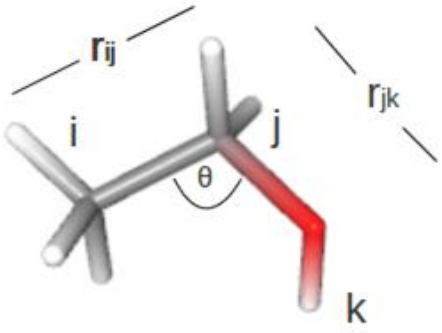


Figure 3-2: The bending interaction between 1-3 bonded atoms i, j and k.

The bending energy is given by Equation 3-5 below,

$$\sum EA_{ijk} = 0.042844 \frac{ka_{ijk}}{2} \Delta\theta_{ijk}^2 (1 + cb\, \Delta\theta_{ijk})$$

Equation 3-5: The bending energy potential

where ka_{ijk} is a constant whose value depends on atoms *ijk* and $\Delta\theta_{ijk}$is the deviation of $\theta_{ijk}$from a reference angle value θ_0 for the atoms *ijk*. The constant *cb* is the cubic bend constant and equals $-0.4\ \mathrm{rad}^{-1}$.

3.3.3 Stretch-Bend Interactions $\sum \boldsymbol{EBA_{ijk}}$

In Equation 3-2, the term $\sum EBA_{ijk}$ refers to the energy related with the stretching of two bonds formed by three atoms *ijk* at an angle θ_{ijk} as shown in Figure 3-3 below.

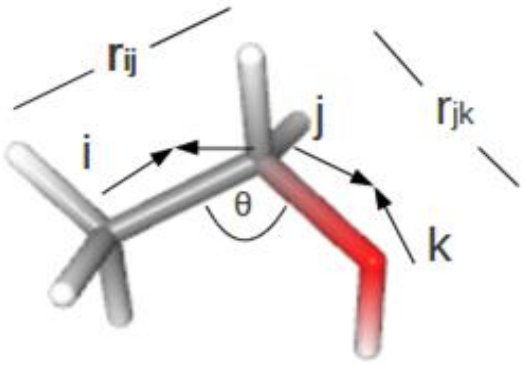


Figure 3-3: The stretch bend interaction between three 1-3 bonded atoms i, j and k.

The stretch-bend energy is given by the Equation 3-6 below,

$$\sum EBA_{ijk} = 2.51210(kba_{ijk}\Delta r_{ij} + kba_{kji}\Delta r_{kj})\Delta\theta_{ijk}$$

Equation 3-6: The stretch bend energy potential

where kba_{ijk} and kba_{kji} in Equation 3-6 are force constants that depend upon the stretches of *i-j* and *k-j* to the *i-j-k* bend respectively. Δr is the deviation of the interatomic distance from the reference value r_0 and $\Delta\theta_{ijk}$ have been already defined in Equation 3-5 above.

3.3.4 Out of Plane Bending Energy

The term $\sum EOOP_{ijkl}$ refers to the non-planar tri-coordinate atoms *l* at an angle x (Wilson angle) between them and the plane formed by atoms *kji* on a planar set-up like an aromatic ring.

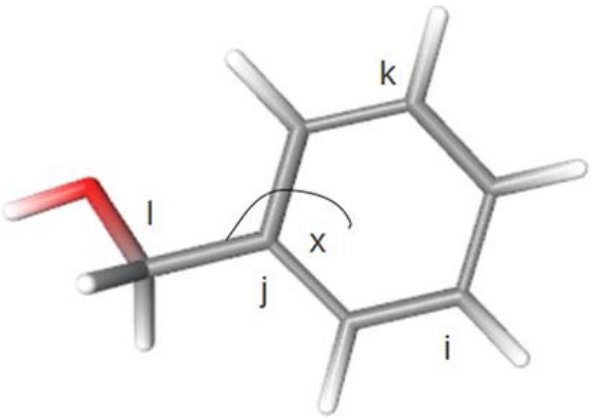


Figure 3-4: The out of plane interaction between atom l and the plane formed by atoms ijk

The Out of Plane Bending Energy is given by the Equation 3-7 below,

$$\sum EOOP_{ijkl} = 0.043844 \frac{koop_{ijkl}}{2} x_{ijkl}^2$$

Equation 3-7: The out of plane energy potential

where koop is a constant depending on atoms *ijkl* and x is the Wilson angle between the *l-j* bond and the *ijk* plane. The out of plane potential (improper dihedrals) is one of the distinct characteristics of the MMFF94s force-field that cannot be found in other commonly used ones such as CHARMM [22], AMBER [23], Gromos [24] and others.

3.3.5 Torsional Energy

The term $\sum ET_{ijkl}$ refers to the torsional energy between 4 bonded atoms.

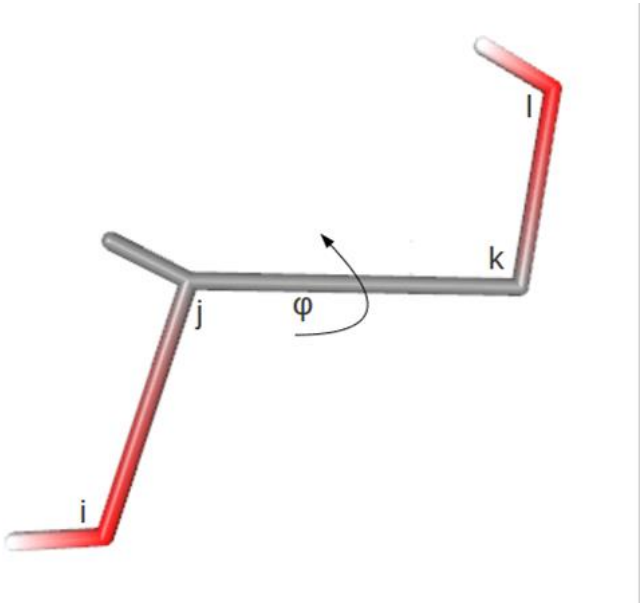


Figure 3-5: The torsional interactions between 1-4 bonded atoms ijkl

The torsional energy is given by the Equation 3-8 below:

$$\sum ET_{ijklk} = 0.5(V_1(1 + \cos(\varphi)) + V_2(1 - \cos(2\varphi)) + V_3(1 - \cos(3\varphi)))$$

Equation 3-8: The torsion energy potential

where the angle ϕ (schematically represented in Figure 3-5) is the dihedral angle formed by the two planes *jkl* and *ijk* . V_1, V_2, V_3 $(kilocalories/mole)$, are three constants depending on atoms *ijkl,* where *i-j*, *j-k* and *k-l* are bonded pairs and *i* is not equal to *l*.

3.4 Further Comments

In reality bond lengths and bond angles do tend to deviate from their reference bond length $r_0$and reference bond angle $\theta_{ijk}$and Wilson angle x (for chemical structures that form a plane) only by a very small factor. For this reason, some modelling applications who focus solely on performance would not perform calculations on bending, stretch bend, angle bend and out of plane energies due to their restricted degree of freedom. However, this is not a

recommended method to apply to large protein macromolecules as small angle deviations may cause big moves further down the chain [14].

3.5 Non- Bonded Interactions

The non-bonded interactions constitute the challenging part of the force-field calculation. Contrary to bonded interactions, their asymptotic complexity is of order O(N^2) as in theory, each atom interacts with every other atom in the N-body molecular system.

3.5.1 Van Der Waals Energy

For a pair of atoms *i* and *j* , where the distance between them is r_{ij}, MMFF94s adopts a special "Buffered-14-7" (Buf-14-7 Equation 3-9) form for the Van Der Waals (VDW) potential instead of the Lennard-Jones or Exp-6 potentials used in other force fields such as the popular AMBER, CHARMm and GROMOS [22 - 24]. The reason is that the aforementioned potential is capable of producing more accurate results as Halgren 1992 proved [20]. This potential is given by:

$$\sum EvdW_{ij} = \varepsilon_{ij}\left(\frac{1.07\mathrm{R}_{ij}}{r_{ij}+0.07\mathrm{R}_{ij}}\right)^7\left(\frac{1.12\mathrm{R}_{ij}^7}{r_{ij}^7+0.12\mathrm{R}_{ij}^7}-2\right),$$

Equation 3-9: the Van Der Waals energy potential

where $R_{ij}$is the minimum-energy separation in angstroms and ε_{ij} is the well depth in kilocalories per mole. The minimum energy separation R_{ij} is derived through the “augmented arithmetic mean” as in Equation 3-10. However, according to Halgren [16], a very good approximation of the potential can be given by the “cubic mean” of Equation 3-11:

$$R_{ij} = 0.5(R_i + R_j)(1 + B(1 - \exp(-12\gamma_{ij}^2)))$$

Equation 3-10: “Augmented arithmetic mean"

B is equal to 0.2 or 0.0 if one of the atoms is polar hydrogen and γ_{ij} is given by Equation 3-12.

$$R_{ij} = \frac{R_i^3 + R_j^3}{R_i^2 + R_i^2}$$

Equation 3-11: "Cubic mean"

$$\gamma_{ij} = \frac{R_i - R_j}{R_i + R_j}$$

Equation 3-12: The Gama value

The well depth ε_{ij} is given by:

$$\varepsilon_{ij} = \frac{181.16 G_i G_j (a_i a_j)}{((a_i/N_i)^{1/2} + (a_j/N_j)^{1/2}) R_{ij}^6}$$

Equation 3-13: The well-depth

where $a_i$is atomic polarizability of atom *i*, N_i and N_j are the Slater-Kirkwood effective numbers of valence electrons and G_i, G_j and A_i are scale factors. In addition, if atom pair *i-j* is classified as a donor-acceptor interaction, R gets further scaled by the factor DARAD = 0.8, and the well depth ε_{ij} gets scaled as given by Equation 3-13 and calculated using the unscaled R, by the factor DAEPS = 0.5.

At this point we can observe that in order to calculate this potential we need to include a degree of conditional branching to account for the above predicates. In a case of a warp of 32 threads, where each thread evaluates the force of the atom *i* held by thread *Ti*. If at least one of the atoms inside a warp is not a polar hydrogen, then all threads in the warp will be paying the penalty of the exponent. Moreover, if there are various combinations of hydrogen bond donating or receiving between the atom pairs that the warp is going to evaluate, then the penalty of executing operations inside many conditional branches is going to be paid during execution. This is a challenging part to implement efficiently in CUDA. A good approximation of the potential can still be given by simply replacing the expensive "augmented arithmetic mean" in Equation 3-10, with the computationally faster "cubic mean" of Equation 3-11. The latter behaves similarly when *Ri* and *Rj* differ by less than a third, which is usually the case.

The fact that the minimum energy separation and well depth are derived at run time is a characteristic that distinguishes MMFF94s and other algorithms that use the simplified

Lennard-Jones potential for the approximation of Van Der Waals forces. In [16], [20] it is shown how this version of VDW calculations contributes into more accurate free energy results.

3.5.2 Electrostatic Interactions

The electrostatic interaction between two non- bonded atoms is given by the following equation,

$$\sum EQ_{ij} = f * 332.0716 \frac{q_i q_j}{D(R_{ij} + \delta)}$$

Equation 3-14: Electrostatics potential

where *D* is the dielectric constant for which the default value is 1. q_i and $q_j$are the MMFF94s partial charges on atoms *i* and *j*, $R_{ij}$is the inter-atomic distance, δ is the "electrostatic buffering" constant of 0.05A. The scaling factor f is 0.75 for 1-4 interactions, and 1.0 otherwise.

3.6 Force calculations and conformational updates

Forces are calculated as the negative gradient of the energy potential of each atom *i* in the system $-\nabla E_i$. Once the force vector $\vec{F}_i$ is available for each atom *i*, a scalar value λ (step-value) needs to be found in order to update the system into the new conformation. The conformational update is performed for each atom in the system in Euclidean space as follows:

$$\overrightarrow{C'}_i = \overrightarrow{C_\iota} + \lambda \vec{F}_i$$

Equation 3-15: The conformation update formula

Where $\overrightarrow{C'}_\iota$ is the updated vector of an atom *i* and $\overrightarrow{C_\iota}$ is its position before the update.

3.7 Approximation using the cut-off convention

Figure 3-6a shows a graphical representation of Van der Waals energy against distance. From the graph we observe that energy decays rapidly after a certain distance r_c. It is obvious that Van Der Waals forces can be computed at a very good approximation following the constraint that the computation will only apply to atoms whose distance is less or equal to some cut-off. This cut-off value is usually in the range of [9-12.5] Å. We can metaphorically say that the cut-off value acts like a dimmer switch that balances speed of execution with accuracy.

Looking at Figure 3-6 b we can observe that electrostatic forces decay more slowly than the VDW forces beyond r_c, but a good approximation of the potential can still be obtained for the above cut-off distance range. A common practice for calculating the non-bonded potential employed in MD packages is to split the calculation domain into short range and long range forces (the ones including atom pairs further than cut-off distance apart). Short range forces are computed for both electrostatic and VDW potentials. Long range potentials are only computed for the electrostatics, which can be approximated by algorithms such as Particle Mesh Ewald in O(NlogN) [25] or Multilevel summation in O(N) [26] on GPGPUs.

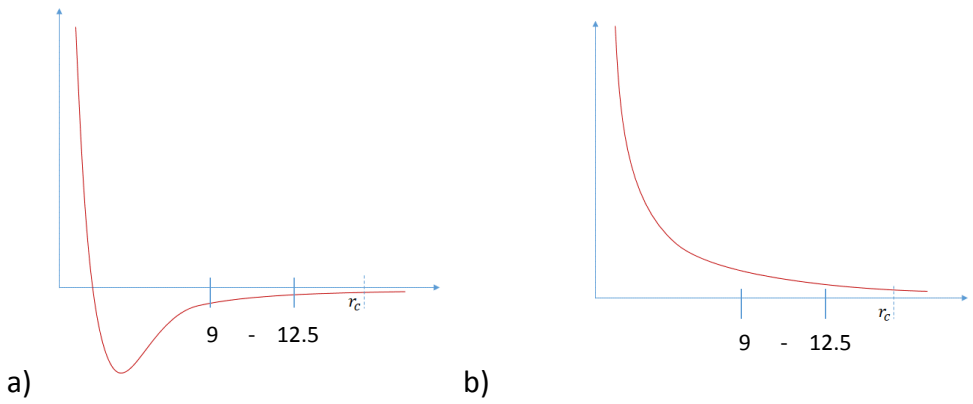


Figure 3-6: The Van der Waals potential energy plot on the left (a) and the electrostatic energy potential plot on the right (b).

3.8 Summary

This chapter described in detail our modelling force-field of choice, the MMFF94s. In particular, in this chapter we explained the equations used for both bonded and non-bonded

interactions. We also provided details of the out of plane potential, which is used to model dihedral angles between an atom and the plane of a ring structure. We also explained that MMFF94s's non-bonded potential requires more FLOPs to compute compared to L-J based forcefields, due to its Van Der Waals interactions mapping. Finally we explained how the quadratic complexity of the forcefield computation can be reduced by using the cut-off convention as well as providing information on the degree of approximation when this technique is employed. This chapter can be a useful guide for Chapter 6 where we explain the algorithms designed for the computation of the atomic forces and energy potential of protein-ligand systems using the aforementioned force-field.

Related Work Overview

This chapter is an overview of the state of the art. It reviews the latest and most innovative approaches in areas relevant to this work. In particular, we review the most notable Haptic-driven protein–ligand docking (HPLD) implementations so far with a small reference to other relevant modelling areas such as steered molecular dynamics. Next we review the state of the art in regards to parallel algorithms in CUDA for the non-bonded interactions calculation problem. A description of bonded interactions implementations follows. Finally, a section (4.7 – 4.8) describing various techniques for potential energy surface exploration is listed to provide some insight into evolutionary strategies and their usage in molecular docking applications.

4.1 Haptic protein–ligand docking

Haptic-driven protein–ligand docking (HPLD) refers to molecular modelling simulations that use a haptic device to control the movement of the ligand against a protein's active site. One of the main driving factors that contributed towards the development of this technique is that the human brain is still better in understanding shape interactions than the computer [27]. However, to really gain the full advantage of human abilities, it is necessary to provide the user with a very natural experience: this means a stable force-feedback, with updates of 1 kHz and a smooth visual representation (updates of 33 Hz). The latter factor has been the first to be tackled and various strategies have been implemented during the years to perform faster energy minimizations and emulate forces and conformations mainly on the ligand. Table 4-1 lists some of the most notable HPLD and IMD (Interactive Molecular Dynamics) applications.

software	Description	Availability	ref
VMD	IMD	Proprietary	[11]
Haptimol	Haptic – interactive molecular modelling	Freeware	[28]
HMolDock	Iterative evaluation of L-J. Force feedback	Proprietary	[29]
MDDriver	IMD	Freeware	[30]
Kinimmerse	Exploring local conformations with the aid of haptics	Freeware	[31]
Samson	HPLD with binary assembly tree technique for molecule flexibility	Proprietary	[32]
Zodiac	HPLD with H-bond highlighting	Freeware	[10]

Table 4-1: HPLD and IMD software

One of the most successful HPLD techniques for force calculation to date has been that of potential grids to describe the energy profile of the binding site. The energies obtained from the grid calculation can be translated into gradient vectors, which are finally converted into force values through an interpolation algorithm [33]. Two remarkable examples of software using such an approach are the chemical force-feedback system [34] and HAMStER [35]. This strategy considerably sped up the energy evaluation, making the required smooth haptic-force feedback feasible to implement for these kinds of programs. However, the grid approach was not able to manage large molecular systems (such as enzymes), forcing researchers to focus their software development in other directions.

A different way to overcome the HPLD drawbacks can be found in the software developed by Subasi and Basdogan of Koc University [36]. Their approach consists of a rigid ligand–rigid receptor exploration, where the user can control the translational and rotational movements of the ligand with the aid of a six-DoF haptic device. This first part of the experiment gives the possibility to the user to explore all the potential cavities of the receptor in order to find the best candidate for his or her lead molecule. An innovative implementation, called the 'active haptic workspace', allows the user to continuously explore all of the surface of the receptor through an automatic shift of the coordinates once the haptic pointer reaches the limits of

the active haptic workspace. Once the user selects a particular cavity and an approximate position of the ligand inside it, the software performs a small MD simulation to correctly fit the ligand inside the cavity. To test their program, the authors performed two experiments: the first was focused to test the effectiveness of the application for the proper understanding of the active sites of the protein; the second aimed to identify whether the system would help to identify the correct pose (position and orientation) of the ligand against a given active site. In both cases, the results obtained were very close to the references taken from the Protein Data Bank (PDB) archive.

4.1.1 Zodiac

A further improvement in the HPLD field is Zodiac [10], designed in our laboratory, capable of handling protein–ligand interactions in real time using the rigid protein-flexible ligand approach. In this application, the six-DoF haptic pointer is fixed to the centre of mass of the ligand. Its conformational flexibility is performed on the dihedral angles only, leaving the angles and bond values of the selected compound fixed. Such flexibility is controlled by a minimization algorithm (steepest descent), which drives the system, step by step, to lower potential energies. Although this is more computationally expensive than the potential grid approach, it has the advantage of helping users during their experiments. While the experimenter is moving the ligand to probe the protein's surface, the total force of the system is calculated through the MMFF94 force-field and the total force component for the ligand atoms is fed back to the user through the haptic device. The application provides force values to the device at a rate of approximately 1 KHz to match the feedback device's requirements (with the aid of an interpolation algorithm). Zodiac is also capable of visualizing the energy contribution of individual atoms by using a colour code. Another visual effect of the software that could be effective for drug-design investigations is the visualization of the hydrogen bond between the protein and the ligand, which is displayed as a red line. This is helpful as it highlights the positions generating dipole-dipole attractions between atoms of the two molecules. These attractions constitute the generation of strong attractive forces and this illustration can help the user identify whether a strong attraction is generated due to hydrogen bonding or not.

More recently, NANO-D research developed a new suite of programs to interactively implement the haptic input in the simulation of biomolecular complexes. This package, called SAMSON [32], contains several new features for the treatment of the molecular systems. Of these features, the binary assembly tree approach [32] is of particular interest: the investigated molecule is split into several rigid subgroups, which are chosen by the user according to his/her requirements. The different subgroups are then joined together through flexible bonds that give the correct flexibility to the overall molecule. The overall result was a fast algorithm that sped up the force calculations required for a smooth haptic feedback.

Most of the research in the past has focused on ways to approximate forces and ligand conformations. Some attempts have been made towards protein flexibility but this would mainly include techniques to emulate conformational sampling in the binding pocket area, ignoring the rest of the intermolecular interactions. The running time complexity of the HPLD problem is high enough to discourage researchers from calculating the force-field for the whole system and introducing full protein – ligand flexibility during protein ligand docking experiments, the problem we address in this work.

4.2 Algorithms for Molecular Mechanics Computations

The following sections list algorithms for the full force-field computation. The first section lists algorithmic techniques for the computation of the non-bonded interactions. The second section provides a brief review on the more trivial algorithms for bonded interactions. Both sections mainly focus on algorithmic techniques used for the CUDA architecture and programming interface.

4.2.1 Non-Bonded Interactions

This section contains a critical literature review on the topic of the computation of the non-bonded interactions. Initially, this problem appears to have a quadratic running time complexity (O(N^2)). However, this problem, as mentioned in previous chapters can be decomposed into short range and long range non-bonded interactions. The short range part computes interactions within the range [0, r_c] and the long range part computes interactions beyond r_c, where, r_c is a predefined cut-off distance. For cut-off distances between 9 to 12.5

Å, a very good approximation of VDWs and electrostatics can be obtained as explained in section 3.7.

4.2.2 Neighbour list algorithms

Neighbour list algorithms have been implemented in the past on clusters [1, 4, 37, 38] and recently on GPGPUs [39 - 41]. They are based on a neighbour list built per atom. The algorithm has been invented from Verlet [42] hence the algorithm is also referred to as Verlet-lists. The purpose of the neighbour list is to reduce the running time complexity of free energy and force calculations for the non-bonded forces with cut-off, from O(N^2) to O(CN), where C is the maximum number of neighbours that an atom can have for a given simulation. The most complex part of the algorithm is the actual neighbour list creation. This is often performed using a space decomposition scheme. Atoms are placed in bins of $r_c{}^3$ volume forming a three dimensional regular grid of cubes containing the atoms of the macromolecule. This spatial arrangement helps in reducing the running time complexity of the following step from O(N^2) to O(C'N), where C' is the maximum number of atoms that the 26-connected neighbour cubic bins can hold. The next step consists of loading a home bin and all of its 26-connected neighbours. The spatial relationships between the home bin and its 26-connected neighbours make sure that, for each atom in the home bin, its whole set of neighbours are contained in this load. Any other bin in the grid is further than a cut-off distance away. On a GPU. each home atom's coordinates can be held in registers by a single thread, and each 26-connected neighbour bin is loaded into the GPU's shared memory, one at a time. Then each thread iterates through all atoms in the bin evaluating the condition $r_{ij} \leq r_c$, where r_{ij} is the distance between the home atom *i* and a neighbour candidate *j* held in shared memory. The indices of neighbours succeeding the conditional are written to atom's *i* neighbour list. The neighbour list is then used for a few time steps to perform the short range non-bonded forces calculations. Neighbour list updates are performed frequently, usually after a fixed number of time steps. Free energy and force calculations are then performed using an atom decomposition scheme where each thread is assigned one home atom and iterates through all of its neighbours to perform the calculations.

4.2.3 Notable GPGPU implementations

Liu et.al implemented a version of this algorithm [39 - 40]. Bin sizes on their implementation were $r_c + \delta$ where δ is termed as a skin value. Neighbour atom pairs *i,j* were built satisfying the conditional $r_{ij} \leq r_c + \delta$. The purpose of the skin value δ was to assist with neighbour list updates. During position updating of atoms in each step a record of their displacement away of their initial position was kept. If the displacement was greater than δ, this would trigger a neighbour list update. The greater the skin value the less frequent the updates would be. This method ensures that no step is performed with an overly outdated neighbour list. The keeping of displacement records does require some extra floating point operations and global memory, which in turn add an overhead to the overall algorithm performance.

Neighbour lists compatibility with GPGPUs has been questioned by many authors in the past [51, 75]. GPGPUs' performance on random global memory accesses has been poor up to the date of this thesis' submission. Storing the neighbour lists and fetching data from it, requires out of order global memory fetches. The same applies for writing back the force values to each neighbour especially in the case that Newton's third law is applied. Anderson *et.al* [41] implemented a Verlet-list version in the GPGPU trying to address the out of order read/write problem. They tried to solve this problem by spatially rearranging the atom indexes using a Hilbert curve spatial sorting algorithm [43] and improved the in order access of the neighbour list and the L2 cache hit rate. This technique achieved up to 4X speed-ups for systems of greater than 50,000 atoms. It is worth mentioning that the original ordering of atoms in the input file follows the order of the residues in the peptides and nucleic acid chains. That in turn achieves good sequential access rates for the bonded forces calculations and some MD packages also exploit this ordering to handle 1-4 bonded atom exclusions.

Scattered writes on these algorithms have been avoided by not taking advantage of Newton's third law. This means that for every atom *i*, neighbours are gathered around all its 26 neighbouring bins. Then for every valid atom pair *i-j* there exists its mirror pair *j-i* which has to be re-calculated leading into doubling the calculation workload. The benefit of it though, is that memory writes for each atom forces are performed in-order. In addition, a great amount of costly atomic operations is avoided as if Newton's third law was implemented, then both *i-*

j and *j-i* force updates should have been performed atomically to avoid race conditions.

Verlet lists are a good example of the difference of clusters with shared memory architectures. Verlet lists have proved to be the best option for years in clusters and variations of this algorithm have been used from almost all the MD packages (Gromacs, NAMD etc.) for their cluster implementation. However, the CUDA architecture offers different capabilities and drawbacks to algorithm designers. Verlet lists on the GPGPU result in processing a big list of neighbours of variable length for each atom. The drawbacks of this fact are two-fold. First, although the CUDA architecture excels at processing large amounts of threads concurrently, it does not perform well when serially executing big loops of variable size within each thread as the variability of the size of the neighbour lists will cause warp divergence. The warp execution time will always be determined on the largest amount of neighbours that any single thread that belongs to the warp holds. The O(CN) global memory reads causes problems for L2 cache hit rates, despite the fact that Anderson *et.al* reported a good solution on this issue. Finally Newton's third law is challenging to implement on GPGPU architectures with capability up to 2.x due to the O(CN) global memory atomic operations that it would require.

4.3 Cell lists

These algorithms have the binning process in common with the Verlet list approaches discussed above. The macromolecule's domain is divided into $r_c{}^3$ cubes and atoms are binned into these cubes according to their co-ordinates. The advantage of this method over the Verlet lists is the ability to exploit the on-chip memory, which is cached. Home shell atom coordinates and parameters are loaded into registers and the 26 connected neighbouring shells atom coordinates and parameters are loaded into the shared memory, one bin at a time. Each register stored atom iterates through all of the atoms in each bin and performs energy and force calculations if their interatomic distance is less than the cut-off. All the reported implementations [38 - 40], chose not to use Newton's third law to avoid scattered writes and atomic operations, which in turn translated into faster overall performance.

Updating the cell list can be performed similarly to Verlet lists at fixed intervals, or by tracking atom movement and updating the cell list when an atom has deviated away from its initial

position by a distance δ as explained earlier. Van Meel [44] implemented this algorithm and performed these updates on the GPGPU using a double buffering method.

Stone et.al [46] implemented this algorithm varying the sizes of the bins to cut-off + δ where $1 \leq \delta \leq 3$ Å. This would vary the actual occupancy of the GPGPU as the size of the bins determines the number of atoms residing in them and the total number of blocks needed for the simulation. Varying the size of the bins above the cut-off distance results in more failed atom distance comparisons. However, higher occupancy levels lead to the ability to manage more warps concurrently, and this can improve performance. In the NAMD case there was a trade-off between higher occupancy and greater number of failed distance evaluations. The published results shown that for the 8 and 10 Angstrom cut-offs, one Angstrom larger bin sizes were degrading performance, but at 12 Å the results between the two approaches were similar. Overall, the GPGPU version outperformed the CPU one by a factor of four.

However, this scheme has some drawbacks. The first is that of load imbalance. As we can observe from Figure 4.1, the number of atoms resident in the cell varies. Due to the CUDA execution model, each computational kernel is launched with a fixed amount of warps *w* containing 32 threads per block [12]. This means that the atoms resident in a block will be less than the total amount of threads that each block executes. As a result, each block's final warp will have 0-31 threads without any work to do, which in turn may waste parallelism. Similarly, the number of warps issued for each block that have work to do will either be less than or equal to *w*, depending on the corresponding cell's atom concentration.

The second major problem is that of void computations. During each voxel pair processing, 86% of the distance calculations checks are performed for atom pairs further than r_c apart [47]. This creates an overhead in these algorithms, especially considering that most implementations have described them as computationally bound. Some applications tried to solve these problems by using finer grained regular grids using $r_c/2$ sided cells [45]. This approach improves the unnecessary distance checks issue, but introduces more global memory transactions. Global memory throughput is slower than simple floating point operations especially when memory access is not in order.

When this decomposition scheme is used for potential energy calculations it is more efficient to use a different traversal approach such as the half shell [48] to generate cell lists. This would guarantee that each cell-pair will only interact once during the simulation and potential energy values for the neighbour cell can be derived using Newton's third law without repeating the whole calculation. Regarding force calculations, using Newton's third law is not as efficient as it requires a vast amount of atomic operations. In order to exemplify this, consider the following code snippet:

```
1. float3 *Fh; //home atom forces held in register
2. float3 *Fnb; //neighbour shell forces held in shared memory
3. for j = 0 → neighbour_cell_number_of_atoms
4.     calculate force between home atom and j and add it to Fh;
5.     subtract Fh from Fnb[j]; //race conditions!
```

Assume a warp of threads 0-32 executing the above snippet in a SIMT (single instruction multiple threads) manner [49]. The subtraction instruction in line 5 will be executed by all 32 threads in parallel creating race conditions. One option would be the use of shared memory atomics, which would ensure accuracy. Our experiments show that this method under-performs by nearly 50% compared to the normal 26-connected neighbour shell traversal. Therefore we can argue that this is the third drawback of cell-list based algorithms regarding force calculations.

