Ricardo Alexandre do Rosário Ribeiro

Master of Science

Protein docking GPU acceleration

Dissertation plan submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Informatics Engineering

Adviser: Ludwig Krippahl, Full Professor, NOVA University of Lisbon

Co-adviser: Hervé Paulino, Associate Professor, NOVA University of Lisbon

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

July, 2018

Resumo

As proteinas são um dos nutrientes indispensáveis à vida na Terra e a um longo prazo no Universo em geral, a capacidade dos seus compostos (que segundo o ramo da biologia/bioquimica se ditam por amino-ácidos) de se juntarem e formarem vastos complexos permite o crescimento fisico dos seres-vivos. Contudo, este processo de ajuntamento em termos informáticos apenas pode ser determinado por matrizes tri-dimensionais em situações em que temos vários compostos interligados, já que os mesmo assumem estruturas tri-dimensionais, sendo mais dificil determinar experimentalmente uma previsão correta sobre estes ultimos do que numa situação em que temos proteinas singulares pois existem muitas combinações possiveis para a docagem entre os pares, logo é necessário recorrer a ferramentas que oferecem a computação adequada para determinar estas previsões, a fim de poder-se avançar no estudo das interações entre proteinas e na concepção de produtos à base das mesmas.

O presente documento aborda uma proposição para a paralelização do algoritmo BiGGER implementado pelo prof. Ludwig Krippahl e outros professores associados à criação do algoritmo, recorrendo a técnicas de computação acelerada i.e. utilizar o GPU da máquina em que corre o algoritmo para auxiliar o CPU na computação que é necessária para o algoritmo na versão actual, de forma a que com mais recursos à disposição, o tempo de execução do BiGGER baixe drasticamente por consequencia do aumento significativo de perfomance.

Em caso de sucesso, a complexidade futura do algoritmo permitirá a adição de mais vantagens face aos seus concorrentes, e por consequência, uma proposta de valor para quem pretenda utilizar o open-chemera como ferramenta de trabalho eficiente no estudo das interações entre os complexos de proteinas em qualquer máquina que tenha uma placa grafica com as caracteristicas adequadas.

Palavras-chave: proteinas, docagem de proteinas,computação acelerada,GPU, Bio-Informática, BiGGER

Abstract

The dissertation must contain two versions of the abstract, one in the same language as the main text, another in a different language. The package assumes that the two languages under consideration are always Portuguese and English.

The package will sort the abstracts in the appropriate order. This means that the first abstract will be in the same language as the main text, followed by the abstract in the other language, and then followed by the main text. For example, if the dissertation is written in Portuguese, first will come the summary in Portuguese and then in English, followed by the main text in Portuguese. If the dissertation is written in English, first will come the summary in English and then in Portuguese, followed by the main text in English.

The abstract shoul not exceed one page and should answer the following questions:

- What's the problem?
- Why is it interesting?
- What's the solution?
- What follows from the solution?

Keywords: Keywords (in English) ...

Índice

Lista de Figuras ix

Lista de Tabelas xi

Listagens xiii

Glossário xv

Siglas xvii

1 Introduction 1
- 1.1 A Bit of History . . . 1
- 1.2 Disclaimer . . . 1

2 Estado-da-arte 3
- 2.1 Classificações de algoritmos . . . 3
- 2.2 Algoritmos de correlação espacial . . . 4
- 2.3 Fast Fourier Transform . . . 4
- 2.4 O algoritmo BiGGER para docagem de proteinas . . . 4
- 2.5 BoGIE . . . 5
- 2.6 O uso de GPU como ferramenta auxiliar . . . 6
  - 2.6.1 APIs para computação acelerada . . . 7
  - 2.6.2 Optimizações dos algoritmos existentes utlizando a GPU . . . 7

3 Plano de trabalhos para a Elaboração da dissertação 9
- 3.1 Possibilidades de optimização . . . 9
- 3.2 Desafios . . . 9
- 3.3 Plano de Trabalhos . . . 10