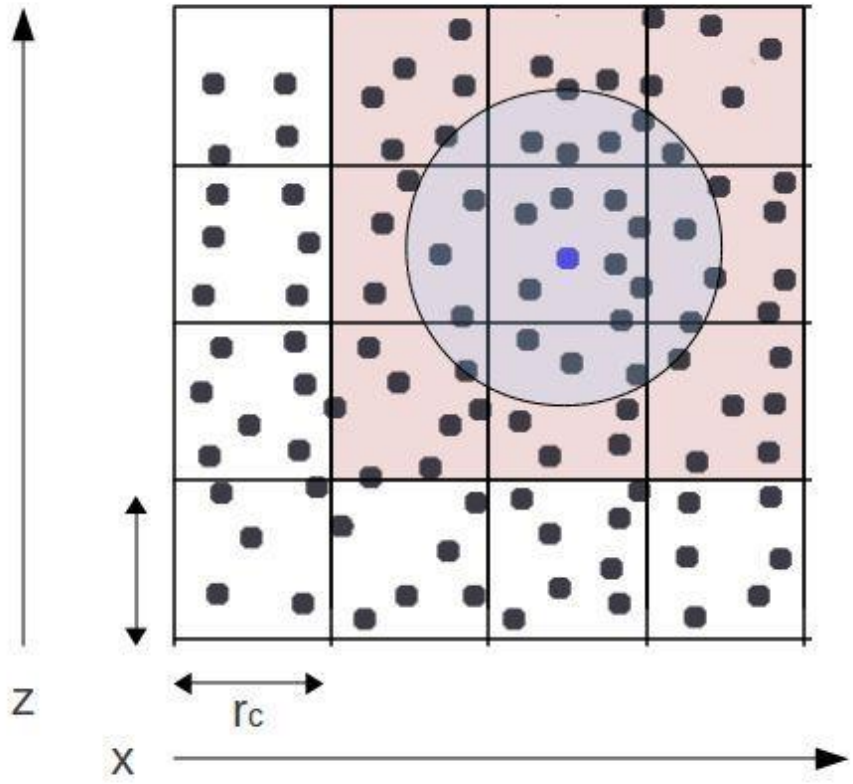


Figure 4-1: A regular grid decomposition scheme in 2D. Load imbalance and atoms beyond cut-off range are highlighted. Atoms in the light cyan circle are within the cut-off range of the blue home atom. Particles in the light red region belong to the 26 connected (8-connected in 2D) cells, but their distance from the blue home atom is greater than r_c.

Pal and Hess [50] designed an approach that uses aspects of both cell and Verlet lists. In particular, they designed a neighbour search method using a fine grained spatial decomposition approach. Atoms were binned in a rectangular grid of fixed base size and variable height (z-axis). The neighbour list's work unit would be a group of eight atoms instead of a single one. This would help minimizing the effects of cache misses when loading atoms and updating forces as the pair interaction would be on groups of eight atoms rather than single atom pairs. Computational kernels would be called in groups of eight 8-grouped work units. This approach aims to minimize the void computations effect evident in cell lists and also improve the cache hits on global memory with the aforementioned 8-atom grouping.

However, it still requires a vast amount of out-of-order memory transactions, including atomic memory writes to global memory. Finally, the authors would execute their computational kernels in groups of eight 8-atom groups, resulting in 64-thread blocks execution, which in turn results into very low multiprocessor occupancy on the GPU.

4.4 Other notable implementations (workgroup lists)

Both cell list and Verlet list algorithm suitability on GPGPUs has been questioned in the past, for reasons that we addressed in this chapter. An implementation designed explicitly to run in CUDA was created in OpenMM [51] .The simulation system was divided in workgroups of 32 atoms, in order to match CUDA's parallel execution unit (warp). A neighbour list of all warps was then created and interacting workgroup pair lists would perform the non-bonded force and energy calculations. The criteria that would flag two workgroups as neighbours would be that the minimum distance between their bounding boxes would be less than the cut-off. This ensures that workgroup pairs that had no interactions are pruned reducing the run time of the algorithm.

This algorithm effectively eliminates the problems associated with warp divergence that were evident in the previous two implementations. In addition, it achieves high levels of L2 cache hits, as decomposition takes place sequentially along the polymer chain. Also, this method takes advantage of Newton's third law on force calculation, at the expense of reducing the forces out of an $(N/32)^2$ matrix

Although the problem of void pairwise distance calculations still persists, it is difficult to understand which approach suffers from it the most. A drawback of this approach is that it needs to be implemented by kernels using 32 threads which again leads to low occupancy levels. In addition, the decomposition scheme is more fine-grained and hence will result in more global memory reads/writes than can be performed in order. The performance of this algorithm was tested against serial codes of existing MD packages. It is reported to perform 28 times faster than that of the serial NAMD, whereas the CUDA implementation of NAMD was only 4-5x faster than its serial one [51]. However these comparisons are subject to bias stemming from benchmarks made in different hardware systems and biological targets.

4.5 Bonded Forces

Bonded forces are a much more simplified potential to compute as the bonds between atoms are known and do not change throughout the simulation. The most common way of calculating this potential in parallel on the GPGPU is through an atom decomposition where a thread is gathering the force for each atom. Xu *et.al* [52] implemented this algorithm with three separate kernels dealing with bond, angle and dihedral potentials. The above algorithm is not very memory efficient as atom indexes and their coordinates will be loaded from global memory multiple times to calculate each potential. Friedrichs *et.al* [53], implemented the bonded forces in a different way in order to minimize global memory reads and writes. The simulation domain was decomposed into quads of atoms forming unique dihedral potentials. Within those quads, all the unique bonded and angle interactions would be computed. Duplicate bond and angle entry parameters would be set to zero in order to ensure that bonded and angle interactions would only be calculated once. Although this algorithm is much more memory efficient, it does increase the number of math operations needed accounting for the ones performing void calculations. It does not entirely solve the problem of multiple atom coordinates out of order reads and force writes from and to global memory. Also, around a single bonded atom pair there can be up to eight different unique dihedral potentials, which means that the same bonded pair will be loaded 8 times from global memory and its forces will be also updated eight-fold.

4.6 1-4 bonded Interactions Handling

One of the most expensive operations in free energy and force calculations is that of exclusions handling. In non CUDA architectures, that could be dealt with accessing an N^2 boolean array checking if atom pair *i-j* is included or excluded in the calculations. Unfortunately, in CUDA, that would hinder performance by a large factor due to the out of order memory reads required.

Friedrichs [53], reduced the size of the N^2matrix to $N^2/4$ by including the exclusion flags of 4 consecutively indexed atoms in each array integer. However that was implemented in an O(N^2) implementation (no Verlet or cell lists) and therefore the exclusions array was accessed

in order and resulted in maximum L2 cache hits.

NAMD [46] adopts an interesting approach to exclusions handling based on the observation that in a .pdb or .mol2 file bonded atoms are sequentially indexed. Therefore there is a maximum index range *r* where two atoms are 1-4 bonded. Accessing exclusion arrays should be performed only for atoms whose index difference lies within this range. Constructing exclusion lists for each atom, helps checking the atoms excluded for every atom pairs whose index range is below the maximum. A further observation made was that for large arrays, the number of unique exclusion arrays would be much less than *N* and restricted to 700-800. Therefore unique exclusion arrays could fit in the GPU's constant array and accessed from there. However, constant memory accesses perform well only when the same memory address is accessed on the same instruction by all threads in a warp (broadcast). In this case though there is no guarantee that all atom indexes in a warp can request the same exclusion sequence, therefore it is doubtful how much performance gain can be realised from storing the exclusion list into the constant memory, especially for smaller system sizes.

4.7 Integration of molecular mechanics computations in a simulation

The objective of finding stable (or preferred) conformations for the system requires locating those conformations that lie on minimum points on the energy surface [54]. However, this is a complex multi-dimensional problem. For a system with *N* atoms, its potential energy is a function of 3*N* Cartesian co-ordinates. Therefore, the molecular energy surface is rather chaotic and searching for the global minimum cannot always be feasible. Failing that, exploring the potential energy surface is the way to find a set of optimal solutions (local minimums) where the best solution can be chosen.

Force-field computations need to run in a continuous loop in order to give simulation results and continuous conformational updates. For the case of molecular dynamics, conformations are updated using Newton's laws of motion on all atoms of the system concurrently using a small fixed time-step to generate new atom positions and velocities. In the case of flexible docking, different poses of the ligand are generated against the protein's binding site and the software evaluates fit considering the poses that generate the lowest delta energy value. This

process is termed “energy minimization”.

Molecular conformations are updated using the appropriate algorithm, depending on the degree of flexibility that the application offers. Zodiac had previously been using the steepest descent approach in order to provide conformational flexibility on the ligand. Steepest descents, is a simple and easy way to find a local minimum and its main characteristic is that it only takes downhill moves on a potential energy surface (PES). However, choosing the first encountered local minimum is not the only way to optimize the force-field function. Exploring the PES can sometimes yield better results to the one initially found. The search for optimal solutions can be facilitated with the use of meta-heuristics [55].

4.8 Exploring energy landscapes

Meta-heuristic methods cover a wide spectrum of algorithmic techniques. Evolutionary algorithms (EA) and genetic algorithms (GA) are methods able to iteratively generate a set of solutions. Evolutionary algorithms are amongst the most popular meta-heuristics and attempt to solve the optimization problem by generating a population of possible solutions from a parent (mutation). From the generated population the fittest is chosen to generate the new pool of solutions and so forth [56]. Genetic algorithms adopt a similar approach, with the addition of the concept of crossover between two solutions to create the new generation [57]. These solutions generations will be gradually improving with respect to the strategy adopted to move across the energy landscape.

The exploration of potential energy surfaces requires the adoption of uphill moves in order to recover from local minima. Such strategies for uphill move selection include simulated annealing (SA), Monte-Carlo (MC), iterated local search (ILS), Guided local search (GLS) amongst others. Simulated annealing uses a temperature variable to generate a probability function that aids into accepting solutions that go uphill on the energy surface. This helps the system avoid being trapped at local minima and be able to also explore other minima [58,59]. Monte Carlo methods adopt a similar approach of escaping local minima by accepting uphill moves based on a probability function such as the Metropolis criterion [60, 61]. ILS reaches optimal solutions by iteratively performing local search[62] to reach a local minimum[63-65].

The way it escapes a local minimum is by perturbing the discovered solution in hope that the perturbation mechanism will bring the system into a well with a lower minimum than the parent one. GLS, escapes from local minima, by gradually increasing the global minimum energy value. This way, the system will escape from a well if it is trapped, by taking an uphill move close to the minimum, which will result into a new landscape with the hope that a better minimum solution will be discovered [66-68].

Various meta-heuristics have been used by many recent molecular docking applications. Autodock-Vina [69] uses a hybrid ILS strategy using the Metropolis criterion to accept optimal solutions and predict binding affinity of protein-ligand systems [69]. Variations of the genetic algorithm have been implemented in various published works [70,71] to predict molecular structures. Soares [72] used an evolutionary strategy to solve this problem and CAI and Shao [73] used a hybrid method of EA combined with SA for structure prediction. Paradiseo [74] approaches the docking problem with a parallel GA in order to leverage additional processing power and improve its performance and efficiency.

4.9 Summary

This chapter reviews the state-of-the art implementations in areas of molecular mechanics and mathematical optimization. It also provides the necessary background in order to judge the level of novelty of the contributions claimed in this thesis and listed in sections 1.8 and 9.2. In particular we described a number of HPLD applications focusing on the approach they follow regarding protein flexibility. Then we described the most notable implementations in force-field calculations, with an emphasis in non-bonded interactions. Finally, we described integration methods for providing conformation sampling, highlighting the use of meta-heuristic approaches and their contribution in conformational sampling.

Initial Experiments

In this chapter we describe some initial unsuccessful implementations. These methodologies' performance was not adequate in order to meet the two objectives set in Chapter 1. However, they provided useful feedback for the final implementation which is listed in detail in Chapters 7 and 8. This chapter does not provide any of the contributions of this thesis, hence the description of the methodologies used as well as the results produced are not discussed in detail.

5.1 Neighbour list experiments

The very first implementation attempted to solve the non-bonded interactions problem included the use of Verlet lists. The implementation was similar to those listed in Section 4.2.2 in its most simple form (ie. no Hilbert curve spatial sorting). The list was created after binning the atoms in a regular grid made of cubes of $r_c{}^3$ dimensions. This implementation was only intended to provide us with an indication on how this algorithm performed in CUDA and for our specific force-field, which requires a greater number of FLOPS compared to other force-fields as explained in section 3.5. The results listed in Figure 5-1 were only able to show that this methodology could not suit the simulation's requirements. Although the algorithm scaled linearly with system size, the results would not be able to meet the 33ms limit. The potential energy kernels would take too long to execute even for very small proteins and a single update step would require at least a force kernel calculation in addition. This would result in the total running time exceeding the 33ms limit. This experiment ran on an Nvidia Tesla 2070.

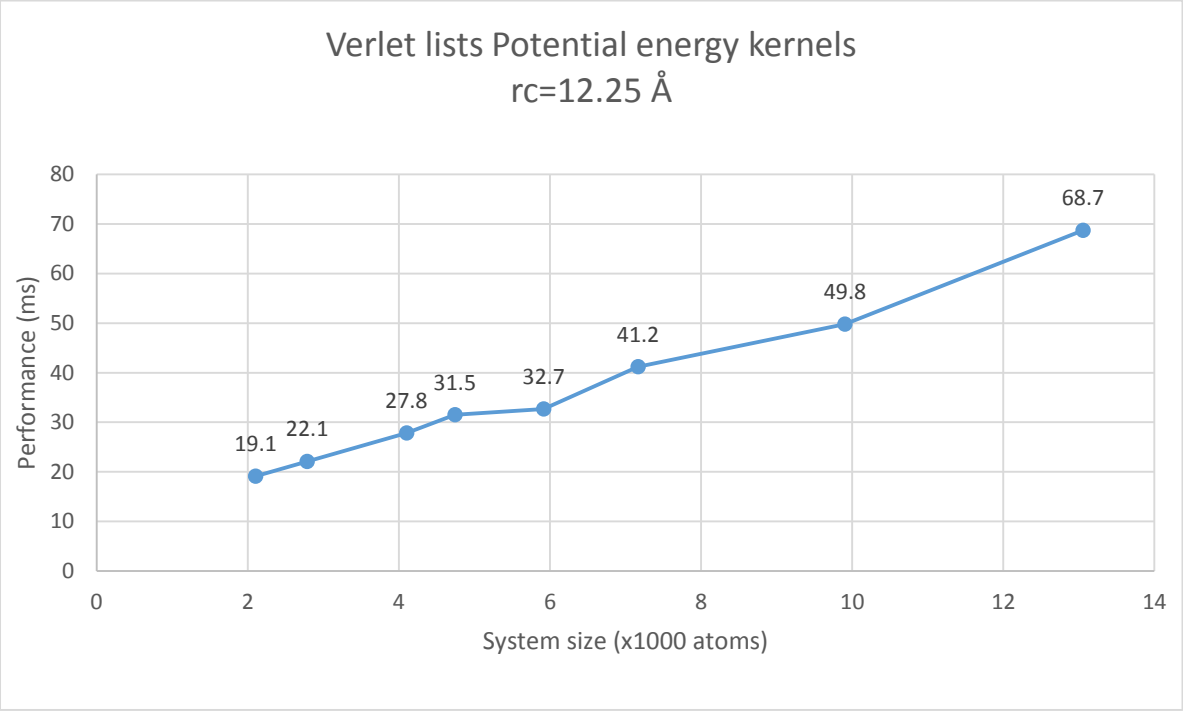
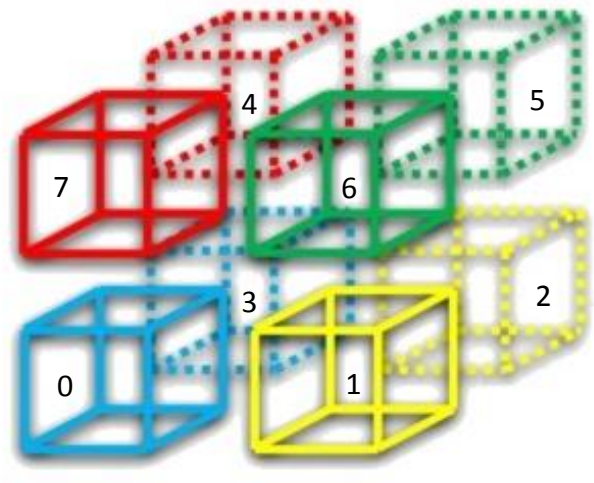


Figure 5-1: A simple neighbour list implementation for potential energy calculations.

5.2 Cell lists

The next attempt was that of a cell list approach similar to that explained in section 4.3 without any attempts to optimize the solution. The results were much better compared to that of neighbour lists for potential energy kernels. Although the results were not sufficient to meet the first objective, the methodology showed good potential. The comparison results between Verlet and cell lists were consistent to already published research supporting that the former are not suitable for GPGPU implementations due to the amount of random memory access generated on the computation kernels[51,75]. Therefore, the decision at the time was to research new optimization techniques in order to improve performance of the simple cell list implementation.

The initial optimization ideas focused on improving the global memory usage and to develop methodologies able to accommodate Newton's third law on force calculations. Regarding the former the decision was to implement an algorithm for the cell list generation based on the 8^{th} cell concept [48] , where cells (atom co-ordinates and parameters) would be loaded as groups of eight as demonstrated in Figure 5-2.



Cell	0	1	2	3	4	5	6	7
0	x	x	x	x	x	x	x	x
1	-	-	-	x	x	x	-	-
2	-	-	-	-	-	x	-	-
3	-	-	-	-	-	x	x	-

Figure 5-2: Left, a group of eight adjacent cells, two in each axis, numbered from 0-7. Right the required cell-pair interactions flagged with an (x). Dash (-) denotes no interaction.

The 3D regular grid needs to be traversed such, that each block will become cell 0 once during traversal and its adjacent cells will be loaded matching the structure illustrated in Figure 5-2. For each loaded 8-cell structure, the required cell pair processing as demonstrated in Figure 5-2 on the right, needs to be performed.

The above cell pair interactions achieve the same result as that of the half-cell traversal approach[48]. The benefit of this approach is the fact that instead of importing thirteen cells from global memory, only eight are needed at each time to perform thirteen cell pair interactions. This results in more data re-use and less global memory transactions in total for both energy and force kernels. However, this approach introduces a new problem, as more on-chip resources are needed at each time-step to accommodate eight cells.

The initial results were based on a code where all 8 cells were loaded in shared memory, but

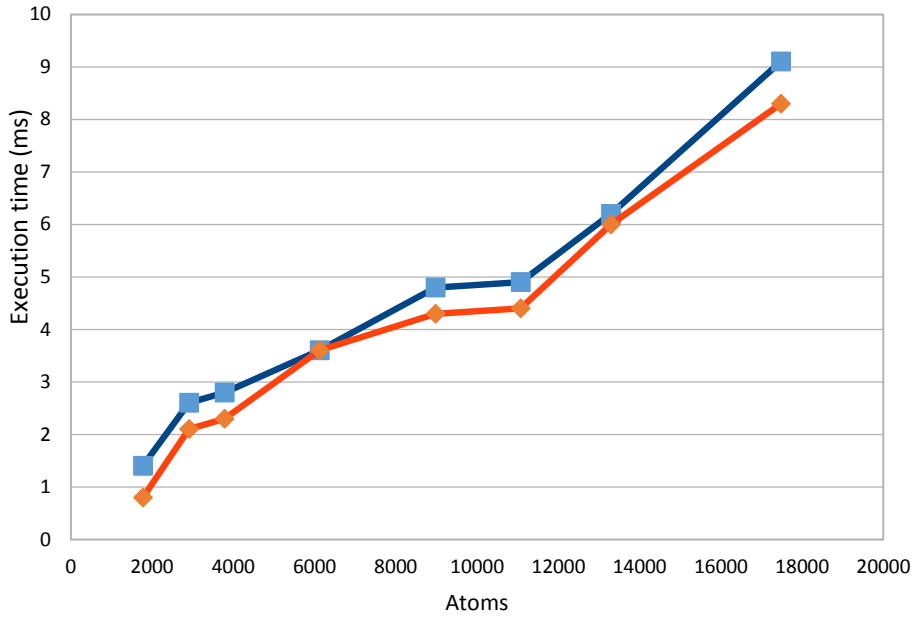
this was under performing as it was suppressing occupancy levels to a minimum and it was obvious that a more on-chip memory efficient method was needed. The final version would have four cells in flight at a time, two of those in registers and the other two in shared memory. This way, all transactions could be processed between the eight cells in a rather complicated algorithm requiring frequent variable switching between global memory, shared memory and registers.

The second issue to tackle in these attempts, was that of Newton's third law. The first idea was that shared memory atomics could avoid the race conditions explained in Section 4.3 during the force computation kernels. This was attempted on Fermi generation cards for both 8^{th} cell and half-cell approaches. The second idea was that of using a shared memory buffer to store the force matrix between groups of 32x32 home versus neighbour groups followed by a reduction, attempted on the half shell approach only.

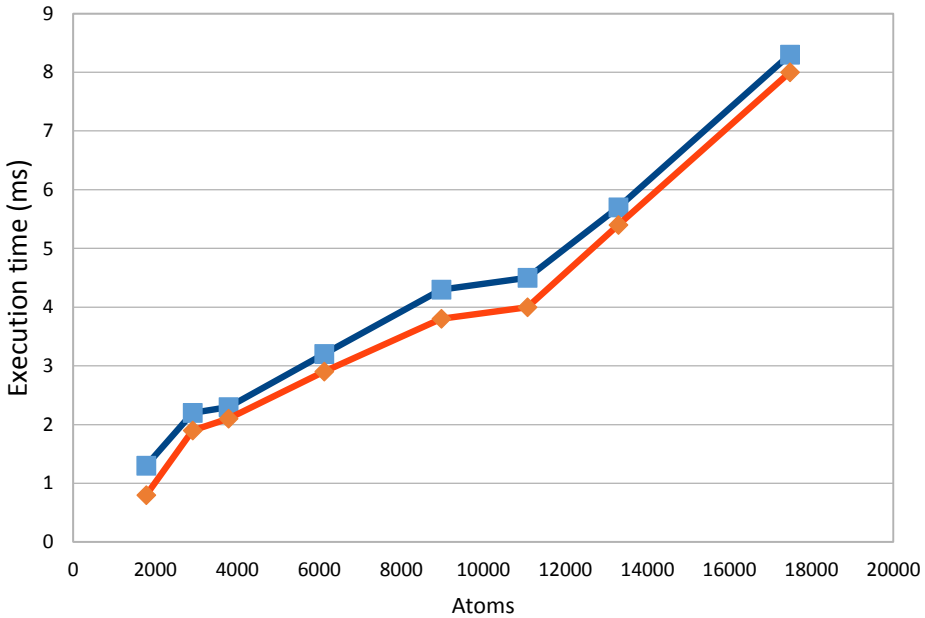
5.3 Results and discussion

The first surprise came in the energy kernels evaluation between the half cell and 8^{th} cell approaches. The two methods were using the same code to calculate the non-bonded energy between cell-pairs. The only difference was the grid traversal and on-chip memory sources between the two approaches. The half-cell method, although less global memory efficient, was outperforming the 8^{th} cell by a small factor of 5-10%.

Energy kernels benchmarks (a)
Cutoff = 12.25 Å



Energy kernels benchmarks (b)
Cutoff = 10.75 Å



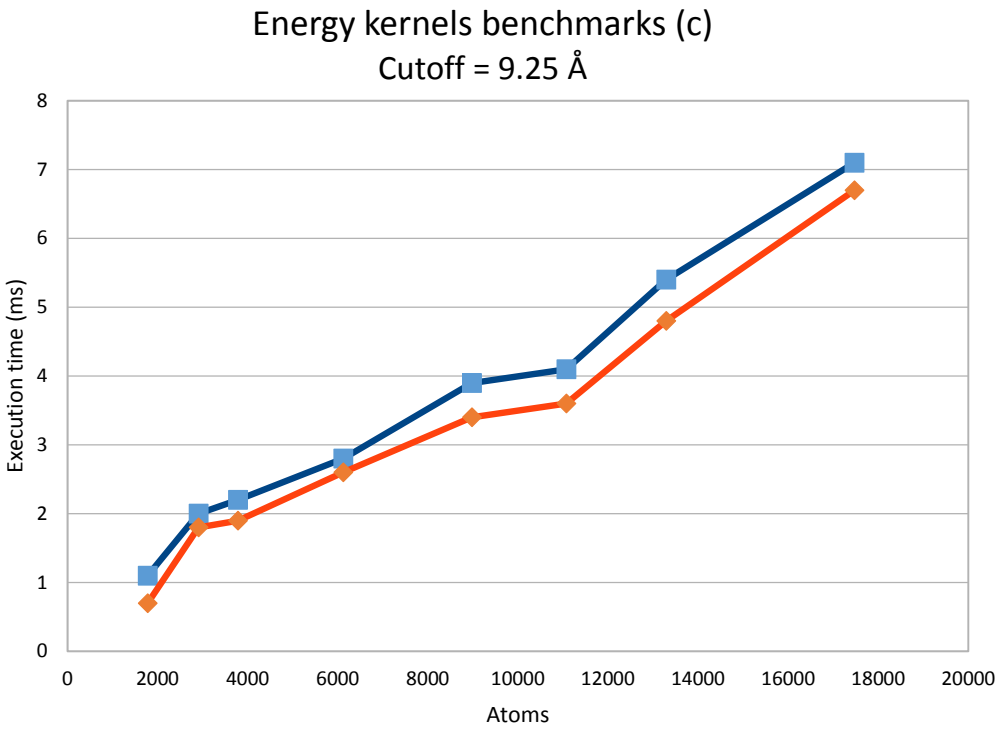


Figure 5-3: From a to c (12.25 – 9.25 Å cut-off distances) Benchmarks for the energy kernels between the 8th cell (blue line) and the half cell (13 neighbours plus home cell, orange line) approaches.

algorithm	Occupancy	Registers/Thread	Shared Memory/Block
8^{th} shell	29.2%	50	14 kB
27 Shell	43.8%	40	8 kB

Table 5-1: Occupancy comparison for the two approaches:

The conclusion from the above benchmarks is that multiprocessor occupancy is vital to our problem. The reason that this was not considered before designing this approach is that previous research focused on the performance benefits of occupancy [76]. The overall conclusion was that occupancy would always benefit memory bound kernels and would sometimes benefit compute bound problems too. Our problem is compute bound, and it is obvious that multiprocessor occupancy is also optimizing it by a factor higher than that shown

in Figure 5-3. This is because it also outweighs the performance gain from fewer global memory fetches and fewer thread synchronisations for the 8th cell algorithm.

In Figure 5-4 we can see the results of eighth cell shared memory atomics for force calculations (8S1_g) versus the half cell and 27-cell approaches (27S_g, HS_g curves). Regarding the comparison between the half cell and the 26-neighbour approaches (where no atomics needed as Newton's third law is not applied), the performance is similar. This means that the cost of introducing shared memory atomics outweighs the amount of FLOPS saved for the kernel. The 8th cell approach still underperforms as the on-chip memory resources for the gradients are greater, due to the forces storage. Also the approach would require an amount of global memory atomics as well (slow on Fermi devices!). This amount could be avoided with the introduction of eight buffers to store unique force values for cells 1-8 followed by a reduction in the end. This technique was implemented, but performance benchmarks following implementation showed that there was no performance gain for doing so. Both buffered and global memory atomics implementations were producing very similar results, meaning that the overhead of using global memory atomics in our case was similar to the one launching an additional kernel for the reduction of the eight buffers. Finally, the attempt to decompose the cell-pair interactions into 32 atom work units and store the forces into a 32x32 matrix in shared memory, followed by a reduction, has shown a similar performance to the 27S_g and HS_g curves. The data were not recorded as at that time the project focus had already switched to a much more efficient implementation.

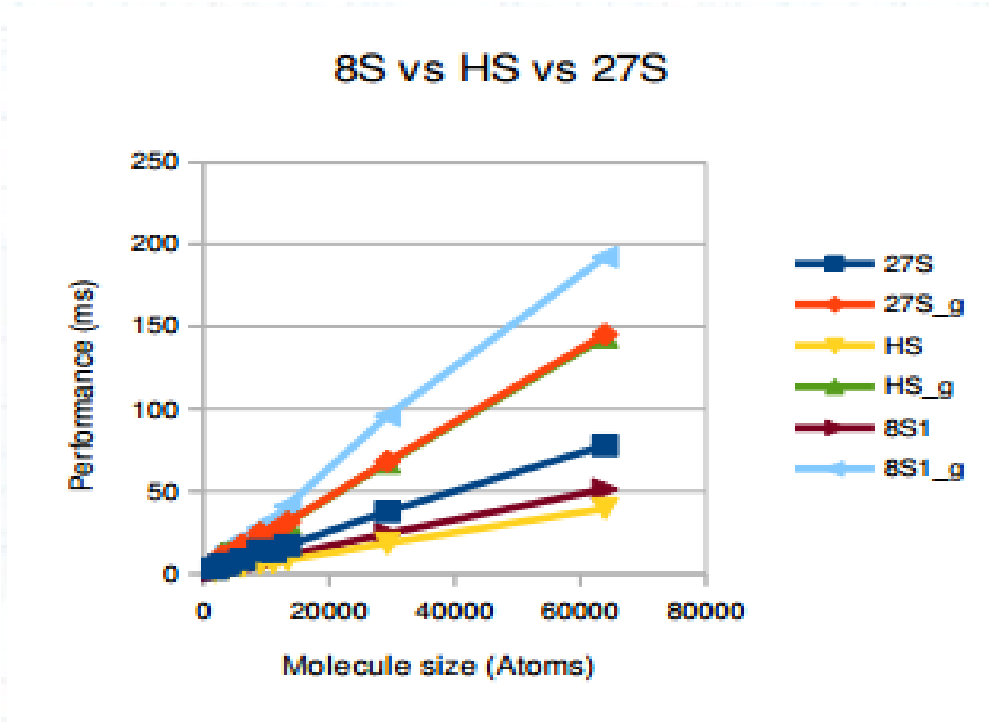


Figure 5-4: An overall benchmark for the 8th cell (8S1) half cell (HS) and 26-neighbour (27S) approaches. In the legend the _g extension stands for gradients (forces) and the rest of the curves represent potential energy computations. (Figure recovered from a 2011 presentation as an image, hence the low resolution. Data not available to reproduce!)

From this chapter we can conclude that there is no point trying to improve the memory efficiency of a compute bound kernel. Instead, an approach to reduce unnecessary FLOPs such as distance checks for atoms beyond cut-off and an efficient way to implement Newton's third law with no extra memory overheads would be more beneficial. In addition, we should seek a method that makes the most of the parallelism available by CUDA and also try to maximise occupancy to match Nvidia's recommended levels of 50% [13].

5.4 Summary

This chapter highlights research conducted between terms 2 - 3 of the 6 term period of this Ph.D. project. This research did not provide any contributions to the body of literature, but provided the foundations for future implementations. Firstly, it proved that in our case, multiprocessor occupancy benefits the performance of our kernels, although we are dealing with a compute bound problem. Secondly, it highlighted the importance of optimizing this

code by minimizing the amount of FLOPS performed. That means, ideally we want the minimum amount of pair distance computations of atoms beyond the cut-off distance. Thirdly, we should aim to avoid computing the same atom pair twice on the gradients computation by finding a memory efficient way to implement Newton's third law.

As a final concluding remark we need to highlight that the above experiments were performed on the Fermi line of NVIDIA GPUs. In regards to a remark made for global memory atomics, we need to mention that for the Keppler architectures and onwards, the performance of global memory atomics is comparable to non-atomic operations. Hence we relied on this feature on newer versions of our algorithm listed in the following chapter.

Chapter 6

CUDA-Parallel Algorithms for Molecular Mechanics Computations

In this chapter, an improved cell-list approach designed to match the Kepler architecture of General-Purpose Graphics Processing Units (GPGPU) is described. An explanation of how this approach improves load balancing for the above algorithm and how warp intrinsics are used to implement Newton's third law for the non-bonded force calculations is provided. We also describe our approach to handling exclusions together with a method for calculating bonded forces and 1-4 electrostatic scaling using a single CUDA kernel. Performance benchmarks are included in the last section to show the linear scaling of this implementation using a step minimization method (greedy algorithm). In addition, multiple performance benchmarks demonstrate the contribution of various optimizations used for the described implementations. The methodologies and results listed in this chapter have been published in the Journal of Computational Chemistry [77].

6.1 Algorithm implementation

Our initial designs were based on regular grid spatial decomposition schemes with $r_c{}^3$ cell dimensions as described in the previous chapter. We decided to redesign and improve the above algorithm to suit the Kepler architecture, aiming to address the three major drawbacks we referred to previously in Section 4.3, namely load imbalance, void distance checks and lack of Newton's third law.

Our first observation was that we could enhance the granularity of the regular grid by logically

sub-dividing a cell's space (super-cell) into 8 sub-cells of $(r_c/2)^3$. This means that cell pair processing can now be decomposed into 8×8 = 64 sub-cell pair processes between an interacting super-cell pair. Sub-cells whose bounding boxes are further than r_c apart, will be excluded from processing (see *Figure 6-1: Spatial decomposition in 2D. Cells are logically divided into 4 $r_c^2/4$ sub-cells. Purple shaded sub-cells are further than r_c apart from the blue home particle*) resulting in fewer distance checks per super-cell pair. This way we could combine the benefits of both finer and coarser grained decomposition schemes, keeping the memory usage efficiency of the former and the fewer distance checks needed for the latter.

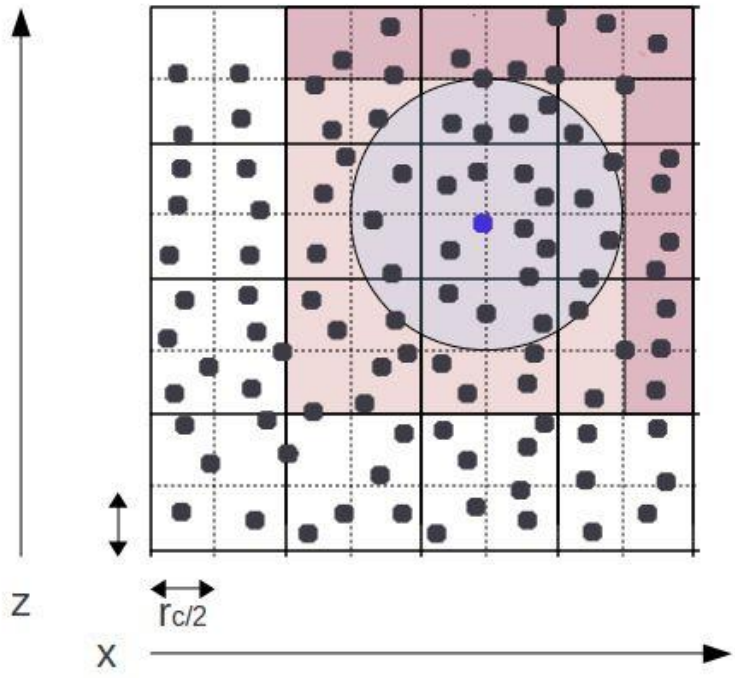


Figure 6-1: Spatial decomposition in 2D. Cells are logically divided into 4 $r_c^2/4$ sub-cells. Purple shaded sub-cells are further than r_c apart from the blue home particle

The execution grid for the above decomposition scheme consists of 256-thread blocks for both energy and force computations for two main reasons. Firstly, the logical subdivision of super-cells could be performed such that we could guarantee a number of atoms not to exceed the size of the warp (32) for each sub-cell (warp-group). This will enable us to use warp

intrinsic functions on a warp-group pair interaction, which, under this decomposition scheme, can be processed by a single warp of 32 threads. This will result in faster exchange of values between registers belonging to different threads without the need of using shared or global memory resources.

Secondly, considering that our computational kernels require 9KB of shared memory size and 60 registers per thread, 256 thread-blocks will result in 50% theoretical multiprocessor occupancy levels. These occupancy levels are sufficient enough to take advantage of the latency hiding optimizations available to GPGPUs [12, 13].

6.1.1 3D irregular grid decomposition

Considering the above, we would now aim to design a decomposition scheme that would produce load balanced cells of 256 atoms, which can be logically divided into 8 warp-groups. It makes sense that the smaller the bounding box of the warp-groups, the less distance checks we will have to perform during force and energy computations. Hence, we would aim for warp-groups to be as compact as possible and have non-overlapping bounds (ie. Their bounding boxes should not overlap).

This could be done by mapping a molecular system's atoms into a three dimensional rectangular grid using cells with base dimensions of r_c^2 and variable height similar to the approach that Pal and Hess implemented in a more fine-grained grid manner [50]. The atoms of a bio-molecular system would be geometrically mapped in two dimensions first, according to their x and y co-ordinates, forming a stack of atoms belonging to each x-y rectangular tile (XY stack). Then for each of these stacks, atoms could be sorted on their z-coordinates and divided into blocks of 256 atoms (cells). In this way the spatial relationships between cells on the x and y directions would be maintained and load balanced shells of variable height can be created.

However, the above approach introduces three new problems. Firstly and most importantly, the the smaller the cut-off distance used the more elongated the cells (and hence the warp-groups)

groups) in the irregular grid would be. This could in turn affect performance by lowering the amount of interacting atoms per warp-group pair and hence introduce more global memory reads. The second problem would be that the last cell on the z direction for each XY stack would would not necessarily be load balanced. The third problem is represented by the poor parallelism caused by incidences of distant cell pairs, as shown in

Figure 6-2c. Although cell atom pairs are formed from adjacent XY stacks, there are no adjacency criteria on the z axis and distant cell pairs can be formed with only a few interacting atoms between them. One of the major side-effects of this scenario is that there will be an amount of global memory fetches of warp-groups with no interactions at all. Another deficiency of this scenario is the fact that a warp of 32 threads is going to be under-utilized to serve very few atom interactions.

In order to avoid the aforementioned problems that a rectangular grid approach would introduce to our code, we implemented the following approach leading to a more load balanced irregular grid decomposition.

In order to address the first problem, proteins were rotated in space such that their orientation would be perpendicular to the XY plane at the beginning of the simulation. This way it would be expected that the top face of the simulating system's bounding rectangle will be its smallest and hence will decrease the amount of imbalanced top cells. The problem would still be evident, but the candidate top cells number would drop and the number of load balanced cells would potentially increase depending on the elongatedness of the protein shape. This technique would not always help, though as we can see in Table 6-1. This means that finding the system's orientation that minimizes its stack count is a more complex optimization problem and its optimal solution cannot be located just by rotating the system such that its principal axis is perpendicular to the XY plane. Certainly a better solution to this optimization problem could be found with a suitable minimization method. We decided not to investigate this issue any further, as the performance benefits expected from a better optimization method would be too small for the time estimated to spend on it. In our code, if the perpendicular principal axis solution produces less stacks, we accept the transformation,

else we reject it.

	Not Rotated		Rotated	
	x, y, z	stacks count	x, y, z	stacks count
1UGM	(4, 3, 6)	9	(4,3,8)	9
3RDD	(4, 3, 6)	10	(4,4,8)	11
3F9E	(5, 5,10)	17	(5,4,12)	14
3FAU	(5, 4,12)	12	(5,4,12)	13
20O0	(5, 5,10)	17	(5, 5, 10)	17
3O05	(6, 6,16)	24	(6, 6, 16)	24
2EAR	(7, 6,20)	26	(5, 6, 22)	21
1AFR	(9, 9,20)	55	(8, 9,20)	42

Table 6-1: Comparison of total XY stacks between transformed and not transformed systems. Transformation was such that the protein's principal axis would be perpendicular to the XY plane.

In order to address the second problem, that of elongated cells, we introduced a skin value δ for the bins base, such that $r_b = r_c + \delta$, where r_b is the base side's length (Figure 6.2a). This technique allows the adjustment of the shape of the cells and warp-groups, in the hope that through investigation the skin value range that produces the highest performance would be found.

In addition, a migration method can improve the first problem (imbalanced top cells) even further. After binning the system's atoms in 2D, a method is used to traverse XY stacks aiming to make their atom count divisible by 256. This can be done using a two pass algorithm as follows. Atoms in each XY stack are ordered according to their x co-ordinates. Then each stack is visited and if its atom count is not divisible by 256, half of the excess atoms are donated to the next XY stack along the x-axis direction. The second pass is a similar step on the y direction.

Atoms on each XY stack are sorted according to their y co-ordinates. Stacks are traversed again and if they have any excess atoms they donate them all to the next stack on the y direction. The two-pass method is used to encourage atom migration in both x and y directions and preserve the cells aspect ratio as much as possible so that's why the first pass moves only half of the excess atoms. At the end of this process, most of the top super-cells on the XY stacks will be load balanced and the resulting structure will be a 3-dimensional irregular grid as shown in Figure 6.2b.

An example of how the migration method assists in reducing the amount of top-level load-imbalanced cells is illustrated in Figure 6.2d, where we can see a reduction of imbalanced cells from 30 to 6. The only problem on this technique is the fact that the outer most stack row (or column depending on whether the x or y axis is processed first) will be load imbalanced. During the design phase of this method, we tried reducing this problem by adding a step migrating atoms on the affected row, resulting in reducing the amount of imbalanced top cells to 1. However, this was not translated into a performance gain and in some cases it even hindered performance so this optimization was cancelled.

Atom migration can take place only as long as the donating super-cell's affected XY side will remain greater or equal to cut-off after migration ($r_b \geq r_c$). This will ensure that cell pairs on the list are going to be formed from adjacent XY stacks and avoid generating cell-pairs with only few atom pair interactions. This technique is effective for large systems (>8,000 atoms) when there is a large enough skin value (ie. $\delta \geq 1\text{Å}$). Smaller systems have fewer atoms in stacks and migrating atoms cannot happen safely without violating the $r_b \geq r_c$ rule.

The reason why this method migrates atoms on a 4-connected stack region and not an 8-connected one, is purely down to performance constraints. The 8-connected option would require more steps, hence more migrating and sorting phases and that would add extra computational overhead on an already rather expensive binning method.

The next step would be to logically sub-divide the resulting cells into 8 load balanced sub-cells (or warp-groups). This is accomplished, by bisecting cells in z, y and x direction. Z-bisection would give us two 3D rectangular segments, each further divided in the y direction resulting

into four segments. The resulting four segments would be bisected in the x direction giving us the 8 warp-groups. Bisection was implemented by sorting particles according to their x, y and z co-ordinates. The final pseudocode of the resulting binning technique for the aforementioned (adjustable) irregular grid decomposition is listed below.

Routine: bin_irregular_grid

```
1.  Rotate system principal axis - perpendicular to XY plane
2.  Bin atoms in 2D (XY axis) forming a structure of XY stacks
3.  Sort all atoms on their x co-ordinates
4.  Traverse each cell and donate half of its excess atoms to
the next adjacent stack on the XY grid on the x direction.
5.  Sort all atoms on their y co-ordinates
6.  Traverse each cell and donate all of its excess atoms to
the next adjacent stack on the XY grid on the y direction.
7.  Sort each XY stack of atoms according to their z-co-ordinate
8.  For each 256 atom block on the resulting irregular grid:
9.   Construct 8 groups of 32 atoms (no overlap between them).
10.  Bisect in Z direction in 2 × 128 atom compartments
11.  Bisect each 128 atom compartment in 2 × 64 atom
compartments in Y direction
12.  Bisect each resulting 64 atom compartment in 2 × 32 atom
compartments in X direction
```

Where `count` is the atom count of each XY stack and `excess_atoms = count % 256`.

The third problem mentioned above is non-trivial and the best work-around it was to filter out (not fetch from memory) warp-groups of neighbour shells with no interactions during the computational kernel phase. This saves the overhead of the required global memory fetches for these atom-groups. Performance results showing the benefit of the techniques explained so far are listed in Section 6.5.

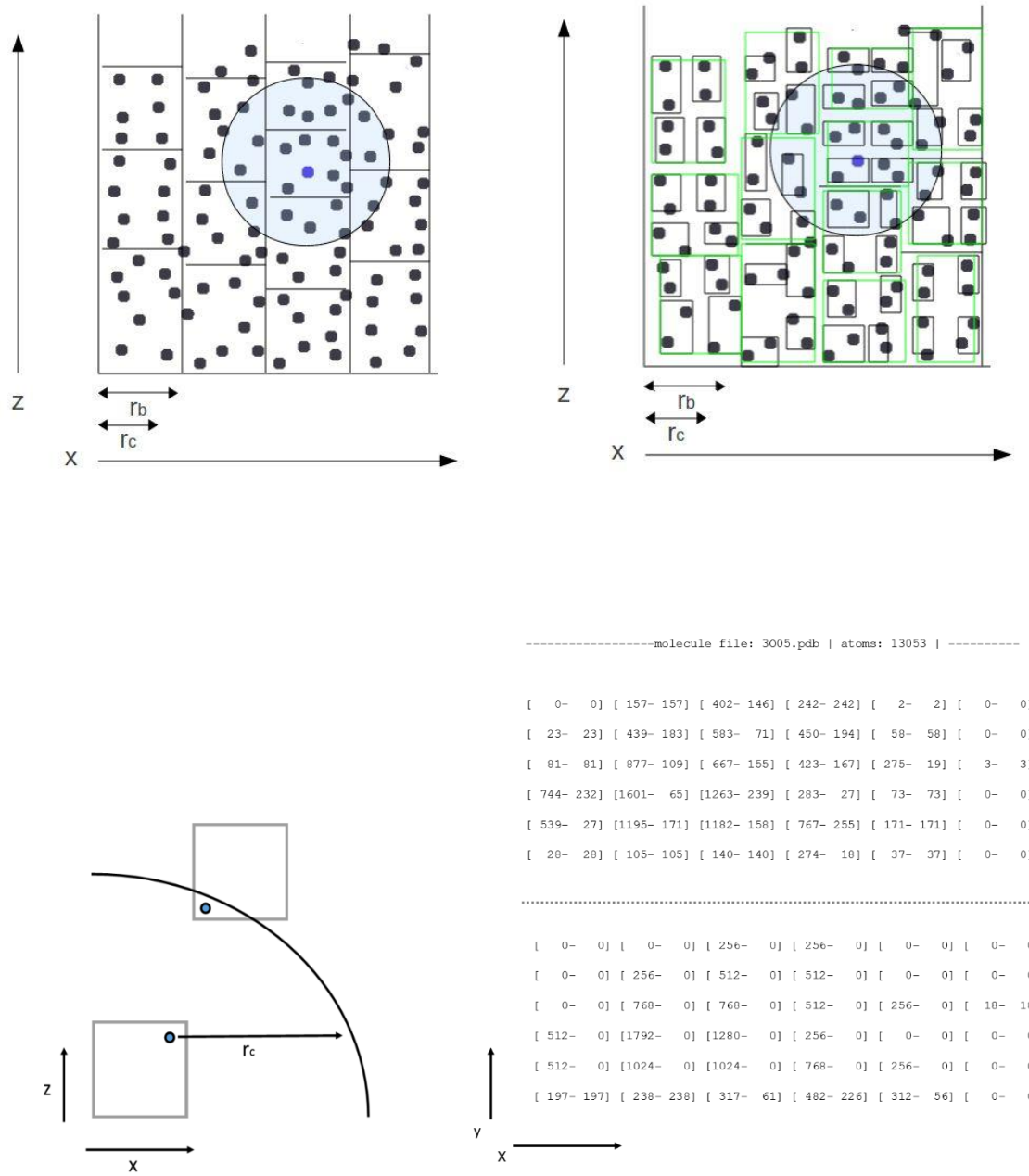


Figure 6-2A: 2D rectangular grid decomposition. Base width is still equal to r_b and height is adjusted to achieve load balance. All cells apart from the last one on the z direction hold the same number of atoms. B: Irregular grid decomposition after atom migration. Super-cells in green borders constitute of four warp-groups (eight in 3D). Most of the top cells of the stack

are now load-balanced. C: A distant cell pair with only one atom pair within a cut-off distance. D: An overview of the XY stacks before and after the atom migration process. The numbers in brackets represent the amount of atoms contained in each stack and its top cell [stack – top]. Above the dotted line is the structure before migration and below the dotted line is the one after. Before migration there were 30 top cells imbalanced versus 6 after the atom migration process.

6.1.2 Cell lists and warp-group interaction bitmaps

After the binning process, an interaction map is formed for the interacting super-cell pairs using the half shell [48] traversal approach. It is stored in the form of an array holding the index of cells interacting with the home cell. In addition, for each interacting cell pair a 64 – bit bitmap (using two 32-bit unsigned integers) is created reflecting the 64 warp-group interactions that take place for each interacting cell pair. Each warp-group pair's bounding box distance is calculated and if it is beyond r_c, then it is guaranteed that there are no valid atom pair calculations between the warp-group pair and the bit representing it can be flagged off. In this way, during force and energy kernels it is possible to determine whether a neighbour warp-group has interactions with any of the home warp-groups, using a small number of bit-shift operations. In a similar way it is possible to determine which home-neighbour warp-group pairs have no interactions, saving 32×32 = 1,024 distance checks per flagged-off warp-group pair.

Both the interaction map and the bitmap are stored in constant memory as in both cases all threads in a block will be accessing the same constant memory address. Considering that constant memory is finite and capped at 64 kB, there will be a point where large molecular systems will require more than 64 kB to store interaction data. For systems up to 100,000 atoms the constant memory size is adequate for storage. For larger systems, this data can be stored in global memory and accessed in order for each block and in an LDU (load uniform) manner for each thread in the block.

Initially, some of the binning methods were calculated on the host (c++). However, during the energy minimization experiments the binning code was redesigned to run entirely on the device in order to exploit the Keppler line's asynchronous streaming capabilities, as explained in the next chapter. That meant that less data copies would take place from host to device and vice versa. In regards to the constant memory allocation. The data was initially stored in global memory and then transferred to constant using the "`cudaMemcpyDeviceToDevice`" flag. This means that the data allocation would take place on the GPU with no need for the host's intervention. In Fermi and Keppler, constant memory is a cashed partition of global memory and explicit allocation is not necessary. However, we believe that explicit allocation is a good practice as both future programmers and the compiler know that the specific data has been designed to be used as constant memory.

6.1.3 Force calculations using warp intrinsics

The data structures obtained from the binning process will be used in energy and force calculation kernels. Force calculation kernels are of special interest as they are using warp intrinsics (shuffle in CUDA terms) to exchange values in force calculations between warp-group pairs. The pseudo-code *warpPairForces* demonstrates how warp-group pair processing is performed. There are two float3 variables held in registers $\vec{F_i}$ and $\vec{F_j}$ holding the home and the neighbour atom's force values. The routine loops over the 32 atoms of the neighbouring warp-group, summing up force values of each home atom *i* interacting with neighbour atom *j* and storing the result in a temporary float3 $\overrightarrow{F_{ij}}$ whose scope is within the loop. $\overrightarrow{F_{ij}}$ is then subtracted from $\vec{F_i}$ which after the end of the loop will hold the total force of atom *i*'s interaction with the 32 atoms of the neighbouring warp-group.

During each step of the loop, the force values $\vec{F_j}$ are also summed for each atom *j* interacting with atom *i* processed by threads with laneId (thread id inside a warp) 0 -> 31 (Figure 6.3), which belong to the same warp-group. This is done using a butterfly reduction with the shuffle command [16] at line 9 of the pseudocode for the variable $\overrightarrow{F_{ij}}$. The reduction sums the force values of each neighbour atom j interacting with home atom *i* hosted by threads with lane id

0 -> 31. At this point all threads in the warp are holding atom *j*'s force vector. Only the thread whose *laneId = j* will be eligible to store it though as each thread has only one allocated register to hold a neighbour atom's force vector. A warp voting instruction is issued to ensure that the neighbour atom's force vector will be stored by the correct thread. Neighbour atoms forces are added to global memory using an atomic add instruction straight after the warp-group pair processing. Then the buffer registers can accommodate the next warp-group's force vectors and so forth.

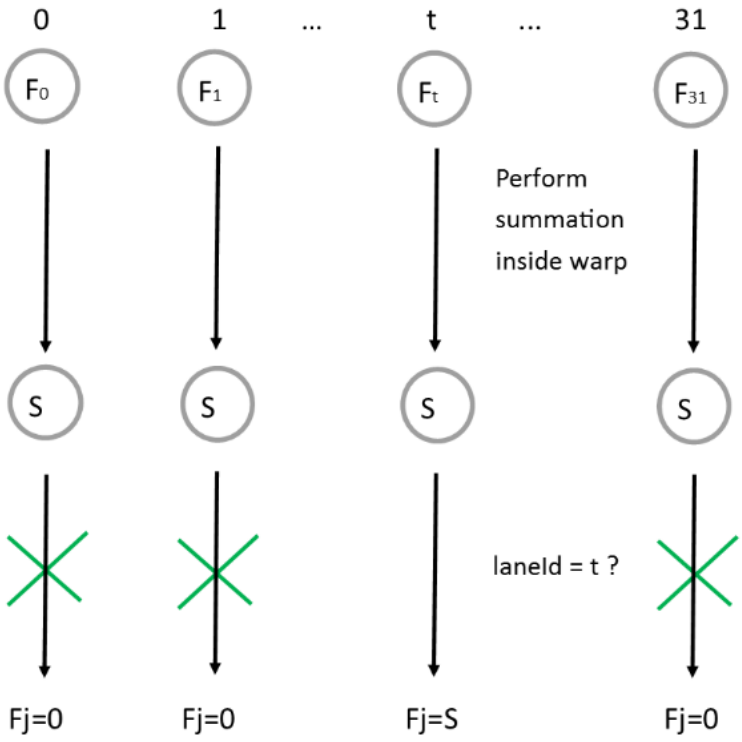


Figure 6-3 Threads with lane Id 0 -> 31 of a given home warp-group hold their individual force values for the interaction with the t^{th} atom of a neighbour warp-group. Threads have the same value S (forces sum) after reduction using CUDA's __shfl(). Only one S will be written after the warp voting, on the thread whose lane id = t.

routine: warpPairForces

//function parameters

float3 $\vec{F_i}$ // forces vector home pointer, held in registers.

float3 $\vec{F_j}$ // forces vector neighbour pointer, held in registers

//member variables

float rc2 // cut-off squared

int laneId // index of thread in warp 0-31

1. for j = 0 to 32
2. Require: rij2 //square distance between atoms i and j
3. Require: excluded
4. float3 $\overrightarrow{F_{ij}}$
5. if rij2 < rc2 && not excluded
6. $\overrightarrow{F_{ij}}$ = calculate VDW
7. $\overrightarrow{F_{ij}}$+= Calculate electrostatics
8. $\vec{F_i}$ += $\overrightarrow{F_{ij}}$
9. sum $\overrightarrow{F_{ij}}$ across all threads in the warp using __shfl
10. if laneId equals j
11. $F_j = F_{ij}$

The term Require refers to variables required for the routine. The variable excluded is a Boolean value which is true if the pair is not 1-3 connected.

6.2 Exclusions handling

In MMFF94s 1-2 and 1-3 bonded atom pairs are excluded and for 1-4 bonded atom pairs electrostatic forces and energy potentials are scaled by 75% [21]. This constitutes a challenging part for forcefield calculation algorithms in CUDA as the obvious solution of storing a 2D exclusion matrix would involve frequent random access global memory fetches and this would affect performance [46][53]. We designed a two-pass solution to this problem: the first step will take into account 1-2 and 1-3 exclusions and the second step will address the 1-4 scaling.

The 1-3 exclusion handling approach is designed around the fact that an atom pair is excluded if and only if its atoms are either directly bonded or bonded to a common atom. Handling 1-4 scaling can be more efficient by subtracting 25% of the force and energy potentials during bonded force and energy calculations rather than designing a method to detect 1-4 bonded atom pairs during computing non-bonded interactions. The above approaches are implemented in practice as follows.

6.2.1 1-3 exclusions

Our 1-3 exclusions implementation is performed with the aid of a 32 bit unsigned integer holding information regarding the bonding properties per atom. The first two bits hold the values 0,1,2 denoting whether the atom is a hydrogen bond donor, acceptor or none. This information is used to decide which calculation path is going to be taken for the range parameter values R_{ij} during the VDW calculations. The second two bits are used to store the bond order for the atom. Values 0,1,2 represent 1,2 and 3 bonds. The remaining $4 \times 7 = 28$ bits are used to store the index difference between the current atom and each non–hydrogen atom it is bonded to. Seven bits are adequate to store index differences in the range of [-64, 63].

PDB and Mol2 files preserve spatial locality amongst bonded atoms generally well, as reported in [46] and for most files the maximum index difference between two bonded atoms is in the

range of 7-63. For files whose indexing does not match the above criteria (ie. index difference between two atoms greater than 63 exists), we perform a one off re-indexing of the file such that the maximum index difference between a bonded pair is below 64. Protein molecules are protonated before simulations and the added hydrogens are expected to have index differences greater than ±63 with the atoms they are bonded to. For this reason, atoms with bond order 0 (meaning one bond) are using the whole 28 bit sequence to store the index difference with the atom to which they are bonded, which leaves adequate space to store the index difference in hydrogen bonds. This way we can easily detect whether an atom pair is directly bonded or bonded to a common atom during distance checks in the main part of the non-bonded force calculations with just a few bit shift and bit masking operations as demonstrated with the CUDA code snippet below.

```
__inline__ __device__ bool excluded(uint iBmp, uint jBmp, uint idx_i,
uint idx_j, uint maxDiff) {

        // bond order for ats i and j
        ushort bo_i = (iBmp & 12) >> 2;
        ushort bo_j = (jBmp & 12) >> 2;

        //shift out bits holding Donnor-Acceptor flags (not needed here)
        iBmp = iBmp >> 4;
        jBmp = jBmp >> 4;

        //bit masks for atoms i and j
        uint msk_i = (bo_i == 0) ? 0 : 25;
        uint msk_j = (bo_j == 0) ? 0 : 25;

        for (ushort i = 0; i < bo_i + 1; ++i) {

                uint ati = (iBmp << (msk_i - 7 * i)) >> msk_i;
                ati = (ati > 0) ? idx_i + maxDiff - ati : idx_i;

                for (ushort j = 0; j < bo_j + 1; ++j) {

                        uint atj = (jBmp << (msk_j - 7 * j)) >> msk_j;
                        atj = (atj > 0) ? idx_j + maxDiff - atj : idx_j;
                        //atoms i, j bonded to each other or common atom?
                        if (ati == atj || ati == idx_j || atj == idx_i)
                                return true;

                }
        }
        return false;
}
```

An easy way to avoid exclusions checking for every single atom pair is by recording the maximum distance of an atom and its 1-3 bonded pairs for each warp-group. Then for a given atom pair between two warp-groups, the exclusions checking need only be evaluated if their distance is less than the minimum of the two warp-groups maximum 1-3 bonded atom-pair distances. Performance checks showed that avoiding the min operation and just using the max 1-3 distance available for the home warp-group performs slightly faster (2-3%), hence we adopted that in our code.

The maximum 1-3 distance data-structure is computed by a fast helper kernel using dedicated data structures for 1-3 bonded pairs. This array can be stored in constant memory, but only for smaller systems (up to 75,000 atoms) as it will lower the available constant memory resources for the interaction pair maps. For larger systems it can be stored in global memory and accessed in a load uniform manner for each warp.

6.3 Bonded forces and 1-4 electrostatic scaling

Bonded forces are a less complex O(*N*) computation and they can still benefit from being implemented on the GPGPU. In contrast to the non-bonded part of the computation this algorithm is not calculation bound but memory bound. Therefore our focus was towards designing a decomposition scheme that minimizes memory accesses. Our solution came through a bonded pair decomposition scheme as depicted in Figure 6-4.

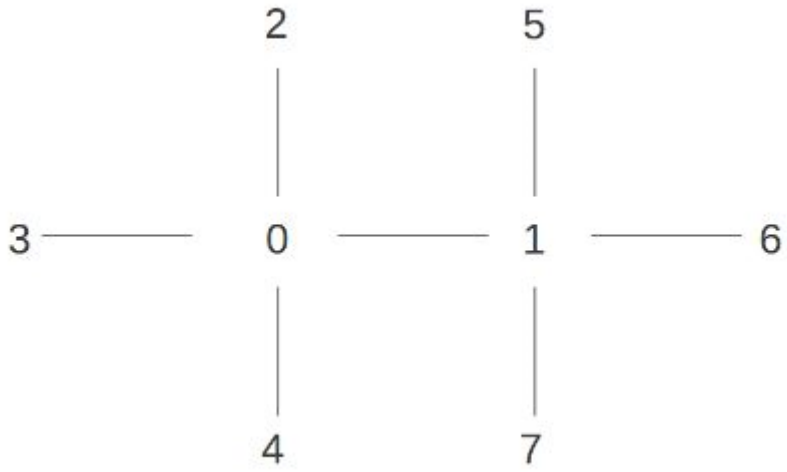


Figure 6-4: Merged structure for bonded interaction. All the bonded atom pairs 0,1 are collected at first. Then their directly bonded atoms on their left and right are attached in positions 2-4 and 5-7 accordingly.

A data-structure containing the parameters and indexes of each covalently bonded pair (0, 1) along with the rest of the atoms directly bonded to the pair (2,3,4,5,6,7) is gathered. For atoms containing less than four bonds, positions 2-7 can become ghost particles with zero parameters and hence no effect on the computations. For example if atoms 0,1 are both carbons connected via a triple bond, atoms 3,4,6 and 7 will be the ghost particles. Using the above scheme we can calculate bonds, angles, dihedrals and out of plane forces and potentials as well as scaling the electrostatics in one kernel. Special care has to be taken for angle and out of plane potentials as the same bonded atom triplets or quadruplets for angle and out of plane potentials can be found multiple times in a kernel. Parameters for duplicate angle triplets or out of plane quads need to be set to zero so their existence will not affect accuracy. Bonds and dihedrals can then only be unique using this scheme.

As mentioned in previous sections, we take advantage of this scheme to perform a 25% subtraction on the electrostatic potential and forces in order to achieve the overall 75% scaling for the 1-4 bonded atom pairs. Some issues here require extra attention: if a 1-4 bonded atom pair belongs to a five member ring, then it is also 1-3 bonded and hence the computation is excluded. In addition if the 1-4 pair is a member of a 6-atom ring, then it may occur in this

structure twice and that means that the duplicate should be excluded as shown in Figure 6-5: An Imidazole ring and a Pyrazine ring. Nitrogen atoms in Imidazole are both 1-3 and 1-4 bonded. Nitrogens in Pyrazine are 1-4 bonded in 2 different dihedral quad structures. Similar issues occur in tricyclic rings and four member rings. In the case of tricyclic, atoms in positions two and five in Figure 6-5 may belong to the same atom and for four member rings they may be bonded. We address all of the above issues by introducing a 32-bit bitmap, which is included in the 8 atom structure and read in order inside the kernel. A single bit represents each 1-4 pair inside the structure and the computation result for the electrostatics is set to 0 if its corresponding bit is set.

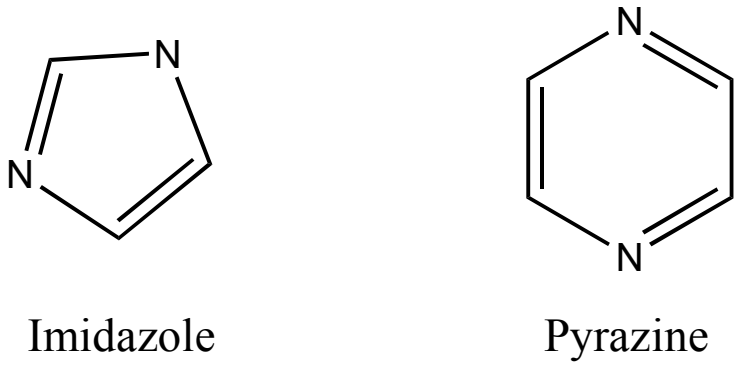


Figure 6-5: An Imidazole ring and a Pyrazine ring. Nitrogen atoms in Imidazole are both 1-3 and 1-4 bonded. Nitrogens in Pyrazine are 1-4 bonded in 2 different dihedral quad structures.

6.4 Minimization

The above algorithms have been incorporated in our drug design application performing manual real-time docking simulations with the help of a haptic device. The application is using a steepest descent (SD) minimisation algorithm, which features a high number of autonomous and fast executing energy minimization steps. This fits well with our application as frequent force feedback is required for the haptic device to match 1 kHz rates. It should be noted that the algorithms listed in this report can be applied to molecular dynamics simulations, possibly using a velocity Verlet method or similar.

6.5 Performance

In order to run performance benchmarks for our algorithm, we used a set of pdb files spanning from 1,500 to 65,000 atoms (Table 6-22), which represents the range of systems we aim to support with our application. The selection criteria for this test set was mainly system size and the files were chosen such that there would be adequate atom count difference between any two subsequent pdb files. For these benchmarks, an NVIDIA GTX 680 GPGPU was used connected to a workstation powered by a quad core Intel Xeon E5335. Our GPGPU code was compiled with -arch=compute_30 and -use_fast_math compilation flags. The former compiles the code for GPGPUs with compute capability 3.0 or higher. The latter forces the compiler to use the fast option for all mathematical operations which in our case affects accuracy on divisions and exponents (plus other functions not included in our code such as trigonometrical functions, logarithms etc.), only by a very small factor [16]. As we can see in the accuracy section that follows, this effect is almost non-noticeable. Some additional information on the system's configuration is given in Table 6-3: Additional information on system configuration3.

filename	*atoms*
1UGM.pdb	2,103
3RDD.pdb	2,786
3FS1.pdb	4,100
3F9E.pdb	4,740
3FAU.pdb	5,917
2O0O.pdb	7,165
3VB3.pdb	9,903
3O05.pdb	13,053
2EAR.pdb	15,528
1TBG.pdb	26,927
1AFR.pdb	34,863
4A0B.pdb	47,339
1A8R.pdb	56,008
3OJ5.pdb	61,418

Table 6-2: The pdb file test set.

NVIDIA Driver	Cuda Version	Ubuntu Version	DRAM	PCIE Slot
326.8	5	10.04LTS	2GB	PCIE x16

Table 6-3: Additional information on system configuration

One of the major differences between our proposed algorithm and that of common cell lists is the binning process. Preparing the 3-dimensional irregular grid is slightly more computationally expensive as it requires three sets of O($NLog^2N$) bitonic sort operations as follows. One bitonic sort on each XY stack of the grid, sorting particles according to their Z co-ordinate to form blocks of 256 z-sorted atoms. Block partitioning uses a sequence of two bitonic sorts (y and x axis) to partition particles into eight warp-groups inside each block. The sequence takes place using a dedicated thread-block for each super-cell inside a kernel and the overhead of the extra sorts is minimal due to data re-use. If atom migration is switched on, then there are two more calls for bitonic sort, sorting particles according to their x and y co-ordinates, in order to donate atoms to neighbouring columns in the x and y direction. Processing times for the routines used during binning are listed in Table 6-44.

Molecule	Atoms	Bin 2D	X/Y/Z-direction Sort	Block Partitioning
3FAU	5,917	10.5	45.9	48
2OOP	13,053	18.4	65.2	69.7
2F3M	22,303	25.76	94.18	105.39
4AOB	47,339	42.15	293.162	218.162

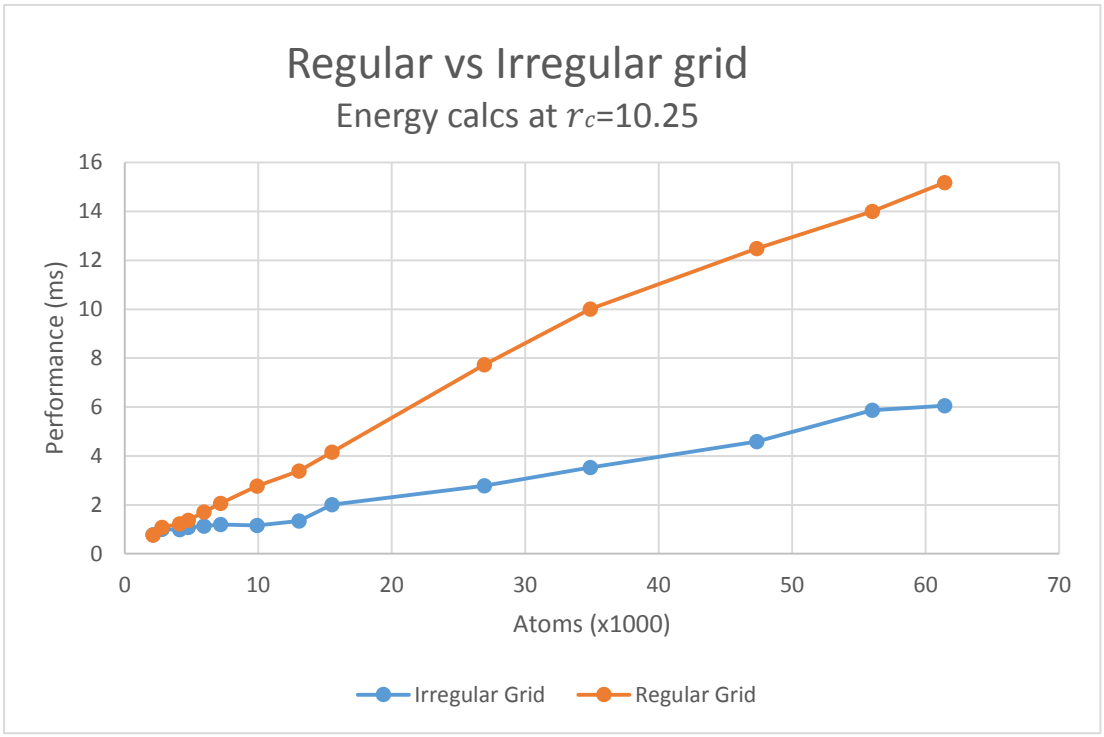
Execution times are displayed in microseconds (µs)

Table 6-4: Performance of binning N atoms into an irregular grid with base $r_b = 13.75Å$. Sort Z and Block Partitioning are the most expensive routines during the binning process.

In Figure 6-6, we can see a direct comparison of the performance of the two decomposition methods during short range non-bonded interaction calculations using a cut-off distance of 10.25Å and a base length of 13.75 Å for our decomposition approach. Both implementations run the same code snippet for the VDW and electrostatic calculations. In order to make a fair direct comparison, 1-4 electrostatics were not scaled and both approaches were running a similar exclusion approach for 1-2 and 1-3 atom pairs. In addition, we disabled the warp-group bit-map optimization on our approach and ran all versus all atom comparisons between cell pairs as our benchmark initial approach did not use this optimization.

Regarding potential energy calculations in Figure 6-6a, for systems up to 9,000 atoms there is a performance gain building gradually up to 2x. For systems ranging between 9,000 and 65,000 atoms we can see a 2x to 3.5x speed up. This outcome was expected as the efficiency of our approach improves proportionally to the system size. On smaller systems (<9,000 atoms) load balancing is harder to achieve using our approach. That is because the XY stacks consist of fewer atoms with the top level unbalanced. Our migration technique is not effective on these sizes for reasons we explained in the algorithm implementation section. That means that the ratio of load balanced to total cells will be lower to that of larger systems, but still higher to conventional cell-lists using the regular grid approach. That explains the small performance gain. For larger systems, the ratio of load balanced cells to total number of cells gets higher and the performance gains of our space decomposition approach follow suit reaching 2.5x.

Regarding the force calculations, Figure 6-6b depicts the comparison between the two approaches. In this benchmark we can see the performance gains from both load balancing and Newton's third law using warp intrinsic functions. The curves show almost an extra 1x speed-up compared to the energy benchmarks (reaching 3.5x in total). This is consistently evident almost throughout our sample test set and solely due to Newton's third law implementation using warp intrinsics.



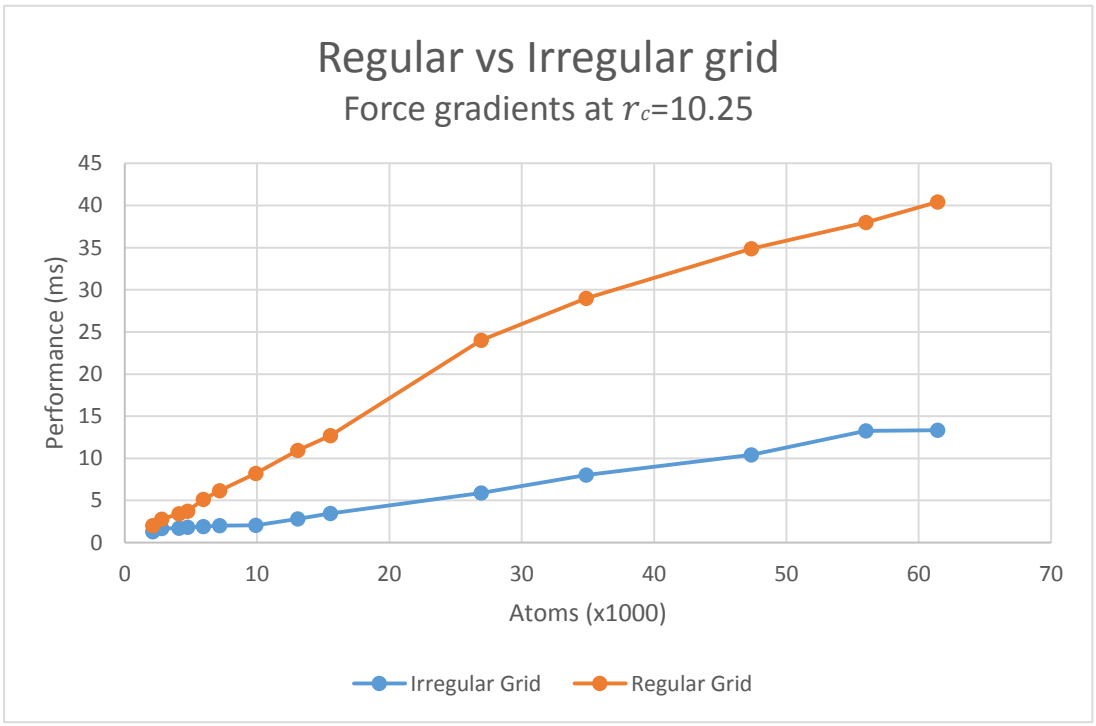
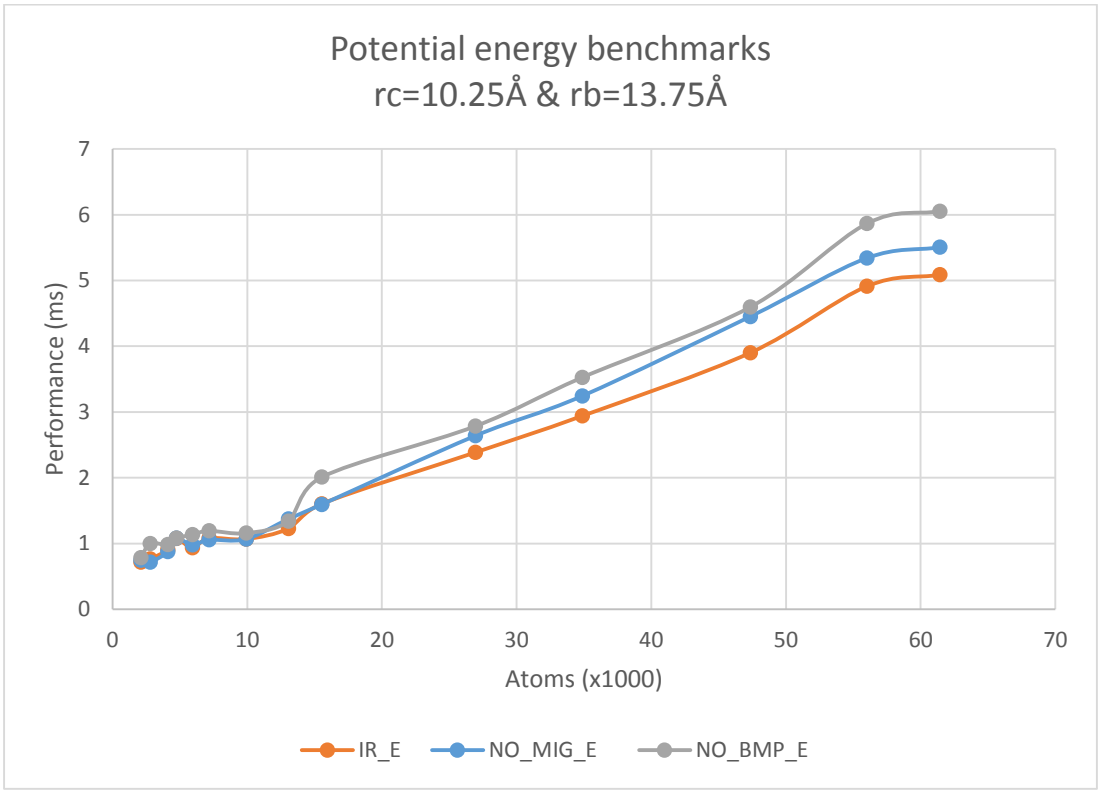


Figure 6-6 a (above) and b (below):. The light blue curves represent the performance of our decomposition approach in energy and force calculations. The orange curves represent the performance of conventional cell-list decomposition schemes for energy and force calculations.

Figure 6-7 shows the speed-up effect of the warp-group bitmap and atom migration optimizations that we referred to in this report. In Figure 6-7a (energy potential computations) we can see that the speed-up effect of minimizing the void distance checks using the warp-group pair 64 bit bit-map is approximately 20%. The same optimization for the force calculations is ranging between 25%-35%. This is because in our force gradients implementation when the warp-group pair bit is set, it also saves 32 atomic force updates to global memory for the neighbour warp-group's atoms. Regarding atom migration we can see a consistent 7-10% improvement for system sizes above 8,000 atoms.



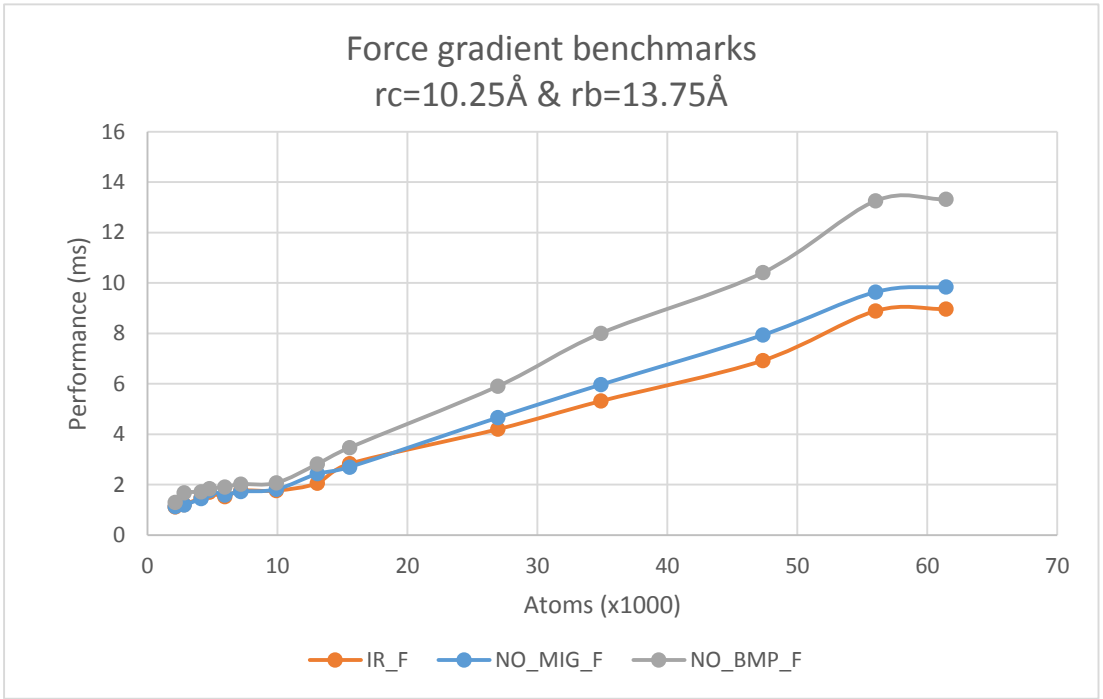


Figure 6-7: Benchmarks showing the performance gains for warp-group bitmap and atom migration techniques on both force and potential energy calculations. The blue curve represents performance when atom migration is disabled. The grey curve shows performance when the warp-group interaction bitmap optimization is disabled. The orange curve represents performance with all optimizations enabled.

In the description of our algorithm we mentioned our concerns regarding the warp-groups bounding box elongatedness and its effect in performance especially for smaller cut-off distances. In order to resolve how it affects performance and determine the best way to launch our grids, we designed a set of experiments. We recorded the average aspect ratio of the warp-groups that different base lengths r_b in the range 8.25-17.25 Å produce for each pdb file across our test set. We repeated the procedure for different cut-off distances $r_c$in the range 9-12 Å, and then we took the average of these readings for each r_b/r_c combination.

Our findings are summarised in the curves of the graph in Figure 6-8. The four coloured curves represent the average aspect ratio for our four different cut-off set-ups. The four curves behave similarly showing a peak of 5.75-6.00 at approximately r_b= 13.75Å. The reason for small deviations in the curves across different cut-off set-ups is due to atom migration. As explained previously, during atom migration, stacks donate atoms to xy direction, as long as $r_b \geq r_c$. Therefore it makes sense that for the same base side lengths and different cut-off distances, slightly different warp-group dimensions will be produced.

In Figure 6-9 we show performance results for various molecular systems (pdb files) running force calculations for the short range non-bonded interactions using three different cut-off set-ups 8, 10 and 12Å. From the curves we can see that performance deteriorates when the base side length approaches the two extrema 8 and 19.25Å. This supports our hypothesis that the cell shape affects performance and that low warp-group aspect ratios deteriorate speed of execution by a factor ranging from 10-40% compared to the peak. From the curves we can observe that the performance deterioration towards the two extrema points of the graph is greater on larger systems whereas on smaller systems is barely evident. That makes sense as smaller systems are processed by smaller grids (ie. 3RDD: 4x3x6 cells), so a large proportion of the cell-pair processing comes from cells located in the borders, which might in turn affecting the outcome of the experiment as border cells are processed by a lesser number of pair interactions. However, this is not a discouraging factor, as the need for performance gains is higher for larger system sizes.

Best performance is usually observed in the base length range of 12 - 14.25 Å. From the curve in Figure 6-8 we can see that this range corresponds to the most compact warp-group shapes

with the highest aspect ratio. However the peak varies in this range for different molecules, which means that there are other factors there that influence performance apart from cell compactness and there is no easy way to identify the base-length that produces the peak for each system. For simplicity reasons we chose to pick a fixed base length of 13.75Å for all our benchmarks in this report and our simulations so far, which stands between the range that produces the peak performance and the most compact warp-groups for our test set. However, a better approach could be to write algorithms to detect in the first few thousand energy minimisation steps which base length produces the peak performance and switch to that for the rest of the simulation.

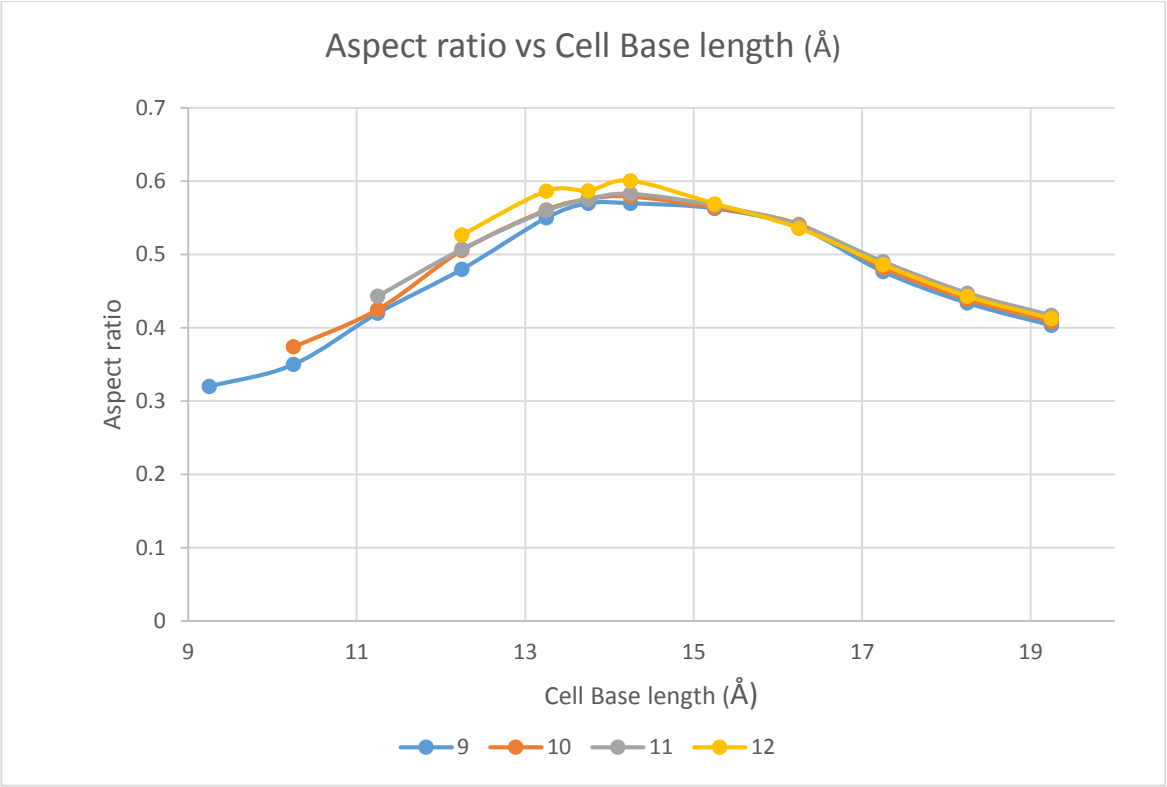
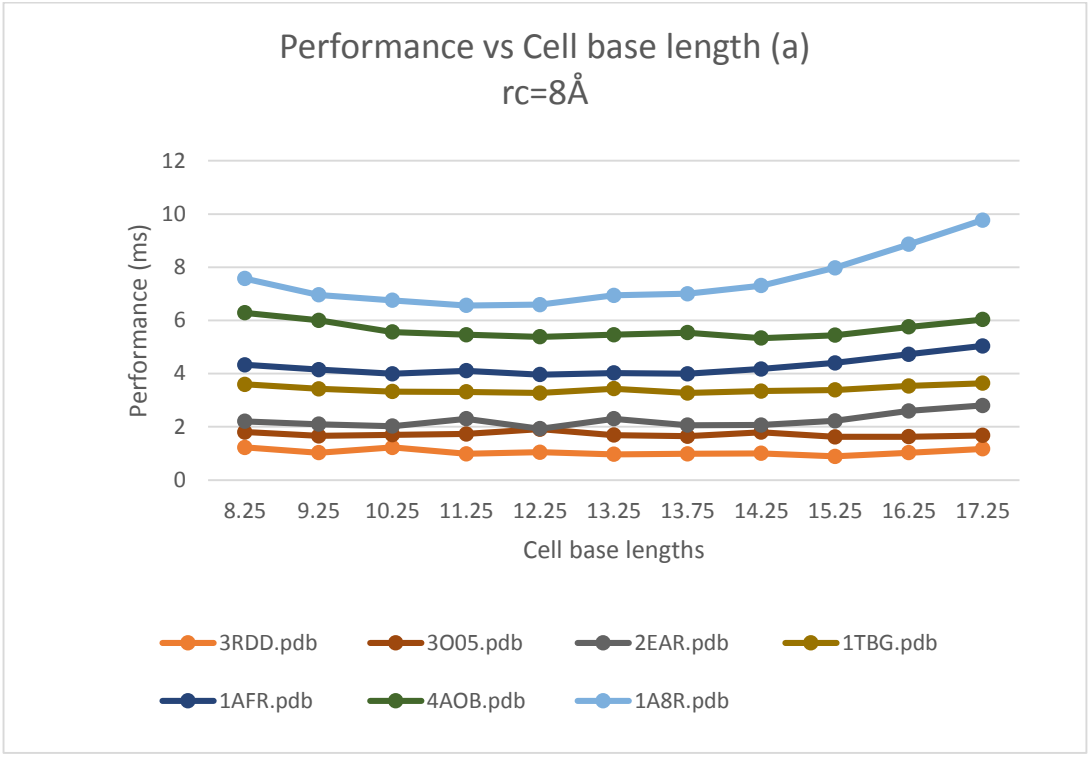
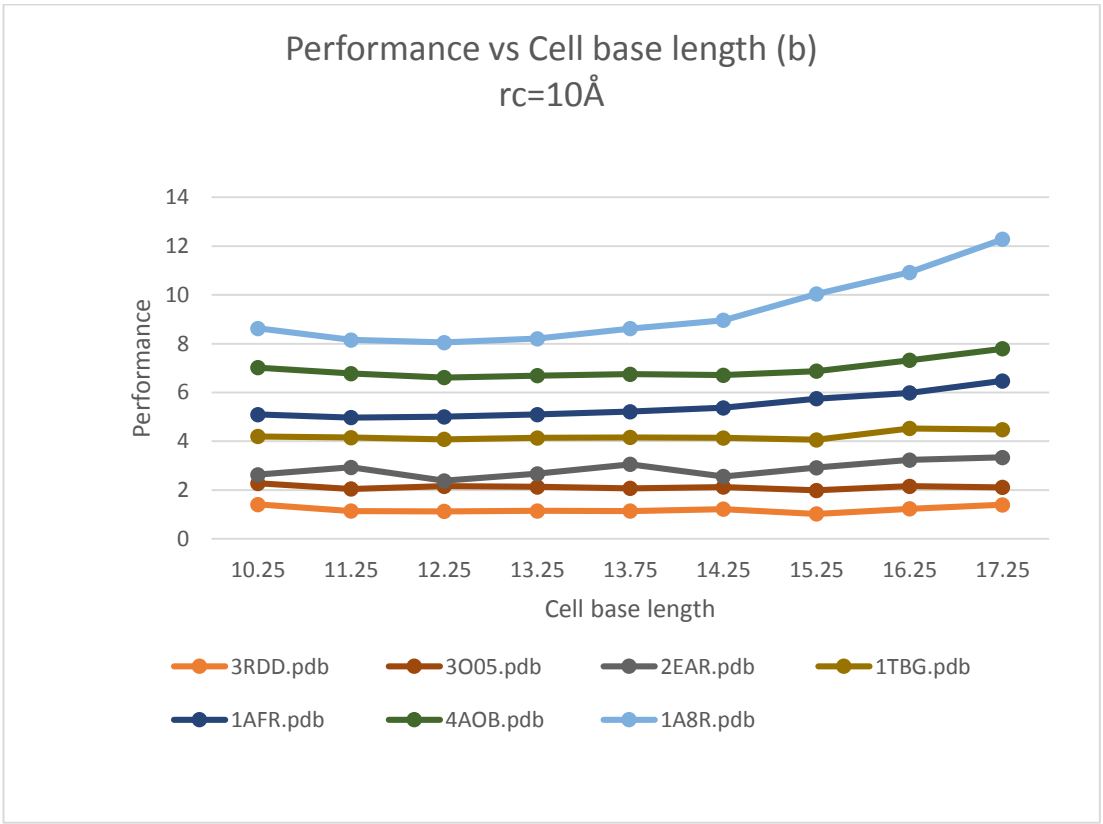


Figure 6-8: The average warp-group aspect ratio recorded across our test set for different base dimensions.


Performance vs Cell base length (a)
rc=8Å
Performance (ms)
12
10
8
6
4
2
0
8.25
9.25
10.25
11.25
12.25
13.25
13.75
14.25
15.25
16.25
17.25
Cell base lengths
3RDD.pdb
3O05.pdb
2EAR.pdb
1TBG.pdb
1AFR.pdb
4AOB.pdb
1A8R.pdb


Performance vs Cell base length (b)
rc=10Å
Performance
14
12
10
8
6
4
2
0
10.25
11.25
12.25
13.25
13.75
14.25
15.25
16.25
17.25
Cell base length
3RDD.pdb
3O05.pdb
2EAR.pdb
1TBG.pdb
1AFR.pdb
4AOB.pdb
1A8R.pdb

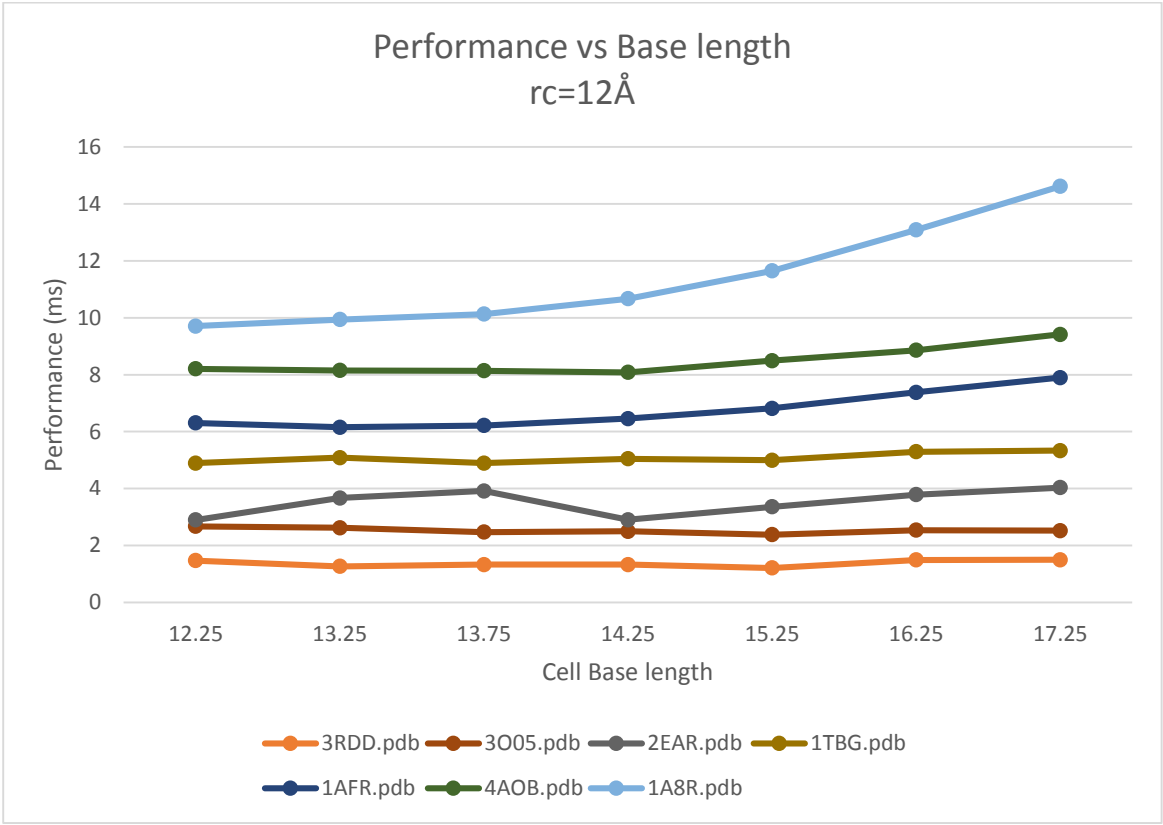


Figure 6-9 a-c (top to bottom): Performance versus cell base lengths for cut-off distances 8, 10 and 12 Å. Peak performance is observed for base dimensions in the range [12 – 14.25 Å].

The curve in Figure 6-10 depicts the performance of our algorithm performing 1,000 steepest descent energy minimization steps calculating the full forcefield (Bonded plus Non-bonded energy potentials and forces) using the algorithms described in the present chapter at r_c = 10.25Å and r_b = 13.75Å for different molecular systems interacting with a 31- atom ligand in vacuum. Cell lists are updated every 15 steps as long as there is a conformational update. The initial aim of our application is to simulate the drug–protein interactions in a haptic driven simulation environment. The ligand is driven by a haptic device, which in turn returns a force feedback at a 1,000 Hz rate. The curve in Figure 6-10 clearly depicts a linear scaling . Regarding the system sizes that we wish to support with our application, the target 1,000 Hz feedback rate magnitude can be reached for smaller system sizes up to 10,000 atoms. For bigger system sizes (10,000 – 70,000 atoms) we are aiming to design an interpolation method that will be able to generate smooth feedback based on the actual results obtained from the energy

minimization.

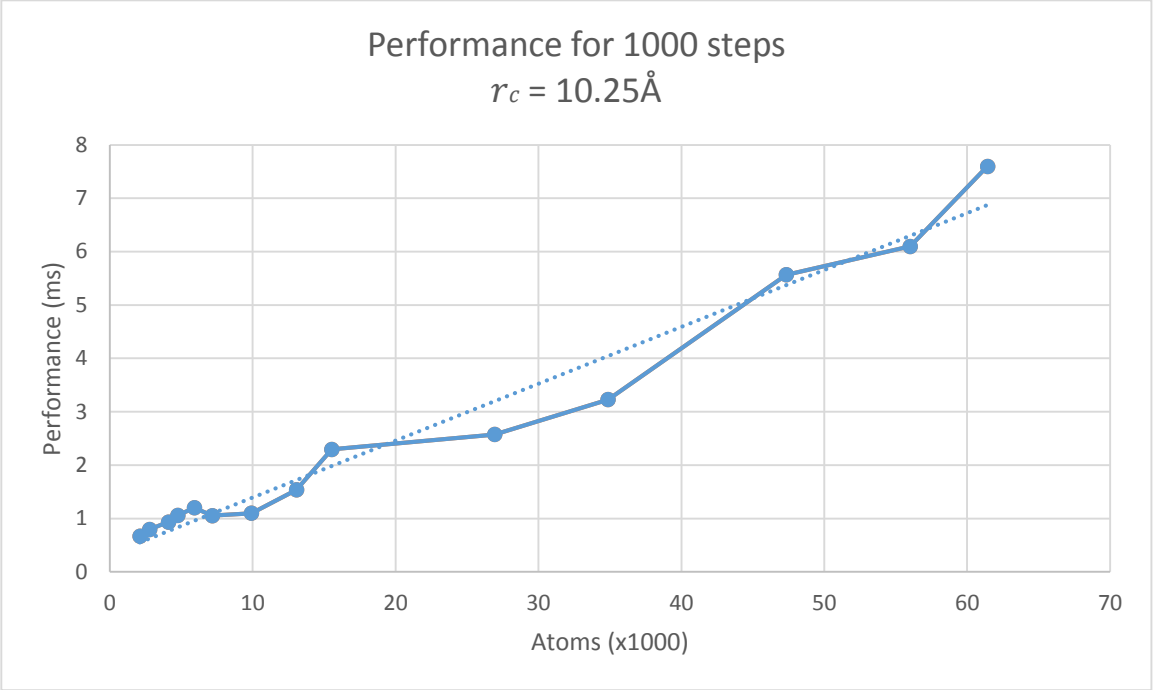


Figure 6-10: Performance benchmarks for executing 1000 energy minimization steps at r_c= 10.25Å

6.6 Accuracy

In order to benchmark the accuracy of our GPGPU implementation we decided to perform tests against the CPU version of our code, which uses the Openbabel 2.3 library to perform MMFF94s computations [31]. Our code at the moment supports single precision only. Comparing codes in single precision between GPGPU and CPUs using the CPU version as the frame of reference can lead to false conclusions, due to the rounding of floating point operations caused by either the structure of the code or the rounding methods used by the hardware architecture. Although, both architectures support both IEEE and FMA [78], we decided to compare the CUDA implementation using the FMA rounding option against the double precision version of our CPU code and investigate rounding deviations from the GPGPU

side only. During our first attempts, we were getting a significant error due to rounding coordinates in the CUDA code, as a result of casting them to single precision (floats) and porting them to the GPGPU. The rounding error was then growing in magnitude as the co-ordinates are used for all forces and potentials computations. For a fairer comparison, we decided to run our experiments by casting the CPU code's co-ordinates to single and then back to double precision, in order to avoid judging for rounding errors of non-GPGPU code. The results are listed in Table 6-55 for the systems of our test set. Regarding energy potentials the metric $dev_E = \frac{E_{CPU} - E_{GPU}}{E_{CPU}}$ measures accuracy as a deviation of the GPGPU energy (E_{GPU}) from the CPU energy value (E_{CPU}). The results are accurate up to 6 significant Figures. Regarding force values, a similar metric is used for the x,y,z components of every atom *i* that belongs in the system $devF_i = \frac{Fgpu_i - Fcpu_i}{Fcpu_i}$. Then the average is computed as $avg\ devF_i = \frac{\sum_0^n |devF_i|}{n}$ for every x,y,z force component for a system with n atoms. The results are similar and the average deviation from the CPU force values is consistently accurate to 6 significant Figures. Finally the maximum deviation from the master CPU value for each force x,y,z component has been recorded and it is accurate to 4 significant Figures.

Assessing the accuracy of our energy potential and force calculations is a non-trivial task. That because it is difficult to argue or find information regarding a cut-off value for our metrics that distinguishes accuracy. Certainly, on our table the lower the magnitude of error the higher the accuracy is. Regarding the force calculations we can argue that our results are adequate as the PDB standard for accuracy is 3 decimal places on XYZ atom co-ordinates. Considering our force calculations guarantee 3 decimal places of accuracy, we can argue that the accuracy of our calculations is adequate. We could also argue that 4 decimal places of accuracy is the cut-off Figure to guarantee accuracy as 3 decimal places would start introducing small errors in our conformations.

Regarding the potential energy accuracy during energy minimizations, 5 decimal places are also adequate for accuracy. During our experiments energy values on conformation updates have been inspected for various reasons. We can only argue here that conformational updates for two different structures whose energy difference was in the band of E-05 would only take place on lamda values in the range of E-05 to E-06. These lamda values can only affect systems

that contain at least one atom whose force vector has at least one component with magnitude similar to ten into the power of two or greater. This in turn means that the atoms are misplaced anyway as this magnitude represents a high attractive or repulsive force and should move towards the trajectory of their force vector. Therefore we can argue that our potential energy accuracy cannot produce any errors during energy minimizations and conformation updates.

In regards to whether there is a cut-off value for potential energy accuracy, this is a difficult question to answer. The fact is that larger systems tend to produce higher potential energy values. There will be system sizes beyond which, 5 decimal places could be inadequate to quantify with certainty the binding of a ligand against a protein cavity. Perhaps a more robust way to quantify energy differences on such large systems could be to isolate the potential energy produced from the atoms around a certain radius of the ligand. However this issue is something to consider on future better performing versions of the software as the present one can only simulate systems up to 30,000 atoms. For these systems five decimal places of accuracy are enough to quantify the binding energy differences of good and bad binding protein-ligand structures.

filename	*devE*	*avg_devFx*	*avg_devFy*	*avg_devFz*	*max_devFx*	*max_devFy*	*max_devFz*
3O05.pdb	1.3E-05	3.116E-06	3.468E-06	3.247E-06	2.328E-04	2.908E-04	2.106E-04
2EAR.pdb	1.0E-06	3.274E-06	3.259E-06	3.234E-06	2.698E-04	2.675E-04	3.095E-04
1TBG.pdb	1.0E-06	3.370E-06	3.385E-06	3.454E-06	3.217E-04	3.619E-04	2.615E-04
1AFR.pdb	3.0E-06	3.251E-06	7.514E-06	2.869E-06	3.099E-04	3.246E-04	2.882E-04
4AOB.pdb	2.0E-06	3.226E-06	4.969E-06	2.812E-06	3.198E-04	3.355E-04	2.950E-04
1A8R.pdb	4.0E-06	3.202E-06	2.424E-06	2.756E-06	3.298E-04	3.464E-04	3.018E-04
3OJ5.pdb	2.0E-06	3.177E-06	1.212E-06	2.699E-06	3.397E-04	3.574E-04	3.086E-04

Table 6-5: Accuracy tests for various molecules across our test set.

6.7 Comparison to other implementations

Looking at the above performance results it can be argued that code able to run faster force and energy calculation results has been reported [41, 51]. However, it should be noted that these algorithms use force-fields that incorporate Lennard-Jones or Exp-6 based potentials

for the non-bonded part of their force-fields. Our force-field of choice uses a special “buffered 14-7”, which is more accurate but also more computationally expensive as Halgren 1995 proved [16]. For this reason, our short-range non-bonded forces code needs considerably more floating point operations, including exponents and some branching compared to other published samples. To be more specific our code for Van Der Waals forces requires 40 FLOPS plus an exponent in the case of a warp-group of hydrogen bond donors, 34 FLOPS in the case of hydrogen bond donor-acceptor pairs and 42 FLOPS plus an exponent in the case of a warp-group that contains at least one of each of the above donor or acceptor groups as demonstrated in the following code snippet. Codes that use the L-J potential require 9-10 FLOPs maximum as in the code snipet reported by Stone in [46]. Knowing that algorithms for short range non-bonded forces are characterised as compute-bound, it makes sense to take into consideration the FLOP requirements of each code, when we perform performance comparisons.

```
__inline__ __device__ void forceCalc(NBAtom_t *sAt_b, NBAtom_t *sAt_h, float3 *fHome, float3 *fOct, uint oct, uint laneId, float rc2, float diff1, uint maxDiff) {

#pragma unroll 4
	for (uint i = 0; i < 32; i++) {

		float3 Fij;
		uint position = (oct << 5) + i;

		//a - b
		Fij.x = sAt_h->x - sAt_b[position].x;
		Fij.y = sAt_h->y - sAt_b[position].y;
		Fij.z = sAt_h->z - sAt_b[position].z;

		float rab2 = Fij.x * Fij.x + Fij.y * Fij.y + Fij.z * Fij.z;

		uint bDA = sAt_b[position].DAE;

		const bool include = (rab2 < rc2) && ((rab2 > diff1) || (!excluded(sAt_h->DAE, bDA, sAt_h->idx, sAt_b[position].idx, maxDiff)));

		if (!include)
			Fij = make_float3(0.0f, 0.0f, 0.0f);

		else {

			bDA = bDA & 3;
```

```
//normalize force vector
float r_ij = rsqrtf(rab2);
Fij.x *= r_ij;
Fij.y *= r_ij;
Fij.z *= r_ij;

//----------------------Van Der Waals------------------------------

float epsilon = (sAt_b[position].GxA * sAt_h->GxA) / (sAt_b[position].vSqrt + sAt_h->vSqrt);

float R_ij = (sAt_b[position].R + sAt_h->R);

//hydrogen bond donor?
if (!(bDA == 1 || (sAt_h->DAE & 3) == 1)) {
		const float g_AB = (sAt_b[position].R - sAt_h->R) / R_ij;
		const float g_AB2 = g_AB * g_AB;
		R_ij *= (1.2f - 0.2f * expf(-12.0f * g_AB2));

}

float R_AB6 = R_ij * R_ij * R_ij;
R_AB6 *= R_AB6;
epsilon /= (R_AB6);

// hydrogen bond acceptor-donor pair
if (bDA * (sAt_h->DAE & 3) == 2) {
		epsilon *= 0.5f;
		// R_AB is scaled to 0.8 for D-A interactions -  epsilon is not scaled.
		R_ij *= 0.8f;
}

r_ij = 1.0f / r_ij;
const float q = r_ij / R_ij;
float q6 = q * q * q;
q6 *= q6;
const float q7 = q6 * q;
const float erep = 1.07f / (q + 0.07f);
float erep7 = erep * erep * erep;
erep7 *= erep7 * erep;
const float term = q7 + 0.12f;
const float term2 = term * term;
const float eattr = (-7.84f * q6) / term2 + ((-7.84f / term) + 14.0f) / (q + 0.07f);
float dE = (epsilon / R_ij) * erep7 * eattr;
//-------------------------------------------------------------

//------------------electrostatics-----------------
r_ij += 0.05f;
rab2 = r_ij * r_ij;
dE -= ((sAt_b[position].cq * sAt_h->cq) / rab2);
//-----------------------------------------------

Fij.x *= dE;
Fij.y *= dE;
Fij.z *= dE;

//gradient vector
```

```
                fHome->x -= Fij.x;
                fHome->y -= Fij.y;
                fHome->z -= Fij.z;

        }

        for (uint ii = 16; ii >= 1; ii /= 2) {

                Fij.x += __shfl_xor(Fij.x, ii, 32);
                Fij.y += __shfl_xor(Fij.y, ii, 32);
                Fij.z += __shfl_xor(Fij.z, ii, 32);

        }

        if (laneId == i)
                *fOct = Fij;
    }
}
```

The present chapter has provided a technical comparison between our improved cell-list based approach to the conventional regular grid one. We proved that load balancing in cell-list based algorithms is feasible and it improves performance by at least 2× for systems containing more than 7000 atoms. We also provided a high performance solution to the Newton's third law issue that cell-list based algorithms faced by using warp-intrinsics available in Kepler devices, which improved performance by an additional 1×.

Comparing this approach to others is a little less trivial. The Verlet-lists are a popular algorithm performing well in clusters but not GPGPUs. This is because it relies solely on GPU's global memory, requesting a vast amount of random memory accesses with a proportional rate of cache memory misses. We initially experimented comparing cell-list and Verlet-list approaches and the latter performed 6-8× slower than the former (see section 5.3). In addition, previous research has shown that Cell-lists perform better in shared memory architectures for their ability to group nearby atoms quite well and use on-chip memory sources such as registers and shared memory for processing [75].

Anderson reported a factor of 4× improved performance for Verlet lists by sorting the system atoms using Hilbert curves and achieving higher L2 cache hits during computations [41].

However, re-indexing atoms to achieve spatial coherence means that the algorithm cannot rely on the consecutive indexing along the polymer chain anymore and that will eliminate the option of exploiting index differences for more efficient exclusions handling. Finally, the above implementation and others published more recently, could not take advantage of Newton's third law as this would generate a vast amount of atomic global memory writes and a resulting slowdown of execution [39 - 40, 79 - 80].

Eastman and Pande reported an approach of assigning atoms into blocks of 32 along the polymer chain of proteins. Then, they performed pair comparisons on groups whose bounding boxes are less than the cut-off apart [51]. This method uses Newton's third law, solves load balancing problems, and exploits fast on-chip memory during group-pair comparisons by loading the atom blocks into shared memory first, similar to ours. However it lacks control of the shape of the warp-groups formed compared to ours as it relies on the indexing of pdb files. As we proved in this chapter work-groups aspect ratio affects performance (Figure 6-9) whereas the method in comparison cannot guarantee an efficient aspect ratio. Moreover, our further grouping of eight adjacent warp-groups in a super-cell results in reduced global memory fetches over their method in comparison, as it enhances data reuse.

6.8 Summary

In this chapter, we presented a new cell-list based algorithm for non-bonded interactions whose performance scales linearly with system size. This algorithm is fine-tuned for GPGPUs and the Kepler architecture. It improves parallelism using an irregular grid approach and boosts performance using intra-warp functions, exchanging forces values between registers. Bonded forces calculations are also designed in a way to improve the overall efficiency of the algorithm by accommodating part of the exclusions handling for the non-bonded interactions. The forthcoming optimization plans included stream-processing based approaches in order to improve our step minimization algorithm's efficiency. Implementation details and performance results are listed in the following chapter.

Exploring potential energy surfaces using hybrid meta-heuristics and CUDA-Streams

In this chapter, a hybrid meta-heuristics method of energy minimization and conformational sampling is provided. The proposed method has been designed to suit real-time molecular docking simulations, where the time-lapse between two successive ligand poses is relatively short (33ms in our case). In these situations the energy minimization problem becomes increasingly complex and chaotic. Its complexity stems from the need to calculate a series of computationally intensive functions to model atom conformations in space (force-field). The problem becomes chaotic (3N-dimensional) during the search of optimal solutions that will drive the system to more accurate conformations and lower energy states in a potential energy landscape. An adequate solution of this problem needs a method that is able to bring the protein-ligand system into successive stable conformational states. In this report we propose a method that solves this problem on a visual docking simulation software for small to medium size fragments (<30,000 atoms). Our proposed method is tuned to take advantage of recent advances in GPGPU computing with asynchronous kernel execution. The present chapter has been submitted for publication in the Royal Society for Chemistry's Faraday Discussion special issue 169: "Molecular Simulations and Visualization".

7.1 Initial experiments

Given that each minimization cycle must be performed within a 33ms timeframe, the initial plan was to find a local minimum and exit searching in the hope that the minimum would correspond to a stable molecular conformation. For this reason we used a simple steepest

descent (SD) step minimization approach. Its characteristic is that it takes only downhill moves, which results in converging into the first local minimum it finds [81-83].

The conformation sampling that we obtained from this method was slow and sometimes erroneous and it only helped us understand some additional aspects of the problem we were trying to solve. An example conformation produced by the algorithm is presented in Figure 7-1 and it is obvious that the atom configurations are wrong. The 6 atoms in the aromatic ring should be perfectly co-planar, but in this instance they are not. The test system was relatively small (<2,500) atoms and there were no performance restrictions. The SD algorithm was converging before *t* expired and the system would quickly optimize its configuration, within a few minimization cycles, up to a certain erroneous extent as demonstrated in Figure 7-1. Further improvements on the system's configuration would be either really slow or not evident and the conformation in the pocket area would always be non-ideal. As a first thought one could suspect that this would be an accuracy problem on the force-field calculations. However, probing the ligand into a different area, inducing further conformational updates, would quickly restore the previously distorted area and cause the same problems in the area of latest interaction.

From the above, we ascertained that, for the type of protein ligand simulations performed by our application, finding the first potential energy minimum on the molecular energy landscape would not guarantee a stable conformation around the area of interaction between the two molecules. This might be because the rest of the protein could be well minimized and no scalar value λ for the Equation 3-15 would exist, able to move all atoms in Euclidean space such that the system would reach a lower energy value. What is also worth noting is that keeping the ligand stalled in its position in order to allow the system to slowly recover would not always work either. This gave the impression that every new minimization cycle would pick up from almost where it left off and a new minimum could not be found. As a result further conformational updates would either not be performed or they would be too few with a very small lambda value (i.e. $\lambda = 10^{-6}$), which would have a quite insignificant impact on the system's configuration.

The first thought to solve this problem was to perform a "shuffle" technique. This means that

at the beginning of each minimization cycle and before searching for a local minimum, all atoms in the system would move along their force vector trajectories using a certain λ value. Experimenting and fine-tuning with different λ values showed that a good value can be found in the 10^{-3} magnitude. In order to avoid big atom moves, a constraint was placed for the maximum value of displacement. This technique improved the situation, but raised concerns for the overall quality of the system's conformation as a result of the frequent shuffling.

Another possibility was to introduce an uphill move approach. That is accepting a random small uphill step as a transient state to recover the system from a local minimum and explore the potential energy landscapes for other valleys in the hope that a lower energy value can be found. This technique improved the local restoration defect evident in previous experiments. The potential energy surface (PES) exploration would not always result in a lower energy configuration, but the configurations visualized within the interaction area between the two molecules improved significantly. This is due to the fact that exploring the energy landscape this way, triggered a higher number of conformation updates. This in turn caused the atoms within the interaction area to move along their force vectors a number of times and eventually form a stable conformation, with correct bond lengths and angles on both molecules.

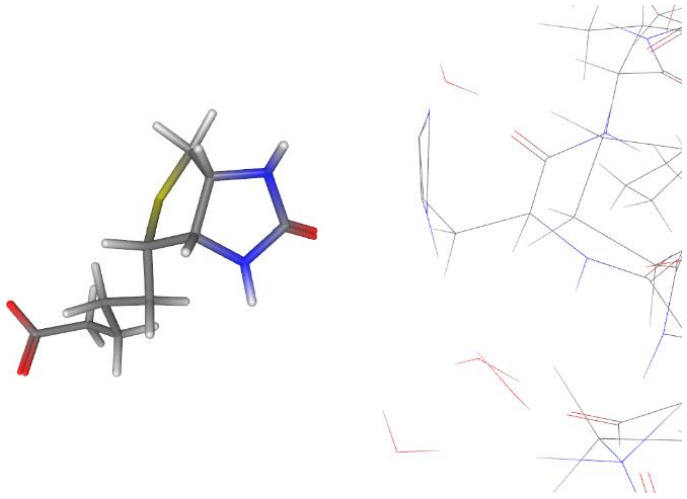


Figure 7-1: A snapshot of a ligand (left) approaching a protein (right) using a greedy local search approach. The algorithm is continuously improving the state of the system after the initial interaction of the two molecules, but settling at erroneous configurations. The atoms of the aromatic ring are not co-planar as they should be.

7.2 Improved Method

The initial experiments, lead us to reformulate our objectives to find a method able to search the potential energy landscape for a good minimum and also restore the area of interaction between the two molecules in a single minimization cycle. The new algorithm should be able to evaluate candidate solutions in an efficient manner. It should also ensure that within each minimization cycle there would be enough conformational updates to restore the area of interaction.

The new implementation uses an evolutionary algorithm to generate a set of candidate solutions to the problem (system conformations). The candidates are evaluated by calculating their potential energy asynchronously using CUDA streams, instead of calculating them serially back to back, which allows for a degree of performance gain as we demonstrate in our results section. The potential energy value of each candidate is used as an indicator of fitness. The fit candidate is the one whose potential energy value is lower than the latest recorded potential energy minimum (old minimum). If no child solution has a value lower than the old minimum, then the fittest child is chosen with criteria set by our uphill move strategy. The fittest child becomes the parent of the new solutions generation and the system's configuration is updated adopting its co-ordinates. Finally after the conformational update, the force vectors for the newly updated system are calculated. The process of mutation, candidates' evaluation, choice of fittest, conformational update and calculation of force gradients is called a move.

7.3 Fast Surface Exploration and Induced Local Restoration

Initially our algorithm will find its first valley and aim downhill until it reaches a minimum. When it does, the parent solution will be unable to generate a population with a candidate solution whose energy value is less than the old minimum. At this point a strategy for escaping the local minima is needed, which would be to immediately try a different valley in hope of finding a better minimum and induce stable conformational updates at the same time. The perturbation strategy of ILS (Iterated Local Search) is ideal for quick jumping from one valley to another by accepting an uphill move when the system lies at a local minima [84 - 85]. This

is achieved by choosing a scalar value λ able to perform a shuffle operation in a similar way as we explained before.

ILS initially gave promising results, however, there were two main problems with it. The first is that it would occasionally show a visual pulsating effect, where atoms would seem to perform a fast oscillation. That was because the time-lapse of the algorithm would expire immediately after a perturbation and the system configuration exiting the algorithm would not be optimal. This problem could be easily solved with a ghost particles buffer holding atom co-ordinates after a perturbation move. The post-perturbation generations would inherit and update co-ordinates on the ghost buffer up until a new minimum is reached lowest to the one associated with the real co-ordinates array. This solved the pulsating visual effect, but initiated the second problem that we referred to where ILS would often find a good minimum within the first few moves, and the background search following the last visited minimum would not be able to find a better solution. This would lead into the aforementioned problems of too few conformational updates and hence inefficient minimization cycles.

Guided local search is ideal for inducing updates on the principle that the old minimum is being raised by a small percentage at every solution evaluation (penalty function) [67]. Eventually, the algorithm can escape from the well, but will spend some time in it waiting for its penalty function to reach the appropriate levels. Experiments showed that this method underperformed compared to ILS for our simulations as shown in the results section (Section 7.5).

Our final solution came as an amalgamation of the two approaches. That is, we introduced a penalty value *p* and an energy function *E*, which is updated at every iteration (energy evaluation) *i*, as described in Equation 7-1. The penalty value is usually in the magnitude range of 0.001 – 0.01.

$$E(i+1) = E(i) \cdot (1+p)$$

Equation 7-1 the penalty function

This means that on every iteration the old minimum bar is slightly raised so that it can allow

for an update of the system's real co-ordinates. This modification aims to keep the fast track of valley exploration that ILS can offer and introduce enhanced conformational update abilities to our approach. We will refer to this algorithm as IGLS (Iterated Guided Local Search). In essence this algorithm achieves the following goals. It accepts updates for energy values close to a recently visited lowest energy configuration, for subsequent solution populations. Failing this, when subsequent solution generations do not produce an update, it prevents the algorithm from wasting too many generations without performing an update, which in turn benefits the visual effect of our application as well as the interacting molecules configuration, especially within their interaction zone.

The implementation of the above technique is performed by using two energy variables, the current_lowest and the global_lowest. Every time a new minimum is reached, both the current_lowest and the global_lowest take its value as demonstrated in pseudocode 7-1. When a local minimum is reached and the new generation cannot provide a candidate with a lower energy value, perturbation takes place for a new well to be explored and the current lowest takes the corresponding energy value. While in the new valley, the global_lowest value is raised at every energy evaluation by a small factor (i.e. 0.5%), while the current_lowest keeps reaching lower values, until we reach the new local minimum.

Now at this point there are two possible outcomes. The first is that the new local minimum (current_lowest), is lower than the global_lowest and a conformation update is being performed. Otherwise, a new perturbation move is being performed. However, the probability of finding a new minimum this way is directly proportional to the number of energy evaluations performed. This way, a bigger ratio of contributing moves compared to the total number of moves is achieved, which in turn assists into solving the local restoration problem. Figure 7-2 illustrates how each of the four methods in comparison performs in a single minimization circle comprising of 8 steps for a protein already minimized using the SD approach..

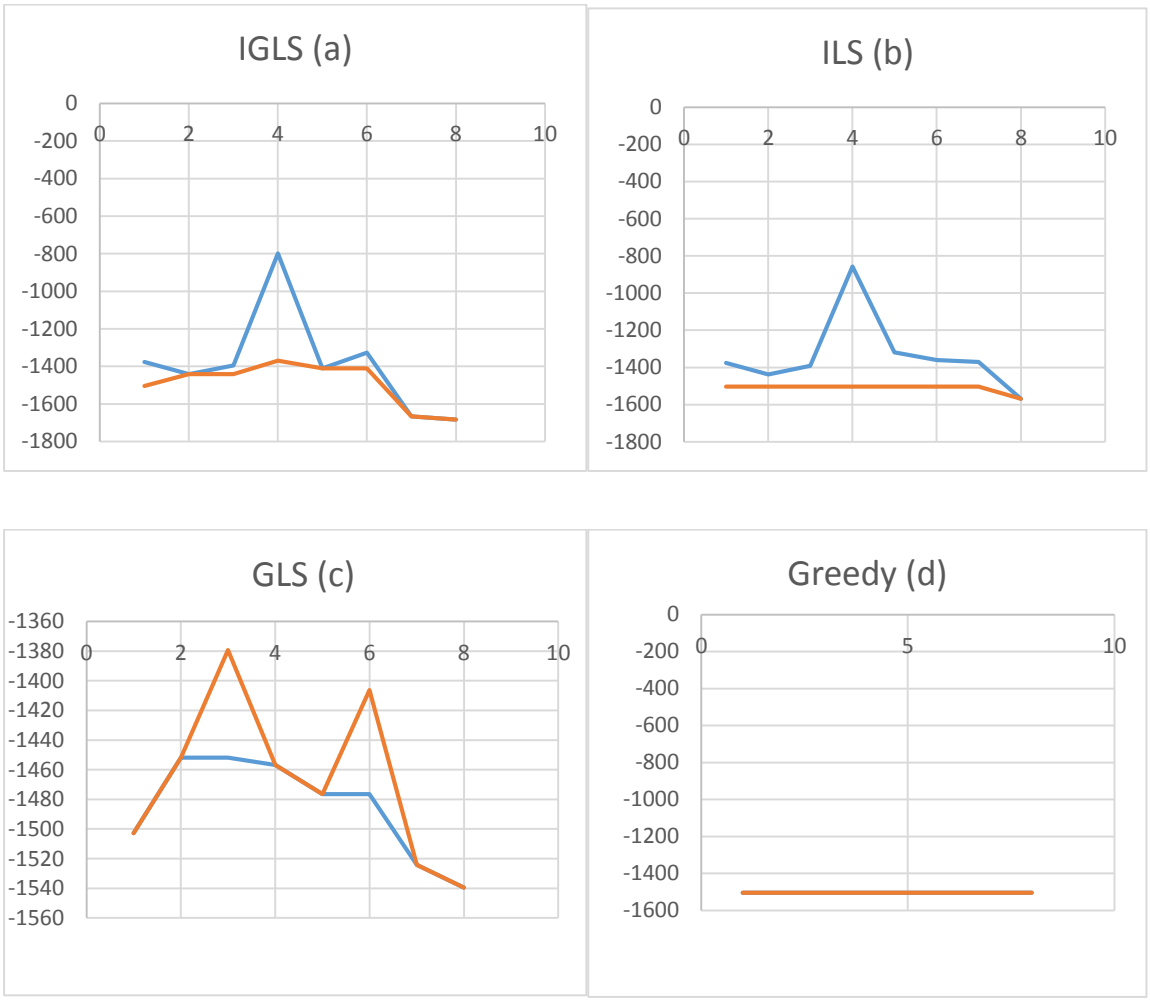


Figure 7-2 a-d. A schematic overview figure of the four algorithms compared in this report, depicting how a calculation circle of t=33ms is subdivided into 8 steps of a group of energy kernel evaluation streams for the 1UGM.pdb protein file. The system is already minimized with a local search method prior to running the experiment. The blue curve depicts the *current_lowest* and the red one the *global_lowest*.

This algorithm allows for quick exploration of molecular energy surfaces, by encouraging more conformational updates than a purely ILS strategy would perform. The visual result is improved by a large factor as there is rapid local restoration on the impact area. However, the fact that this hybrid approach induces extra conformational updates generates an important ambiguity. That is whether there is a systematic sacrifice of the final energy value due to the uphill move strategy adopted. This was investigated and discussed in section 7.7 .

7.4 Improvements

The above algorithm can be improved by adding a stopping criterion for the guided local search phase. That is to stop inducing conformational updates after a certain condition has succeeded. For example if the max force value is close to the mean or an adequate number of conformational updates has been performed. This can help us choose the conformation with the energy minimum that we want, such as the lowest we came across after local restoration was achieved, rather than the last visited one. This improvement is only worthwhile for smaller system sizes (<10,000 atoms), where each minimization cycle can include a number of population evaluations.

7.4.1 Asynchronous execution

The performance of the above algorithm is important for our application as the potential energy landscape exploration is constrained to run for a small finite time-lapse *t*. The more moves we are able to perform in the landscape the better the quality of the resulting solution will be. In addition, the more parent solutions the algorithm generates, the higher the probability of discovering lower energy minimums. From the above, we understand that performance is vital for our simulations. Hence we designed our algorithm to take advantage of asynchronous kernel execution using CUDA-streams. On the current generation of graphics cards (GK110), we can evaluate fit of a whole generation of solutions asynchronously and almost in parallel for smaller protein-ligand systems. This means that the first energy evaluation kernel hides the latency of the remaining *s-1* evaluation kernels scheduled to run asynchronously in *s* streams within a single move (a population evaluation series, followed by a force calculation for the fittest child), where s<=16. However, if the conditions mentioned in the background section are not met, then the number of asynchronous streams running concurrently at any time drops as demonstrated in *Figure 7-3*.

In addition, we can hide the latency of binning the atoms into a 3D irregular grid for the cell-list generation under the force gradients kernel. This means that binning overheads are no-longer an issue and binning can be safely performed at every single move, with a small skin value δ' that caters for the biggest displacement that we anticipate during each move (i.e.

Cutoff_list = cut-off + δ')

The present stream-based solution is simple in its implementation and its only challenge is synchronisation between different streams or stream groups in the code. The performance gain using this approach comes on the assumption that on average the stream approach will need less time for child solutions evaluations. In order to exemplify this let's assume for the example case of *Figure 7-3* that we run a 3 step simulation using both approaches. If the serial approach finds a minimum on the second, fifth and eighth energy kernel evaluation, then the serial version will have computed 14 energy kernels (both bonded and non-bonded) in total. The stream approach will have evaluated 42 energy kernels, but will have done so in the time needed to evaluate 9. In addition, the stream approach will pick the best child solution each time, whereas the serial one will choose the first solution to reach a lower energy value. The stream solution can potentially be extended in multiple GPUs especially on the arrival of the Pascal line, where GPU-GPU communication will be achieved directly (and faster) between them and not via a PCIe bus. The multi-GPU option has not been explored in the present implementation as we lacked the equipment.

Figure 7-3 Visual profiling data for a move. Asynchronous execution of s streams where the total block and shared memory conditions are not met. Overlapping is divided into stream groups that meet the hardware's block and shared memory restrictions. The kE blocks represent the non-bonded energy kernels, the dispersed blue blocks are the bonded energy potentials the kF block is the non-bonded forces kernel. The bonded forces and the 3D grid binning kernels are represented by the small blocks below kF.

Pseudo code 7-1: Hybrid algorithm asynchronous for n streams

```
//get initial potential energy and force vectors
Energy_nonBonded<<<stream 0>>>
Energy_bonded<<< stream 1>>>
ForceGradients_nonBonded<<< stream 2>>>
ForceGradients_bonded<<< stream 3>>>

//explore various molecular energy landscapes till time expires
while (time_left)
 //generate solution population for different lambda values
 for i = 0 -> n
      mutate<<<stream(i)>>>(i, lambda[i])

      //calculate energy for each child asynchronously
      for i = 0 -> n
       Energy_nb <<< stream(2i+1)>>>((i, step[i])
            Energy_b <<< stream(2i)>>>(2i, step[i])
 //the fittest child becomes the parent solution
      Choose fittest solution
 //calculate the force gradients
      ForceGradients_nb<<< stream (1)>>>
      ForceGradients_b<<< stream (2)>>>
      //prepare the 3D irregular grid for the parent
 BinToGrid<<< stream (3)>>>
```

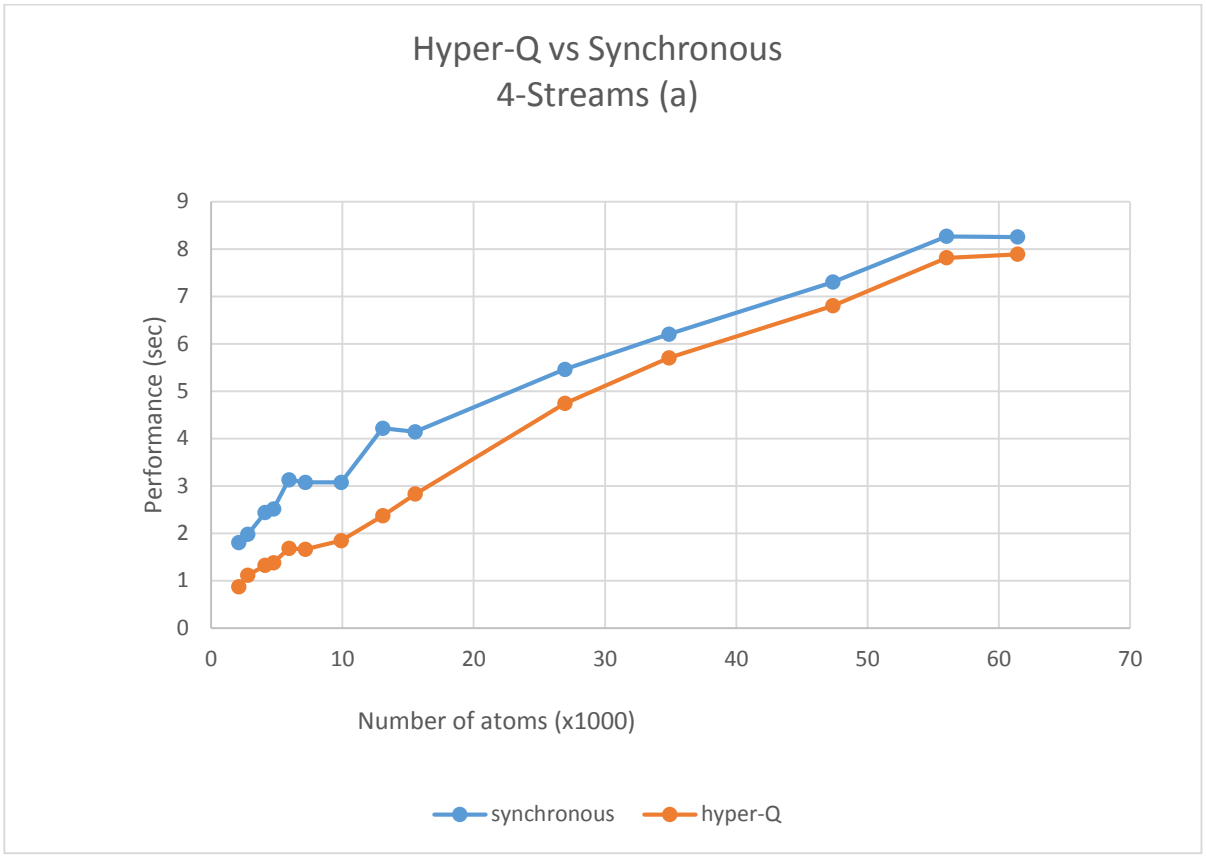
7.5 Results and discussion

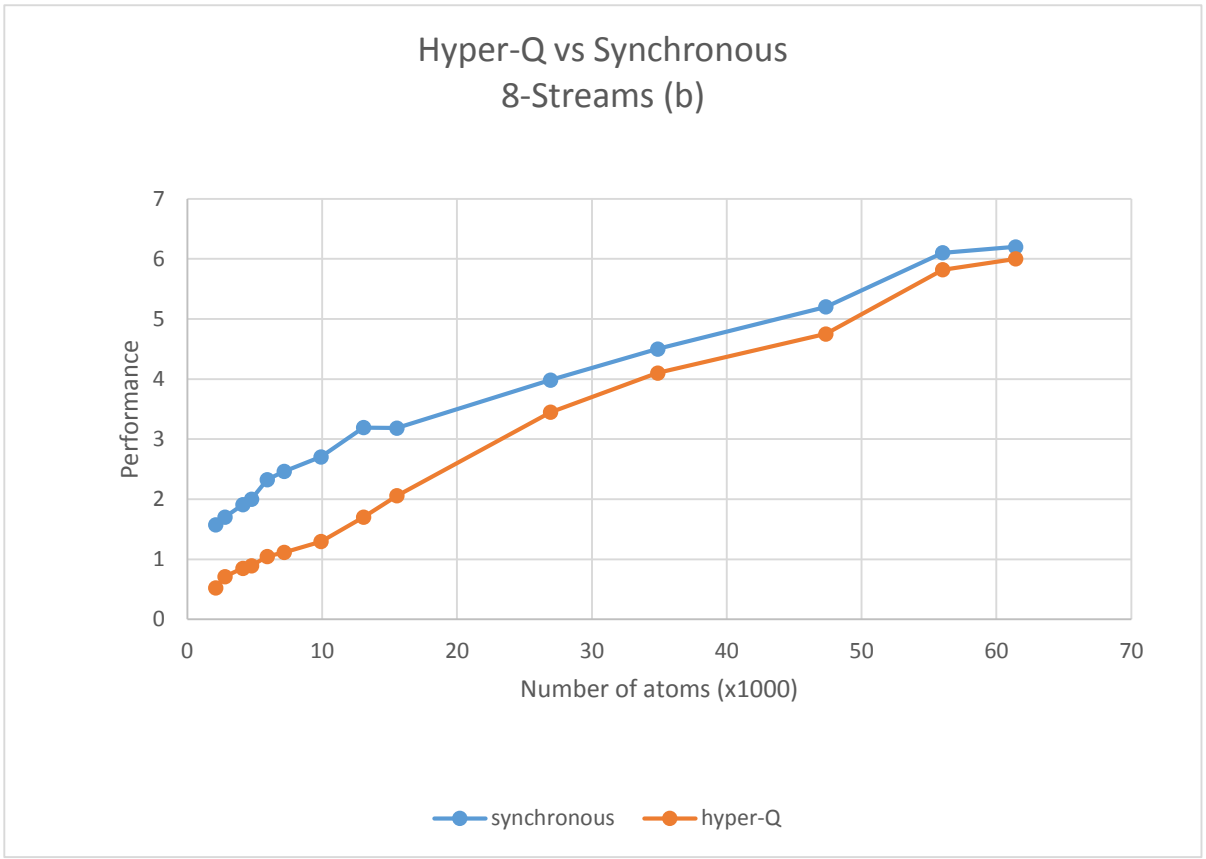
The improved hybrid algorithm's performance results are appended in the following section. The visual result was better as the reviewers of the code could observe that the aberrant structures produced were more accurate than any other previous versions. In order to assess the aberrant conformations, a project student, Gaia Pasqualetto performed qualitative and quantitative studies on the code's ability to simulate protein-ligand interactions for her final year project and a summary of her outcomes is included in Chapter 8.

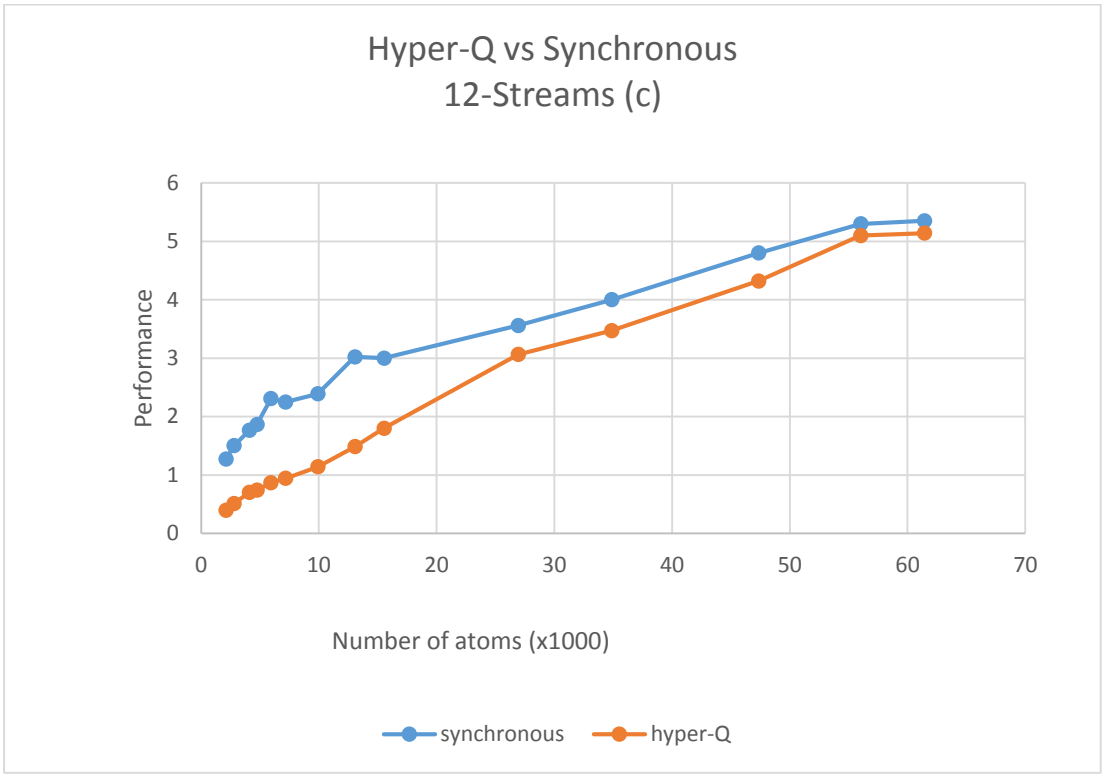
7.6 Performance

The performance of our CUDA-streams implemented hybrid approach has been benchmarked against a synchronous execution version of the above algorithm. In order to perform a fair comparison, benchmarks were scheduled such that both versions would perform the exact same amount of energy and force gradient kernel calls. The results are listed in Figure 7-4 a-d for 4—16 concurrent streams. From the Figures we can observe that there are performance gains fluctuating from 1.2X - 4X depending solely on the system size and number of streams. Regarding shared memory usage, both our non-bonded energy and force gradient kernels are using 9Kb whereas the bonded energy and forces kernels do not use shared memory. This means that our concurrent execution on energy evaluations at any time is restricted to five as a sixth one would exceed the 48kB shared memory per SM limit.

Our algorithm achieves five concurrent energy evaluations for system sizes up to 10,000 atoms. This is because the energy kernels called, divide execution into blocks of 256 threads, where each thread is associated with one atom. This equates into a maximum of 40 blocks per stream, which is good enough to saturate the limit of five concurrent streams given from shared memory restrictions (5 x 40 < 208 concurrent blocks limit). Beyond that limit, the number of blocks needed to process the energy kernels rises above 208 and that means that CUDA cannot handle the total number of blocks scheduled for asynchronous executions, hence concurrency efficiency starts deteriorating. Finally for systems above 50,000 atoms, concurrency almost vanishes on the energy kernels. This is because the number of blocks needed to process one system alone is close to the architecture's limit. However, concurrency in the forces-binning stream pair still holds as our binning method has been designed using a series of fast executing kernels able to run using only a few blocks. In addition, bonded forces kernels require fewer thread blocks and hence can contribute towards some asynchronous execution too. This is why even in the case of large systems (50,000 – 65,000 atoms), there is still a small performance gain.


Hyper-Q vs Synchronous
4-Streams (a)
Performance (sec)
0
1
2
3
4
5
6
7
8
9
0
10
20
30
40
50
60
70
Number of atoms (x1000)
synchronous
hyper-Q


Hyper-Q vs Synchronous
8-Streams (b)
Performance
0
1
2
3
4
5
6
7
0
10
20
30
40
50
60
70
Number of atoms (x1000)
synchronous
hyper-Q



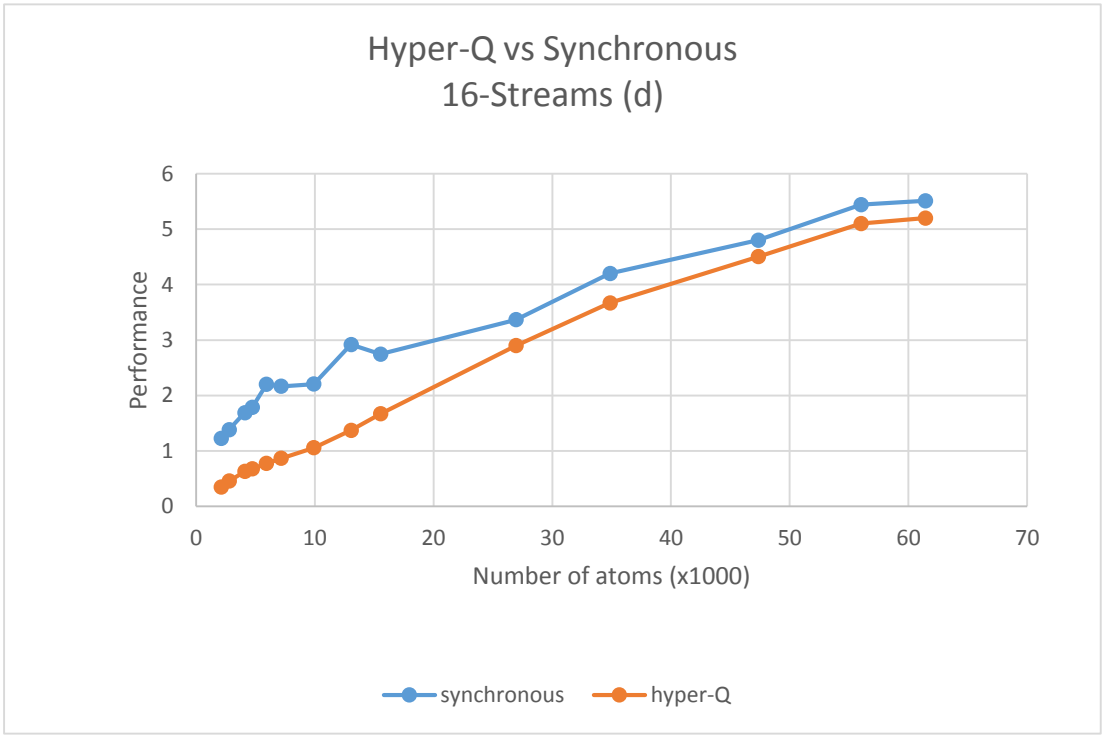


Figure 7-4 a-d Performance benchmarks between concurrent and serial stream versions. The same amount of energy kernels (1000) and Force gradient kernels (1000/nstreams) are evaluated from both approaches in each graph.

Tables 7-1 a and b demonstrate a comparison of the elapsed time of execution for the calculations demonstrated in ***Figure 7-4*** between CUDA-synchronous and CUDA-asynchronous executions. In addition, it appends execution times for the serial implementation on a workstation equipped with a Tesla K-20 GPGPU card and Intel Xeon E5-4620 CPUs running at 2.20 GHz. The serial execution uses the OpenBabel library for the MMFF94s calculations.

4-Streams (a)

Proteins	K atoms	synchronous	hyper-Q	Serial
1UGM.pdb	2.103	1.8	0.87	127.5
3RDD.pdb	2.786	1.976	1.115	235
3F9E.pdb	4.74	2.508	1.378	647.5
3FAU.pdb	5.917	3.126	1.677	1017.5
2O0O.pdb	7.165	3.074	1.662	1447.5
3O05.pdb	13.053	4.216	2.368	7430
2EAR.pdb	15.528	4.14	2.826	7450
1TBG.pdb	26.927	5.46	4.74	32277.5

12-Streams (b)

Protein	K atoms	synchronous	hyper-Q	serial
1UGM.pdb	2.103	1.272	0.393	89.16667
3RDD.pdb	2.786	1.502	0.511	165
3F9E.pdb	4.74	1.862	0.74	455.8333
3FAU.pdb	5.917	2.307	0.868	719.1667
2O0O.pdb	7.165	2.248	0.941	1015.833
3O05.pdb	13.053	3.022	1.486	5870
2EAR.pdb	15.528	2.999	1.8	5423.333
1TBG.pdb	26.927	3.559	3.062	25385.83

*Table 7-1 a-b:*Results for CUDA-synchronous, CUDA-asynchronous and serial executions.

7.7 Heuristic ability of our approach

It is important to also evaluate our approach in respect to its heuristic abilities. In other words, its ability to reach good local minima, considering the fact that we encourage it to take a number of guided uphill moves at the start. For this reason a set of experiments was designed to show the fluctuation of energy values using different algorithmic approaches including our hybrid method. In order to perform this evaluation, we ran four protein-ligand simulations using a different protein each time. For each simulation, the ligand was induced into a protein pocket and the resulting conformation was saved. The above simulations were performed using the simple local search method that fails to restore the proteins structure well. This way, the different approaches would be evaluated against our problem situation which is to find a good minimum, from an erroneous initial structure in a minimum amount of minimization steps.

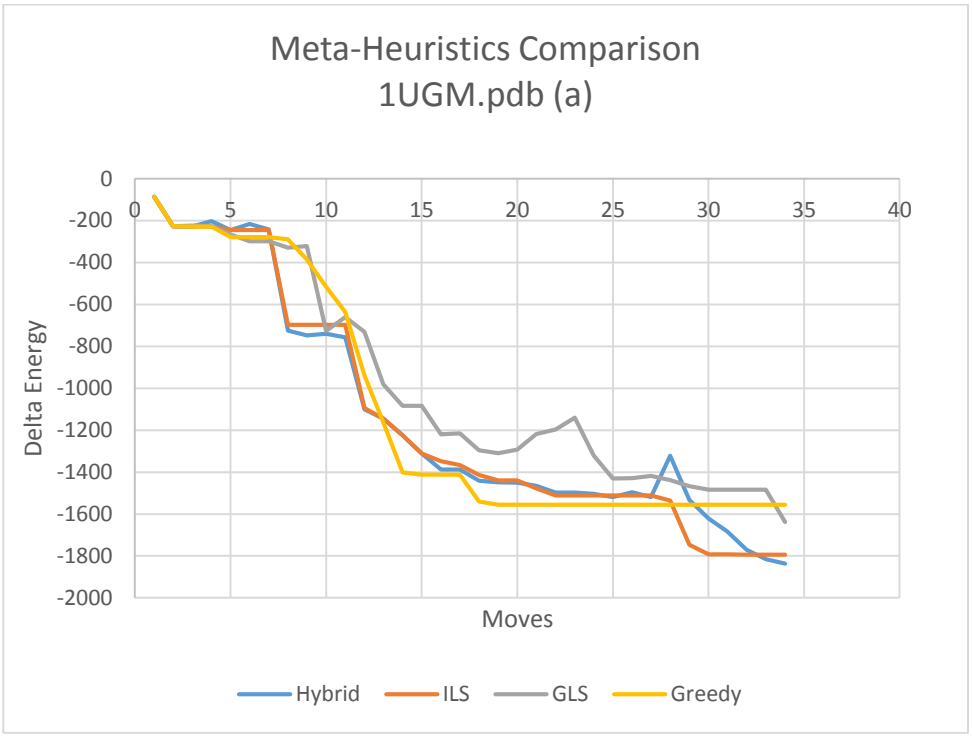
In the simulation results presented in Figure 7-5 a and c we can see that the hybrid method and the ILS clearly outperform the GLS and the greedy approach. It is also interesting to see that in all four Figures the curves of the hybrid and ILS methods having a similar trend, with the hybrid one following a more turbulent trajectory due to its GLS element. Figure 7-5 b, is an interesting case as we can see that ILS is trapped at a minimum from which it cannot escape. From Table 7-22 we can see that for the experiment with 3F9E.pdb, ILS could only reach four minima and hence perform four updates. This explains why this method cannot perform well on our simulations, as this case can happen quite often. Four updates are usually not enough to bring the system back to a stable conformation nor to restore the conformation of the interaction area between the two molecules. Our method on the other hand, using the GLS element, can escape such a minimum and carries on exploring different energy landscapes, finding new minimums and updating the system into better conformations. Regarding GLS on its own, as we can see it always underperforms compared to the hybrid method and it only performs better than the ILS at the 3F9E experiment, where ILS gets stuck at a well.

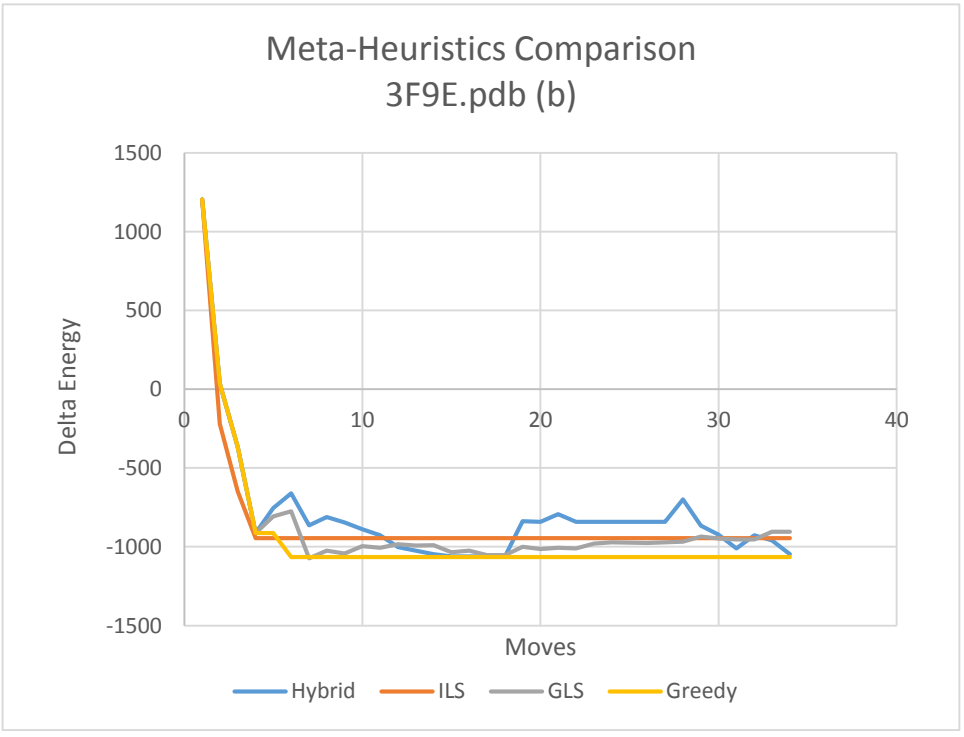
Looking at the results in Figure 7-5 and Table 7-22, it is evident that the hybrid approach is the most suitable for the type of simulations that our software performs for its ability to find a

very good potential energy minimum by inducing more conformational updates than it would following a pure ILS strategy. Also, the GLS element in it does not really mean that the resulting global minimum would be lower than the corresponding ILS nor the opposite. The hybrid method seems promising in solving other optimization problems too, due to its ability to change over different energy landscapes and evaluate lots of different minima at a small temporary potential energy cost.

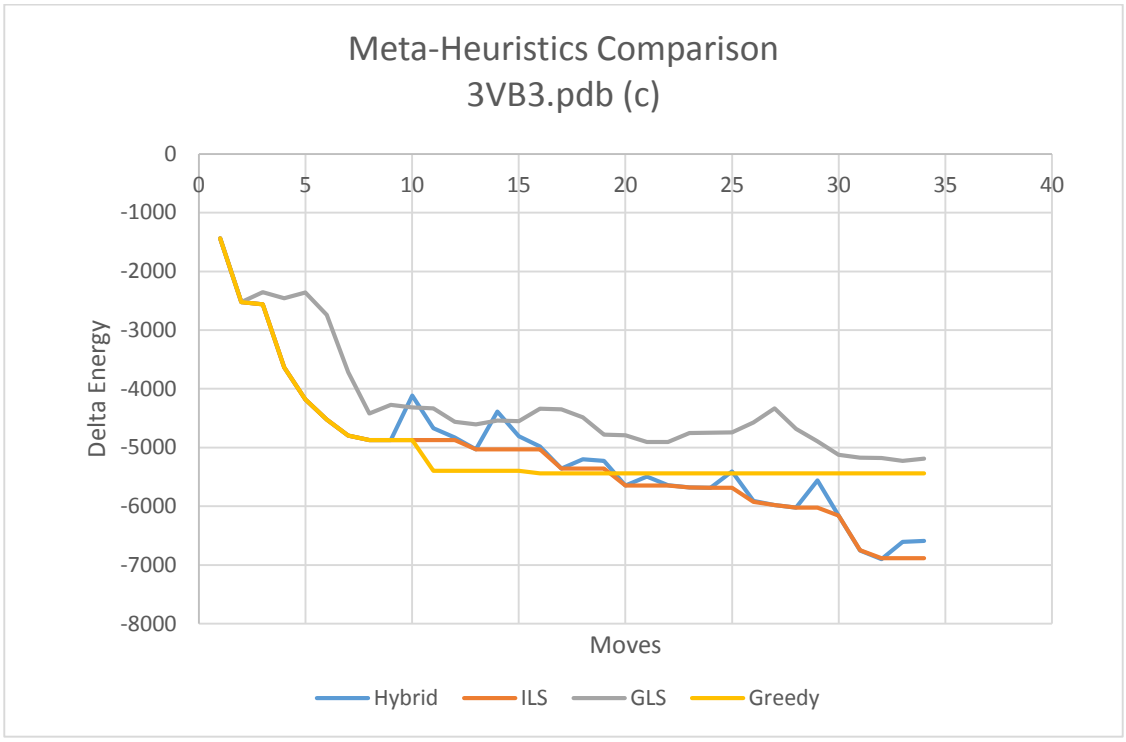
	Hybrid	ILS	GLS	Greedy
1UGM.pdb	27	24	20	15
3VB3.pdb	26	20	22	11
3F9E.pdb	18	4	17	5
3O05.pdb	26	20	18	10

Table 7-2: Number of conformational updates for each experiment for the four different algorithmic approaches.


Meta-Heuristics Comparison
1UGM.pdb (a)
Delta Energy
0
-200
-400
-600
-800
-1000
-1200
-1400
-1600
-1800
-2000
0
5
10
15
20
25
30
35
40
Moves
Hybrid
ILS
GLS
Greedy


Meta-Heuristics Comparison
3F9E.pdb (b)
Delta Energy
1500
1000
500
0
-500
-1000
-1500
0
10
20
30
40
Moves
Hybrid
ILS
GLS
Greedy

`



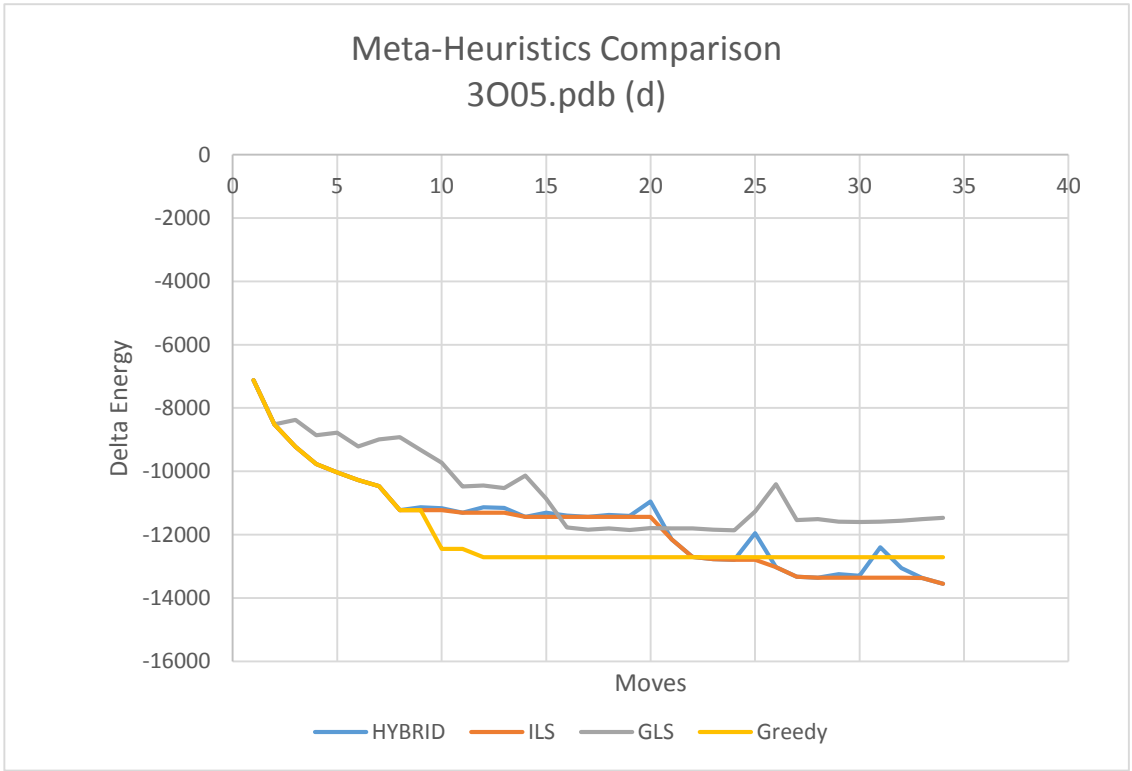


Figure 7-5 a-d: Comparison of the 4 different approaches (lower equals better score).

7.8 Summary

In this chapter we demonstrated a hybrid evolutionary strategy of exploring molecular energy landscapes in a small elapsed time period. Our algorithm has been tuned to take advantage of the extra parallelism that asynchronous CUDA streams can provide. We explained the reasons behind opting for energy landscape exploration rather than adopting a simple local search method exiting at the first local minimum. The proposed method works well for conformational sampling in situations where the exit criterion for the meta-heuristic is the expiration of a very small time-lapse.

Discussion of Results

8.1 Evaluation of results

Looking at the results reported in the previous two chapters, we will now evaluate to what extent the objectives set in Section 0 are met. To be more specific, regarding objective number one:

"Design fast and accurate algorithms for the computation of MMFF94s force-field, able to complete a series of energy and force calculation functions within the 33 ms time-limit."

The related results are presented in Figure 6-7. The performance results of our irregular grid approach on the previous generation card GTX 680 (the current equivalent is GTX 780) is able to perform force calculations in less than 4 ms and energy calculations in less than 2.5 ms for a system containing 30 000 atoms. This stands at the top of the size range of systems we aim to simulate with Zodiac.

Regarding the second objective:

"Design efficient energy minimization algorithms able to generate accurate conformations for systems up to 30,000 atoms within the 33Hz requirement."

The results are presented in Table 8-1. This table shows how many steps can be performed during a 33ms period (minimization cycle), which is the time distance between two successive protein-ligand poses, using the IGLS technique developed in this project. This table shows that the number of moves is a factor of how many concurrent streams are employed and system

size. The number of steps completed per 33ms period could be treated as an index to how the user can interact with the software.

For smaller systems of up to 10,000 atoms, the software is able to complete 4-5 minimization cycles within a 33ms period, which in turn ensures frequent conformation updating and protein-ligand interactions are mapped on the screen in interactive time. However for bigger systems, the user will have to use the haptic slower. This means that a stable conformation can be achieved in 2-4 minimization cycles depending on the system size.

At this point it makes sense to wonder how many minimization circles needed for a system in order to ensure quality conformations during a simulation. An answer can be attempted by looking at Figure 7-5 and proteins 1UGM and 3F9E at the specific configuration and pose against the BTN_Model ligand that these results were recorded. In this figure, we can observe that our algorithm (as well as the other approaches tested) can reach a good energy minimum in significantly fewer steps for the 3F9E system.

This explains that there is not an absolute cut-off steps value to guarantee conformation quality as the number of steps needed for a good conformation varies between systems and protein-ligand poses. It is also evident that the smaller the system, the more minimization steps we can complete per minimization circle and the faster our algorithm can find an optimal conformation. However, we can here give a cut-off value of minimization steps needed in order to be able to provide simulations and this value is one. We can also suggest that with the current state of the software, the user needs to be able to understand how fast the algorithm discovers stable conformations for the protein-ligand system and adjust the speed at which he/she moves the ligand inside the pocket accordingly.

The above results have been recorded with a Tesla k20 card. This means that similar performance can only be achieved with cards supporting Hyper-Q such as the GTX 780 (GK 110 chipset). For older generation cards such as the GTX 680, the results will be slightly inferior as asynchronous kernel execution is not as efficient in GK104 chipsets.

pdb file	Atoms	Streams: 4	Streams: 8	Streams: 12
1UGM.pdb	2,103	11	11	9
3RDD.pdb	2,786	11	9	7
3FS1.pdb	4,100	9	7	5
3F9E.pdb	4,740	9	6	5
3FAU.pdb	5,917	6	5	4
2O0O.pdb	7,165	6	4	3
3VB3.pdb	9,903	5	4	3
3O05.pdb	13,053	4	3	2
2EAR.pdb	15,528	3	3	2
1TBG.pdb	26,927	2	2	1

Table 8-1: Number of minimisation cycles completed per 33ms period.

As to whether the second objective has been addressed by the outcomes of this thesis, the answer cannot be a simple yes or a no. We can conclude that for systems up to 10,000 atoms, the application performs well, able to serve 160-200 steps per second and produce good quality updates at the rate of 33Hz. For bigger systems, the application needs the help of the user and requires a slightly slower rate of ligand move in the protein cavity.

However, this is an answer from a programmer's point of view. In reality, this application should be used such that the experimenter/user would try to stabilize the ligand against a protein cavity, which would give him/her time to observe the reactions and give time to understand the direction of the forces felt. Taking this into consideration, the software is usable for systems in the top size range.

Another factor that needs to be taken into consideration is the constant evolution of graphics cards. Future generations are expected to have greater FLOP capabilities. Moreover, it is always possible that new cards can accommodate more warp schedulers enhancing parallelism, benefiting more from high multiprocessor occupancies or even improve Hyper-Q performance by allowing more concurrent thread blocks to be processed. The reported algorithms have been designed with the above facts in mind. The vast majority of computations are floating point operations, with integer arithmetic avoided as much as

possible, so that future generation cards would achieve better results according to NVIDIA's published FLOPS road map (Figure 1-1). The reason that integer arithmetic is avoided is because it has slower speed of execution, which is 32 multiply instructions or 160 add/subtract ones compared to 192 of 32-bit FLOPS for both multiplications and additions/subtractions.

Regarding our research hypothesis:

"Is it possible to design algorithms for the latest CUDA architecture that would perform such that an HPLD simulation could run in Zodiac with accurate conformational updates modelled with the MMFF94s force-field for systems up to 30,000 atoms, matching the visual rendering rate of 33 Hz?"

We can provide a positive answer. The application works as a fully flexible HPLD application and there are already researchers in the molecular modelling lab working on it with systems up to 16,000 atoms on GTX 680 cards. In addition, there has been another researcher that used the software with the previous version of the minimization algorithm (Steepest Descents) in the past with smaller system sizes.

8.2 Limitations of our approach

There are some limitations on the efficiency of our application at the moment. The first and foremost is that it does not take into consideration the long range electrostatics. As explained in Chapter 3, the cut-off convention provides a very close approximation for the VDW interactions, but regarding the electrostatics the approximation is less accurate and it is recommended for MD applications to include the long range electrostatics for more accurate conformation sampling.

8.3 Future research recommendations

In future releases, the application could benefit from the implementation of long range electrostatics. A CUDA algorithm can be designed using the PME method similar to that in Harvey 2009 [25]. The resulting CUDA algorithm can be used as a stream together with the

rest of the concurrent streams inside the hybrid energy minimization algorithm. Considering that this algorithm's complexity is O(*NLogN*) it would be expected that a full force range version (short plus long range) should be able to run well on smaller system sizes. The benefit of such an approach would be enhanced conformational quality and hence enhanced clarity on research outcomes.

A further improvement of the designed methodology would be to improve the graphical representation of the software. At the moment the only graphics mode that can be supported with the flexible approach is that of a wire model. However, space filling models (CPK) [86] are superior for protein cavity visualization and give a better understanding of the molecular system's space to the human eye. CPK surfaces can be rendered using marching cubes algorithms[87-89]. Marching cubes can also be implemented in CUDA showing very good performance gains in parallel compared to serial versions as in Johanson 2006 [90]. Again this algorithm can be programmed as a CUDA stream running concurrently in the same stream group as the Forces kernels. Its impact on performance will be minimal, as it will only need to be present once every interaction cycle together and returned from the minimization method along with the final co-ordinates.

Another improvement, would be the use of multiple GPGPUs for the simulation of larger systems. This would mean that on the concurrent energy kernels evaluation, kernels can be assigned to multiple GPGPUs, which in turn can improve the performance of population evaluation of the evolutionary part of the algorithm. That can also have substantial benefits on a potential version with a PME kernel for long range electrostatics. The forthcoming line of GPGPUs (Pascal), is expected to provide faster GPU-GPU communication on multi-GPU systems (5x faster than the present PCIe data transfer speed).

The software can further improve by implementing an improved force feedback technique on the haptic for the flexible mode. As a matter of fact the force feedback available now is at a rate of 33Hz. However, a better force experience is fed to the user at 1 kHz updates. A sensible solution would be to interpolate feedback between two cycles and feed it back to the user. However implementing this in real time would require intelligent and fast heuristics in order to guess the magnitude and direction of the feedback on the next circle. An easy way to

implement this could be to delay force feedback for 33Hz, which is going to be unnoticeable by the user because this time-lapse is small enough for the human to detect. This way feedback between two minimization cycles can be easily interpolated and fed back to the user.

8.4 Practical applications of the algorithms developed in this project

The code developed for the purposes of this project has been incorporated in Zodiac user interface and several medicinal chemistry students have tried to produce some docking results in order to evaluate the efficiency and accuracy of this project's efforts. The first attempts were not very promising as they were carried out before the evolutionary approach was developed and in non-complete versions of the force-field calculations. The last efforts made by project student Gaia Pasqualetto who used this code for her project thesis, were very promising and submitted for publication along with the outcomes of Chapter 7 in the Faraday Discussion May 2014 issue. The code produced very accurate aberrant conformations for protein-ligand systems requiring a small degree of flexibility during the interaction of a ligand in the protein's cavity. For systems requiring a very high degree of flexibility during binding such as the HIV reverse transcriptase case, the approach shown a good potential by producing aberrant conformations on the right trajectory, but the need for producing a larger set of candidate conformations was evident to produce a perfect end result. The evaluation was carried out on a GTX680 GPU card, which does not support the Hyper-Q functionality and we believe that the results could have been substantially better on a GK110 chip GPU such as the GTX780. A more detailed report on the outcomes of the algorithm evaluation composed by G. Pasqualetto can be found in Appendix 1 at the end of this thesis.

8.5 Summary

This chapter provides a critical evaluation of our aims and objectives set in Section 1.7 as well as an answer to our research question. Our first objective was met as demonstrated from results provided in Chapter 6. The second objective was met as demonstrated with the performance quality of our application being higher for smaller systems. The application is also able to model systems in the range of 15,000 -30,000 atoms, but conformations need 2-3 minimization cycles to converge into a stable state. We also listed recommendations for

future research. An implementation of PME for the long range electrostatic forces as an asynchronous stream has been proposed in order to provide higher conformation sampling accuracy to the application at the expense of a lower system size range. In addition the graphical display can be enhanced by CPK surface rendering with the aid of a well-designed Cuda-parallel implementation of the marching cubes algorithm. This implementation can run as an asynchronous stream on every update together with the forces and binning kernels.

Conclusions

9.1 Algorithm usability in Zodiac

This thesis describes the efforts made within a three year period to transform Zodiac, from a rigid protein, semi-flexible ligand HPLD application to a fully-flexible simulation. The conformation sampling that Zodiac is able to perform and the delta energy results it is capable of producing are a very close approximation to what MMFF94s can model. The level of approximation depends on the cut-off distance set by the user. Another parameter that the user can adjust is the cells aspect ratio by setting the initial cell base width to the desired value as explained in section 6.1.1. Our recommended optimum value is 13.75 Angstrom, but the user is free to test the system with different values and find the best one for each candidate system. Finally, the user can select the desired number of asynchronous streams that he/she wants for each simulation, in the range of 4-16. Four streams are the minimum value in order to cover a good range of values for lambda and 16 is the maximum number of streams that our conformations sampling algorithm can run asynchronously.

9.2 Contributions

Our contribution to knowledge from this thesis can be summarized as follows. In Chapter 6 we provide a solution for the non-bonded interactions computations that maintains the benefits of coarse grained kernel executions. That is, high occupancy levels and fewer thread blocks, which enables us to run more computational kernels asynchronously in GK110 chipsets. Our algorithms contributed a solution to the load balancing problem, by providing the structure of an irregular grid with adjustable cell aspect ratios. The irregular grid is

designed from a rectangular grid of adjustable base lengths and variable height, followed by a step of atom migration, where atoms move along the X and Y axis such that each column of the grid can be divided in cells of 256 atoms.

The adjustable cell aspect ratios were designed on the hypothesis that the more compact the blocks and hence the warp-groups aspect ratios, the more efficient the computational kernels become and that is proved by the results shown in Figure 6-8 and Figure 6-9. This observation was generated after reviewing the Eastman and Pande 2010 article [51], where the authors were decomposing the molecules space in an irregular grid of randomly shaped cells of 32 atoms with potential overlap, grouping atoms sequentially by their index in the pdb file. Our hypothesis then was whether a new approach, performing 32x32 atom calculations in a more spatially efficient way could be designed.

Our algorithm improves the above approach, by providing a methodology that can perform the same calculation with fewer 32x32 atom interactions, resulting from the improved spatial efficiency. In addition, our approach can exploit the benefits of higher occupancies, better Hyper-Q utilization and enhanced memory efficiency through our nested list approach stemming from cell decomposition into eight non-overlapping 32 atom groups.

A further contribution to the existing literature is our combined way of providing the bonded forces calculations and 1-4 atom exclusions. Our decomposition approach is able to calculate with a single kernel all the bonded force interactions in addition to subtracting the 1-4 electrostatic potential. This is very important as it becomes easier to hide these kernels' latency with asynchronous stream execution. In addition, this approach is the most memory efficient one to our knowledge up to the time of submission of this thesis, as it only loads atom co-ordinates and parameters once for each bonded pair in positions 0-1 of Figure 6-4.

An additional contribution of our approach is the exclusions handling method. Our implementation avoids the construction of NxN matrices for exclusion handling as in Friedrichs 2009 [53]. This is achieved by subtracting the scaled 1-4 electrostatics in the bonded forces kernels and using a bitmap for each atom storing information of its hydrogen bonding properties as well as the index difference of each non-hydrogen bonded atoms. This way, with

only a few bit-shift and bit-mask operations we identify whether two atoms are bonded to each other or a common atom, hence they are excluded. This way we provide a memory efficient method for 1-2 and 1-3 excluded atom pair identification as well as 1-4 atom pairs electrostatic scaling. The above contributions have been published in an article titled “GPU Accelerated Molecular Mechanics Computations” published in the Journal of Computational Chemistry [77].

The algorithm described in Chapter 7 for conformation sampling has several contributions to existing literature. First and foremost it is the first occurrence of the combination of the ILS and GLS methods into a single hybrid approach. Our results proved that the IGLS performs similarly to ILS and more efficiently than GLS. Its main characteristic is that it arrives at a similar result to ILS, but with the ability to sample results from valleys shallower to the last visited deepest ones. This helped our application to provide faster restoration to the impact of inducing the ligand near a protein. Probing the ligand near the protein, results in high forces between the two molecules, which in turn have to be simulated ideally within one minimization cycle. Wasting algorithmic steps in order to find the optimum value results in slow restoration and thus future ligand movements are based on erroneous conformation around the impact zone. The IGLS approach provided a very robust solution to this problem. The above contributions have been submitted for submission at the Faraday Discussion 169: “Molecular Simulations and Visualization”

Finally, our application is the first to perform fully flexible real time HPLD up to the point of this thesis submission and to the best of our knowledge. Our recent research with the state of the art of HPLD applications provided in Ricci *et.al* [7], proves that Zodiac is the first application able to provide real time HPLD with fully flexible conformation sampling with cut-off accuracy. Also, our GPGPU implementation is the first to utilize the MMFF94s force-field.

9.3 Concluding remarks

This document listed methodologies and algorithms designed for the CUDA architecture for the solution of the problem of real-time conformation sampling during protein simulations in Zodiac. These algorithms are able to run on Kepler generation cards only and perform best on

GK110 chipsets and beyond as they support the Hyper-Q feature, which our conformational sampling algorithm is designed to take advantage. These algorithms are available in the new version of Zodiac, which is available from the molecular modelling lab of Cardiff School of Pharmacy and Pharmaceutical Sciences.

Bibliography

[1] S. Plimpton and B. Hendrickson, “Parallel Molecular Dynamics Algorithms for Simulation of Molecular Systems,” in *Parallel Computing in Computational Chemistry*, vol. 592, T. G. Mattson, Ed. Washington, DC: American Chemical Society, 1995, pp. 114–132.

[2] M. Snir, “A Note on N-Body Computations with Cutoffs,” *Theory Comput. Syst.*, vol. 37, no. 2, Jan. 2004.

[3] K. J. Bowers, R. O. Dror, and D. E. Shaw, “The midpoint method for parallelization of particle simulations,” *J. Chem. Phys.*, vol. 124, no. 18, p. 184109, May 2006.

[4] M. Pütz and A. Kolb, “Optimization techniques for parallel molecular dynamics using domain decomposition,” *Comput. Phys. Commun.*, vol. 113, no. 2–3, pp. 145–167, Oct. 1998.

[5] V. A. Voelz, G. R. Bowman, K. Beauchamp, and V. S. Pande, “Molecular simulation of ab initio protein folding for a millisecond folder NTL9(1-39),” *J. Am. Chem. Soc.*, vol. 132, no. 5, pp. 1526–1528, Feb. 2010.

[6] A. Delévacq, P. Delisle, M. Gravel, and M. Krajecki, “Parallel Ant Colony Optimization on Graphics Processing Units,” *J. Parallel Distrib. Comput.*, vol. 73, no. 1, pp. 52–61, Jan. 2013.

[7] A. Ricci, A. Anthopoulos, A. Massarotti, I. Grimstead, and A. Brancale, “Haptic-driven applications to molecular modeling: state-of-the-art and perspectives,” *Future Med. Chem.*, vol. 4, no. 10, pp. 1219–1228, Jun. 2012.

[8] V. Hayward, O. R. Astley, M. Cruz-Hernandez, D. Grant, and G. Robles-De-La-Torre, “H aptic interfaces and devices,” *Sens. Rev.*, vol. 24, no. 1, pp. 16–29, 2004.

[9] M. Ouh-young, M. Pique, J. Hughes, N. Srinivasan, and F. P. Brooks, “Using a manipulator for force display in molecular docking,” *Robotics and Automation*, vol.3, pp. 1824–1829, Apr 1988.

[10] Zonta N, Brancale A, Grimstead I J, and Avis N J, “Accessible haptic technology for drug design applications,” *J. Mol. Model.*, vol. 15, no. 2, pp. 193–196, Feb. 2009.

[11] J. E. Stone, J. Gullingsrud, and K. Schulten, “A system for interactive molecular dynamics simulation,” 2001, pp. 191–194.

[12] “CUDA C Programming Guide :: CUDA Toolkit Documentation.” [Online]. Available: http://docs.nvidia.com/CUDA/CUDA-c-programming-guide/index.html. [Accessed: 30-Jan-2013].

[13] “CUDA C best practices Guide.” [Online]. Available: http://docs.nvidia.com/CUDA/CUDA-c-best-practices-guide/index.html. [Accessed: 30-Jan-2013].

[14] A. Leach, *Molecular modelling : principles and applications*, 2nd ed. Harlow England ;;New York: Prentice Hall, 2001.

[15] T. A. Halgren, “Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94,” *J. Comput. Chem.*, vol. 17, no. 5–6, pp. 490–519, Apr. 1996.

[16] T. A. Halgren, “Merck molecular force field. II. MMFF94 van der Waals and electrostatic parameters for intermolecular interactions,” *J. Comput. Chem.*, vol. 17, no. 5–6, pp. 520–552, Apr. 1996.

[17] T. A. Halgren, “Merck molecular force field. III. Molecular geometries and vibrational frequencies for MMFF94,” *J. Comput. Chem.*, vol. 17, no. 5–6, pp. 553–586, Apr. 1996.

[18] T. A. Halgren and R. B. Nachbar, “Merck molecular force field. IV. conformational energies and geometries for MMFF94,” *J. Comput. Chem.*, vol. 17, no. 5–6, pp. 587–615, Apr. 1996.

[19] T. A. Halgren, “Merck molecular force field. V. Extension of MMFF94 using experimental data, additional computational data, and empirical rules,” *J. Comput. Chem.*, vol. 17, no. 5–6, pp. 616–641, Apr. 1996.

[20] T. A. Halgren, “The representation of van der Waals (vdW) interactions in molecular mechanics force fields: potential form, combination rules, and vdW parameters,” *J. Am. Chem. Soc.*, vol. 114, no. 20, pp. 7827–7843, Sep. 1992.

[21] T. A. Halgren, “MMFF VII. Characterization of MMFF94, MMFF94s, and other widely available force fields for conformational energies and for intermolecular-interaction energies and geometries,” *J. Comput. Chem.*, vol. 20, no. 7, pp. 730–748, May 1999.

[22] MacKerell A D, *et.al*, “All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins,” *J. Phys. Chem.*, vol. 102, no. 18, pp. 3586–3616, Apr. 1998.

[23] Cornel W D, Cieplak P, Bayly C B, Gould I R, and Merz K M, “A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules,” *J. Am. Chem. Soc.*, vol. 117, no. 19, pp. 5179–5197, May1995.

[24] Schmid N, Eichenberger AP, Choutco A, and Van Gunsteren WF, “Definition and testing of the GROMOS force-field versions 54A7 and 54B7.,” *Eur. Biophys. J.*, vol. 40, no. 7, pp. 843–856.

[25] M. J. Harvey and G. De Fabritiis, “An Implementation of the Smooth Particle Mesh Ewald Method on GPGPU Hardware,” *J. Chem. Theory Comput.*, vol. 5, no. 9, pp. 2371–2377, Sep. 2009.

[26] D. J. Hardy, J. E. Stone, and K. Schulten, “Multilevel summation of electrostatic potentials using graphics processing units,” *Parallel Comput.*, vol. 35, no. 3, pp. 164–177, Mar. 2009.

[27] F. Fleuret, T. Li, C. Dubout, E. K. Wampler, S. Yantis, and D. Geman, “Comparing machines and humans on a visual categorization test,” *Proc. Natl. Acad. Sci.*, vol. 108, no. 43, pp. 17621–

17625, Oct. 2011.

[28] S. D. Laycock and A. M. Day, “A Survey of Haptic Rendering Techniques,” *Comput. Graph. Forum*, vol. 26, no. 1, pp. 50–65, Mar. 2007.

[29] X. Hou and O. Sourina, “Haptic Rendering Algorithm for Biomolecular Docking with Torque Force,” 2010, pp. 25–31.

[30] O. Delalande, N. Férey, G. Grasseau, and M. Baaden, “Complex molecular assemblies at hand via interactive simulations,” *J. Comput. Chem.*, vol. 30, no. 15, pp. 2375–2387, Nov. 2009.

[31] J. N. Block, D. J. Zielinski, V. B. Chen, I. W. Davis, E. C. Vinson, R. Brady, J. S. Richardson, and D. C. Richardson, “KinImmerse: Macromolecular VR for NMR ensembles,” *Source Code Biol. Med.*, vol. 4, no. 1, p. 3, 2009.

[32] A. Bolopion, B. Cagneau, S. Redon, and S. Régnier, “Comparing position and force control for interactive molecular simulators with haptic feedback,” *J. Mol. Graph. Model.*, vol. 29, no. 2, pp. 280–289, Sep. 2010.

[33] Y.-G. Lee and K. W. Lyons, “Smoothing haptic interaction using molecular force calculations,” *Comput.-Aided Des.*, vol. 36, no. 1, pp. 75–90, Jan. 2004.

[34] P. B. Persson, M. D. Cooper, L. A. E. Tibell, S. Ainsworth, A. Ynnerman, and B.-H. Jonsson, “Designing and Evaluating a Haptic System for Biomolecular Education,” 2007, pp. 171–178.

[35] A. M. Wollacott and K. M. Merz Jr, “Haptic applications for molecular structure manipulation,” *J. Mol. Graph. Model.*, vol. 25, no. 6, pp. 801–805, Mar. 2007.

[36] E. Subasi and C. Basdogan, “A New Haptic Interaction and Visualization Approach for Rigid Molecular Docking in Virtual Environments,” *Presence Teleoperators Virtual Environ.*, vol. 17, no. 1, pp. 73–90, Feb. 2008.

[37] S. Plimpton and B. Hendrickson, “A new parallel method for molecular dynamics simulation of macromolecular systems,” *J. Comput. Chem.*, vol. 17, no. 3, pp. 326–337, Feb. 1996.

[38] B. Hess, C. Kutzner, D. van der Spoel, and E. Lindahl, “GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation,” *J. Chem. Theory Comput.*, vol. 4, no. 3, pp. 435–447, Mar. 2008.

[39] W. Liu, B. Schmidt, G. Voss, and W. Müller-Wittig, “Molecular Dynamics Simulations on Commodity GPGPUs with CUDA,” in *High Performance Computing – HiPC 2007*, vol. 4873, S. Aluru, M. Parashar, R. Badrinath, and V. K. Prasanna, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 185–196.

[40] W. Liu, B. Schmidt, G. Voss, and W. Müller-Wittig, “Accelerating molecular dynamics simulations using Graphics Processing Units with CUDA,” *Comput. Phys. Commun.*, vol. 179, no. 9, pp. 634–641, Nov. 2008.

[41] J. A. Anderson, C. D. Lorenz, and A. Travesset, “General purpose molecular dynamics simulations fully implemented on graphics processing units,” *J. Comput. Phys.*, vol. 227, no. 10, pp. 5342–5359, May 2008.

[42] L. Verlet, “Computer ‘Experiments’ on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules,” *Phys. Rev.*, vol. 159, no. 1, pp. 98–103, Jul. 1967.

[43] B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz, “Analysis of the clustering properties of the Hilbert space-filling curve,” *Ieee Trans. Knowl. Data Eng.*, vol. 13, no. 1, pp. 124–141, Feb. 2001.

[44] J. A. van Meel, A. Arnold, D. Frenkel, S. F. Portegies Zwart, and R. G. Belleman, “Harvesting graphics power for MD simulations,” *Mol. Simul.*, vol. 34, no. 3, pp. 259–266, Mar. 2008.

[45] M. J. Harvey, G. Giupponi, and G. D. Fabritiis, “ACEMD: Accelerating Biomolecular Dynamics in the Microsecond Time Scale,” *J. Chem. Theory Comput.*, vol. 5, no. 6, pp. 1632–1639, Jun. 2009.

[46] J. E. Stone, J. C. Phillips, P. L. Freddolino, D. J. Hardy, L. G. Trabuco, and K. Schulten, “Accelerating molecular modeling applications with graphics processors,” *J. Comput. Chem.*, vol. 28, no. 16, pp. 2618–2640, Dec. 2007.

[47] P. Gonnet, “A simple algorithm to accelerate the computation of non-bonded interactions in cell-based molecular dynamics simulations,” *J. Comput. Chem.*, vol. 28, no. 2, pp. 570–573, Jan. 2007.

[48] K. J. Bowers, R. O. Dror, and D. E. Shaw, “Zonal methods for the parallel execution of range-limited N-body simulations,” *J. Comput. Phys.*, vol. 221, no. 1, pp. 303–329, Jan. 2007.

[49] D. Kirk, *Programming massively parallel processors : a hands-on approach*. Burlington MA: Morgan Kaufmann Publishers, 2010.

[50] S. Páll and B. Hess, “A flexible algorithm for calculating pair interactions on SIMD architectures,” *Comput. Phys. Commun.*, Jun. 2013.

[51] P. Eastman and V. S. Pande, “Efficient nonbonded interactions for molecular dynamics on a graphics processing unit,” *J. Comput. Chem.*, vol. 31, no. 6, pp. 1268–1272, Apr. 2010.

[52] J. Xu, Y. Ren, W. Ge, X. Yu, X. Yang, and J. Li, “Molecular dynamics simulation of macromolecules using graphics processing unit,” *Mol. Simul.*, vol. 36, no. 14, pp. 1131–1140, Dec. 2010.

[53] M. S. Friedrichs, P. Eastman, V. Vaidyanathan, M. Houston, S. Legrand, A. L. Beberg, D. L. Ensign, C. M. Bruns, and V. S. Pande, “Accelerating molecular dynamic simulation on graphics processing units,” *J. Comput. Chem.*, vol. 30, no. 6, pp. 864–872, Apr. 2009.

[54] C. D. Christ, A. E. Mark, and W. F. van Gunsteren, “Basic ingredients of free energy calculations: A review,” *J. Comput. Chem.*, p. NA–NA, 2009.

[55] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization,” *Acm Comput. Surv.*, vol. 35, no. 3, pp. 268–308, Sep. 2003.

[56] *Evolutionary computation*. Bristol ; Philadelphia: Institute of Physics Publishing, 2000.

[57] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*. Reading, Mass: Addison-Wesley Pub. Co, 1989.

[58] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, May 1983.

[59] E. Weinberger, “Correlated and uncorrelated fitness landscapes and how to tell the difference,” *Biol. Cybern.*, vol. 63, no. 5, pp. 325–336, Sep. 1990.

[60] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, “Equation of State Calculations by Fast Computing Machines,” *J. Chem. Phys.*, vol. 21, no. 6, p. 1087, 1953.

[61] W. K. Hastings, “Monte Carlo sampling methods using Markov chains and their applications,” *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[62] R. Battiti, *Reactive search and intelligent optimization*, 1st ed. New York: Springer, 2008.

[63] Meta-Heuristics International Conference, *Meta-heuristics: theory & applications*. Boston: Kluwer Academic, 1996.

[64] E. B. Baum, “Towards practical ‘neural’ computation for combinatorial optimization problems,” AIP Conference proceedings 151 on Neural Networks for Computing. 1986, vol. 151, pp. 53–58.

[65] T. Stützle, “Iterated local search for the quadratic assignment problem,” *Eur. J. Oper. Res.*, vol. 174, no. 3, pp. 1519–1539, Nov. 2006.

[66] T. L. Lau and E. P. K. Tsang, “Solving the Processor Configuration Problems with a Mutation-Based Genetic Algorithm,” *Int. J. Artif. Intell. Tools*, vol. 06, no. 04, pp. 567–585, Dec. 1997.

[67] C. Voudouris, E. P. K. Tsang, and A. Alsheddy, “Guided Local Search,” in *Wiley Encyclopedia of Operations Research and Management Science*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2011.

[68] P. Kilby, P. Prosser, and P. Shaw, “Guided Local Search for the Vehicle Routing Problem with Time Windows,” in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, S. Voß, S. Martello, I. H. Osman, and C. Roucairol, Eds. Boston, MA: Springer US, 1999, pp. 473–486.

[69] O. Trott and A. J. Olson, “AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading,” *J. Comput. Chem.*, p. 455–461, 2009.

[70] J. Fuhrmann, A. Rurainski, H.-P. Lenhof, and D. Neumann, “A new Lamarckian genetic algorithm for flexible ligand-receptor docking,” *J. Comput. Chem.*, p. 1911–1918, 2010.

[71] S. M. Long, T. T. Tran, P. Adams, P. Darwen, and M. L. Smythe, “Conformational searching using a population-based incremental learning algorithm,” *J. Comput. Chem.*, vol. 32, no. 8, pp. 1541–1549, Jun. 2011.

[72] C. R. S. Brasil, A. C. B. Delbem, and F. L. B. da Silva, “Multiobjective evolutionary algorithm with many tables for purely ab initio protein structure prediction,” *J. Comput. Chem.*, vol. 34, no. 20, pp. 1719–1734, Jul. 2013.

[73] W. Cai and X. Shao, “A fast annealing evolutionary algorithm for global optimization,” *J. Comput. Chem.*, vol. 23, no. 4, pp. 427–435, Mar. 2002.

[74] R. Murty and D. Okunbor, “Efficient parallel algorithms for molecular dynamics simulations,” *Parallel Comput.*, vol. 25, no. 3, pp. 217–230, Mar. 1999.

[75] Fomin ES, “Consideration of data load time on modern processors for the Verlet table and linked-cell algorithms.,” *J. Comput. Chem.*, vol. 32, no. 7, pp. 1386–99, May 2011.

[76] R. Farber, *CUDA application design and development*. Waltham, MA: Morgan Kaufmann, 2011.

[77] A. Anthopoulos, I. Grimstead, and A. Brancale, “GPU-accelerated molecular mechanics computations,” *J. Comput. Chem.*, p. 2249–2260, Jul. 2013.

[78] “Precision & Performance: Floating Point and IEEE 754 Compliance for NVIDIA GPGPUs. http://docs.nvidia.com/CUDA/CUDA-c-programming-guide/index.html. (Accessed: 30-Jan-2013).” .

[79] J. Yang, Y. Wang, and Y. Chen, “GPGPU accelerated molecular dynamics simulation of thermal conductivities,” *J. Comput. Phys.*, vol. 221, no. 2, pp. 799–804, Feb. 2007.

[80] N. Schmid, M. Bötschi, and W. F. Van Gunsteren, “A GPGPU solvent-solvent interaction calculation accelerator for biomolecular simulations using the GROMOS software,” *J. Comput. Chem.*, p. 1636–1643, 2010.

[81] P. Deift and X. Zhou, “A Steepest Descent Method for Oscillatory Riemann--Hilbert Problems. Asymptotics for the MKdV Equation,” *Ann. Math.*, vol. 137, no. 2, p. 295, Mar. 1993.

[82] *Encyclopaedia of mathematics: an updated and annotated translation of the Soviet “Mathematical encyclopaedia.”*Dordrecht ; Boston : Norwell, MA, U.S.A: Reidel ; Sold and distributed in the U.S.A. and Canada by Kluwer Academic Publishers, 1988.

[83] L. S. Schulman, "*Techniques and applications of path integration*." Mineola, N.Y: Dover Publications, 2005.

[84] H. R. Lourenço, “Job-shop scheduling: Computational study of local search and large-step

optimization methods," *Eur. J. Oper. Res.*, vol. 83, no. 2, pp. 347–364, Jun. 1995.

[85] P. H. V. Penna, A. Subramanian, and L. S. Ochi, "An Iterated Local Search heuristic for the Heterogeneous Fleet Vehicle Routing Problem," *J. Heuristics*, vol. 19, no. 2, pp. 201–232, Sep. 2011.

[86] R. B. Corey and L. Pauling, "Molecular Models of Amino Acids, Peptides, and Proteins," *Rev. Sci. Instrum.*, vol. 24, no. 8, p. 621, 1953.

[87] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *Acm Siggraph Comput. Graph.*, vol. 21, no. 4, pp. 163–169, Aug. 1987.

[88] C. Montani, R. Scateni, and R. Scopigno, "A modified look-up 3 for implicit disambiguation of Marching Cubes," *Vis. Comput.*, vol. 10, no. 6, pp. 353–355, Jun. 1994.

[89] T. S. Newman and H. Yi, "A survey of the marching cubes algorithm," *Comput. Graph.*, vol. 30, no. 5, pp. 854–879, Oct. 2006.

[90] G. Johansson and H. Carr, "Accelerating marching cubes with graphics hardware," 2006, p. 39.

Appendix 1

The work presented in this appendix is courtesy of Gaia Pasqualetto and describes her efforts evaluating the practical application of the algorithms presented in this thesis.

Algorithm practical applications

To further validate and assess our system we have also carried out a set of simulations that aimed to mimic the actual use of the software. To evaluate the induced fit effect, we have selected a series of known biological target for which a series of crystal structures are available (table 1) as apo form and co-crystallised with one or more ligands.

Protein	PDB	Ligand Name
CDK2	1HCL[17]	-
	1GZ8[18]	MBP
	1JVP[19]	LIG
	2R3F[20]	SC8
	2R3H[20]	SCE
	2R3I[20]	SCF
	2R3R[20]	6SC
	4EK4[21]	1CK
	4FKL[21]	CK2
HCV NS5B	1NB4[22]	-
	3UPH[23]	0C1
	3U4O[24]	08E
	3GNW[25]	XNC
	4J02[26]	1JE
	4J06[26]	1JG
	4JJU[27]	1MB
HIV RT	1RTJ[28]	-
	1VRT[29]	NVP Fragment

Table 1. Structures used in the haptic-driven simulations

Our approach was to extract a ligand from a specific complex, then place it in the apo form of the corresponding protein using the haptic-driven simulator. The results obtained from the docking were then compared with the structure of the complex from which the ligand was extracted, measuring both the root mean square deviation (RMSD) of the ligand and the protein residues of the binding pocket. Furthermore, the pocket residues were also

compared with the corresponding amino acids in the apo form and their RMSD calculated. In this manner, we could estimate of how much the protein had reacted to the ligand binding and if the induced-fit effect was comparable to the actual crystallographic structure of the complex.

The protein we have chosen presents three scenarios: CDK2 presents a fairly rigid binding pocket and only few minor side chain movements are visible between the apo form and the ligand/protein complexes; the two chosen allosteric pockets of the HCV NS5b present a more evident induced-fit effect and, in some cases, there is also a clear protein backbone movement; the HIV reverse transcriptase presents one of the most dramatic effects of induced-fit, as the binding pocket of the non nucleoside reverse transcriptase inhibitors (NNRTIs) is not even visible in the apo form. All the simulation were performed on a 2009 quadcore MacPro, running Ubuntu 12.04, equipped with a NVIDIA Geforce 680 and a Phantom Omni haptic device. As we consider the idea of developing a natural and intuitive interface at the core of our software development efforts, we have asked an undergraduate Medicinal Chemistry student to perform the simulations.

2.6.1 CDK2 results.

For CDK2 we have run 8 simulations, one for each ligand reported in table 1. In terms of ligand placement, the haptic-driven simulation performed relatively well; the ligand RMSD values ranged between 0.89Å and 2.23Å. The binding site residues RMSD also produced some very interesting results: Figure 1 shows two example of the results obtained and it is possible to see how the protein has moved in the simulation. Interestingly, the biggest movements are seen on the residues that are indeed the most flexible when the apo and the complex structures are compared.

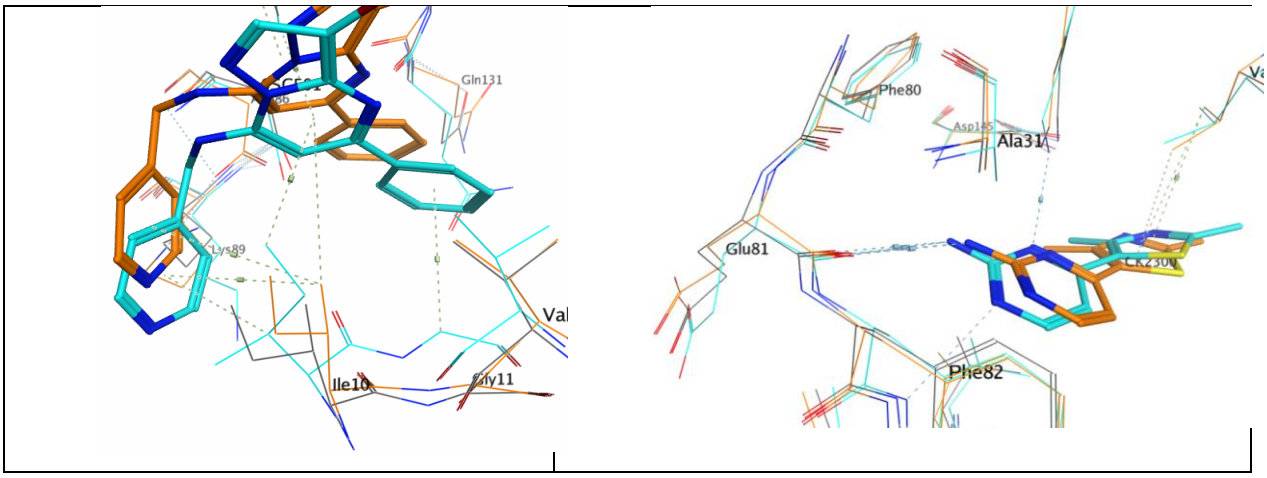


Figure 1: Snapshot of two ligands manually docked in the CDK2 binding site (a: SCF; b: CK2) as examples of residue movements. The apo structure represented in grey; the crystallised complex represented in cyan; the structure obtained from the simulation is indicated in orange.

Figure 2 shows how the different residues have changed and if their movement is toward the conformation present in the complex structure. In this case, it is possible to see a more complex scenario, where some residues move towards the corresponding position in the complex structure (indicated in green) and others that move away from that ideal position (indicated in red).

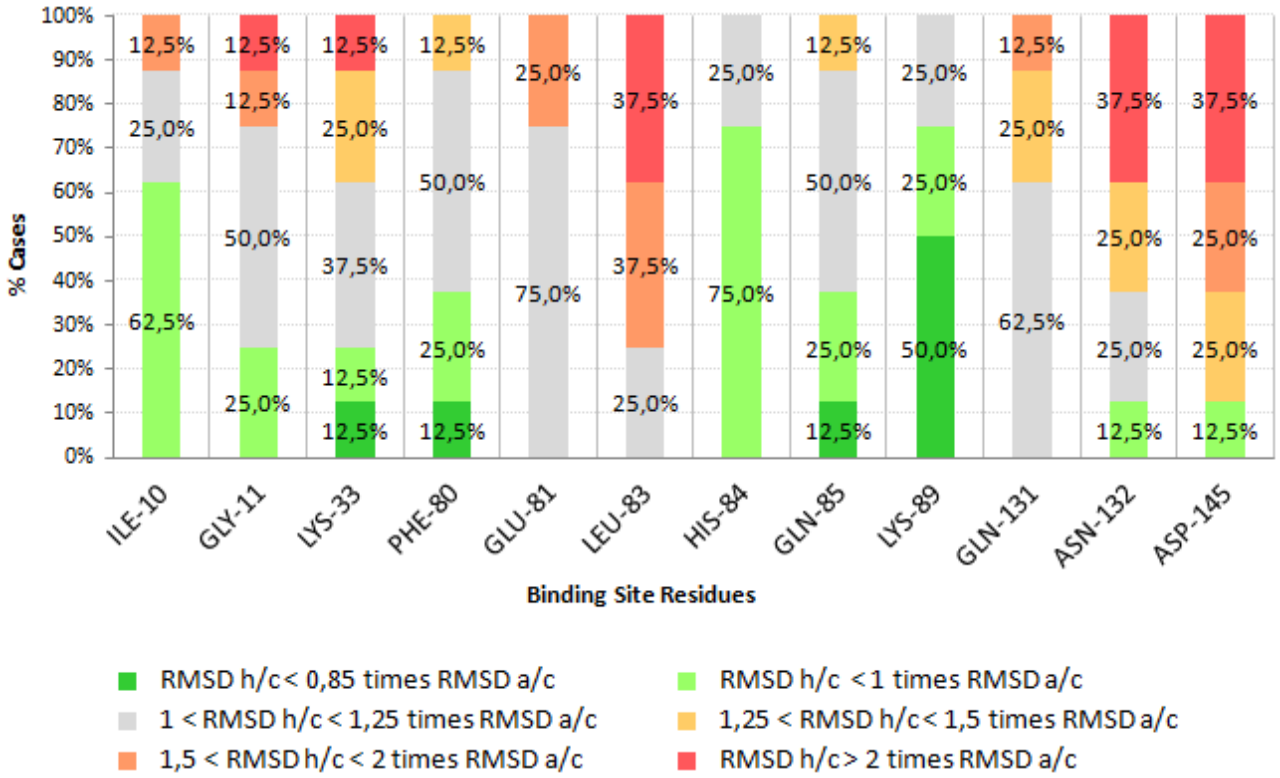


Figure 2. Visual representation of how CDK2 residues had moved during the simulation. The percentage of structures (8 structures – 100%) whose residue RMSD- cryst /haptic. (referred as RMSD c/h in the legend) is lower than correspondent residue RMSD-apo/cryst. (referred as RMSD a/c) are shown in green while higher ones are shown in red.

However, we should point out that the intervals parameters represented in Figure 2 are not absolute values but they represent how far away the haptic generated structure is to the crystallised structure, relative to the the RMSD of the crystallised structure vs the apo structure. As most of the residues virtually retain the same conformation in all the different structures, their actual absolute RMSD-apo/cryst. values are very small. Hence, any minor, favourable or unfavourable, movement generated from the haptic-driven simulation will be flag out as significant. An exception is represented by Lys33, which moves considerably from the apo structure to the 1JVP crystal structure (RMSD of 2.45Å). In the haptic driven

simulation, the residue move toward the position present in the crystal giving a RMSD-haptic/cryst. of 1.75Å (figure 3). In this case it is clearly evident the induced-fit effect generated by the hapic-driven ligand placement.

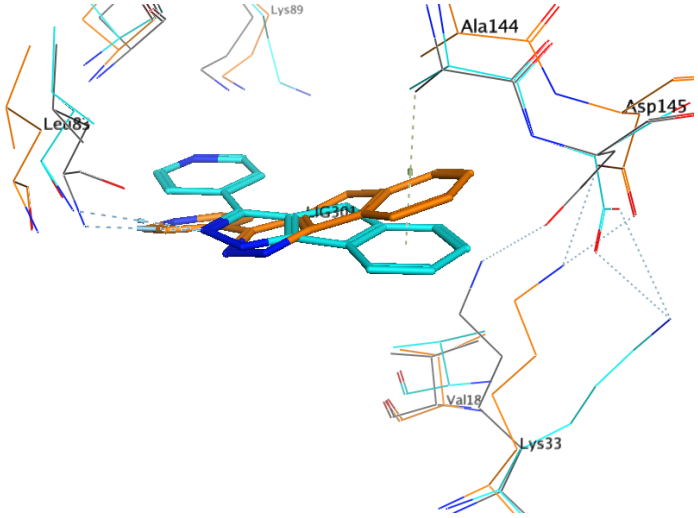


Figure 3: Snapshot of ligand "LIG" in the CDK2 binding site. Lys33 moves toward the position present in the crystallized structure. The apo structure represented in grey; the crystallised complex represented in cyan; the structure obtained from the simulation is indicated in orange.

2.6.2 HCV NS5b results.

The two pockets selected for this target (Palm and Thumb-2) present a more obvious induced-fit effect, compared to what has been observed with CDK2. Indeed, the results obtained are more informative (figure 4) as significant conformational changes from the apo structure to both the haptic generated structure and the crystal complex are evident for a good number of residues.

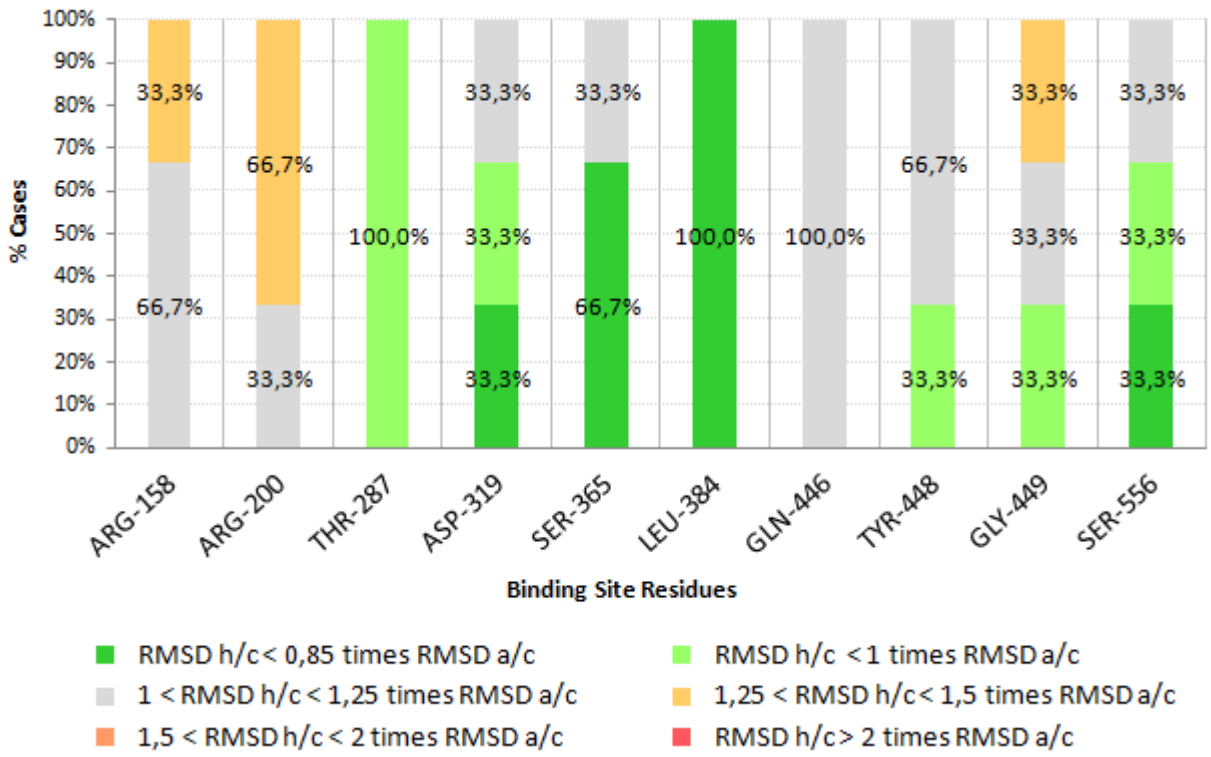


Figure 4: Visual representation of how HCV NS5B residues had moved during the simulation. The percentage of structures whose residue RMSD- cryst /haptic. (referred as RMSD c/h in the legend) is lower than correspondent residue RMSD-apo/cryst. (referred as RMSD a/c) are shown in green while higher ones are shown in red.

The data presented in figure 4 suggests that there is a general tendency of the binding site residues to move in the direction of their respective complex conformation. In one specific case, even a relevant backbone movement is present (figure 5). Interestingly, some residues move away from the ideal final position and this seems to be happening for two reasons: the placement of specific ligands is not accurate (the ligands RMSD range between 1.10Å and 3.10Å); some specific residues should undergo a considerable conformational change and the algorithm is not yet able to explore the full rotamer space available to the

residue.

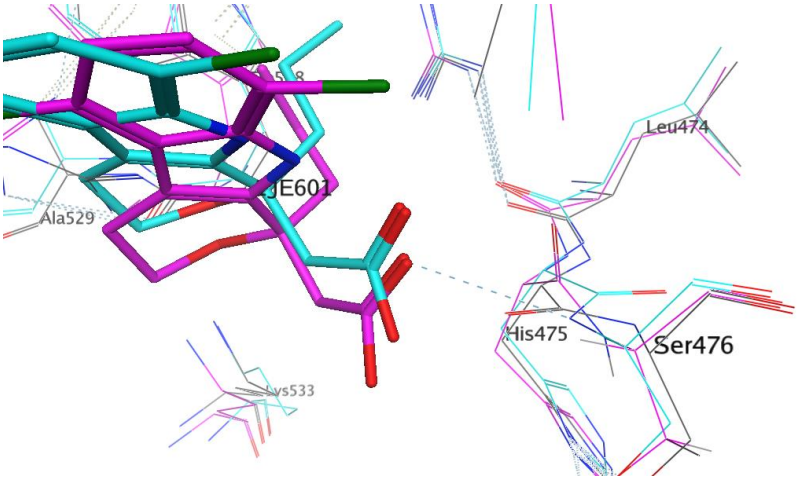


Figure 5. Snapshot of 1JE in the HCV NS5b binding pocket. His475 backbone movement is highlighted. The apo structure represented in grey; the crystallised complex represented in cyan; the structure obtained from the simulation is indicated in purple.

2.6.3 HIV RT results.

The HIV reverse transcriptase presents a very different scenario: the NNRTIs pocket is not structured or visible in the apo form. The extent of the induced fit effect, in this case, is really remarkable. The haptic-driven placement results are shown in figure 6 and they clearly demonstrate the complexity of these simulations. The user can push the ligand inside the pocket and all the RMSD values shows a positive move.

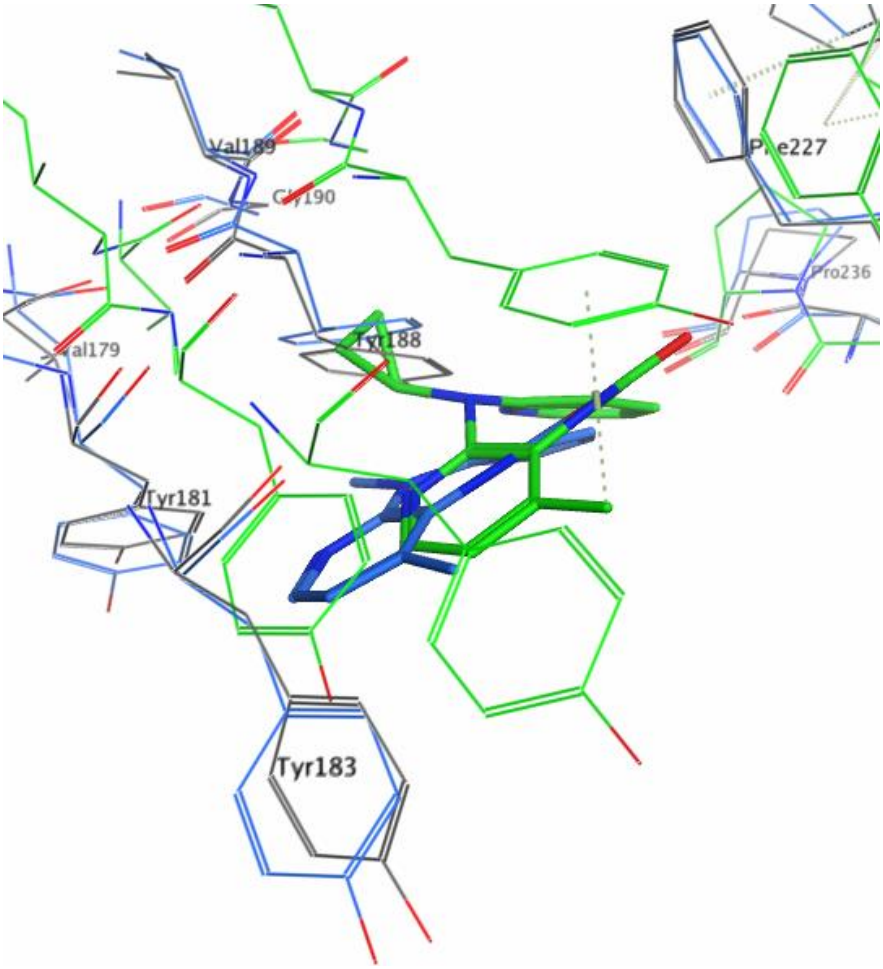


Figure 6. Snapshot of ligand NVP in the HIV RT. The apo structure represented in grey; the crystallised complex represented in green; the structure obtained from the simulation is indicated in blue.

To a certain extent, this is not surprising, considering how different the apo structure and the crystal complex are. Indeed, we can see that although the haptic-driven simulation is able to induce a substantial rearrangement in the protein to accommodate the ligand in a reasonable manner, the extent of the protein structural change upon ligand binding in reality is so extensive that, once again, the algorithm cannot explore efficiently such conformational space.

3. Conclusions

In this paper we have reported a hybrid evolutionary strategy for exploring molecular energy landscapes in a small elapsed time period. Our algorithm was tuned to take advantage of the extra parallelism that Cuda streams can provide as the forcefield used was also coded in Cuda. The proposed method works well for conformational sampling in situations where the exit criterion for the meta-heuristic is the expiration of a very small time-lapse. We have also tested the algorithm in a possible actual usage scenario, by implementing it in our haptic-driven molecular modelling simulator. Using a series of crystal structures, we have examined how the fully flexible haptic-driven simulation was able to mimic an induced-fit effect. Overall, we believe the results are positive and very encouraging. The haptic-driven simulations occur in a very natural and intuitive fashion. Furthermore, the results obtained shows that we can induce meaningful and appropriate conformational changes into a protein by placing a ligand using an interactive, real time approach. The data obtained also shows some limitations of this method. Although the algorithm performs extremely well, it is possibly too conservative when sampling the conformational space of the protein residues. This should not be considered always a negative aspect, as, for example, this more cautious exploration would preserve some important structural information in a binding pocket in those cases when the user applies a strong force to the ligand through the haptic device. However, as in the case of HIV RT, sometime the changes are indeed dramatic and they would require an algorithm able to sample a considerably bigger conformational space to obtain an accurate prediction. In conclusion, the results presented are very promising and we are now taking the next step in the validation of this approach, by comparing the haptic-driven simulator to other commercial flexible docking packages.