Bibliografia 11

Lista de Figuras

2.1 Representação em 2D das matrizes resultantes do segundo passo do BoGIE, as células preenchidas a tracejado diagonal correspondem à matriz de superficie, as com pontos correspondem à matriz *core* e a núvem com tracejado continuo representa o corte associado à esfera de van der waals com a proteina localizada ao centro [4]. 6

Lista de Tabelas

Listagens

Glossário

aliquam	tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris..
computer	An electronic device which is capable of receiving information (data) in a particular form and of performing a sequence of operations in accordance with a predetermined but variable set of procedural instructions (program) to produce a result in the form of information or signals..
cras viverra	metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat..
donec nonummy	pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo..
integer sapien	est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus..
lorem ipsum	dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris..
maecenas lacinia	nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem..
morbi ac	orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus..
morbi dolor	nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum..

nam lacus	libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi..
nam dui	ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo..
name arcu	libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo..
nulla malesuada	porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis..
sed lacinia	nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus..

Siglas

aaa	acornym aaa.
aab	acornym aab.
aba	acornym aba.
abbrev	abbreviation of a longer text.
AEU	adipiscing elit ut.
AFM	aenean faucibus morbi.
AMD	a magna donec.
ANP	ac nunc praesent.
ATG	amet tortor gravida.
AVF	adipiscing vitae felis.
bbb	acornym bbb.
CAS	curabitur auctor semper.
CDG	curabitur dictum gravida.
CEA	congue eu accumsan.
CIV	consectetuer id vulputate.
DIA	duis eget orci.
DNM	dolor nulla malesuada.
DNMC	duis nibh mi congue.
DRN	dignissim rutrum nam.
EII	est iaculis in.
ENE	et netus et.
EPA	eu pulvinar at.
ESQ	eleifend sagittis quis.
ESV	eget sem vel.
ETS	eu tellus sit.

FUP	fringilla ultrices phasellus.
LID	lorem ipsum dolor.
LNE	libero nonummy eget.
LUB	leo ultrices bibendum.
LVU	lectus vestibulum urna.
MAC	mollis ac nulla.
MFA	malesuada fames ac.
MNA	mauris nam arcu.
MTS	morbi tristique senectus.
NDV	nulla donec varius.
NPH	neque pellentesque habitant.
OER	orci eget risus.
PEV	purus elit vestibulum.
PIS	placerat integer sapien.
PQV	pretium quis viverra.
SAO	sit amet orci.
SNE	sem nulla et.
STC	sit amet consectetuer.
TEM	turpis egestas mauris.
ULC	ut leo cras.
UPA	ut placerat ac.
VAE	vehicula augue eu.
VMR	viverra metus rhoncus.
xpto	and extension of a xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto xpto.

Capítulo 1

Introduction

1.1 A Bit of History

The *novathesis* was originally developed to help MSc and PhD students of the Computer Science and Engineering Department of the Faculty of Sciences and Technology of NOVA University of Lisbon (DI-FCT-NOVA) to write their thesis and dissertations Using LaTeX. These student can easily cope with LaTeX by themselves, and the only need some help in the bootstrap process to make their life easier.

However, as the template spread out among the students from other degrees at FCT-NOVA, the demand for am easier-to-use template as grown. And the template in its current shape aims at answering the expectations of those that, although they are not familiar with programming nor with markup languages, so still feel brave enough to give LaTeX a try and rejoice with the beauty of the texts typeset by this system.

1.2 Disclaimer

It is up to you, the student, to read the FCT and/or NOVA regulations on how to format and submit your MSc or PhD dissertation.

This template is endorsed by the FCT-NOVA and even linked from its web pages, but it is not an official template. This template exists to make your life easier, but in the end of the line you are accountable for both the looks and the contents of the document you submit as your dissertation.

Capítulo 2

Estado-da-arte

2.1 Classificações de algoritmos

Com o passar do tempo cada vez mais algoritmos e respetivas adaptações para simular a docagem dos complexos de proteinas têm surgido, surgindo também formas de os classificar. Um algoritmo pode ser classificado em função da forma que trata a rigidez dos pares de proteinas a juntar ou até mesmo pelo modelo matemático, como por exemplo se aplicam a Fast Fourier Transform ou não. Segundo o relatório de Huang Wenfan sobre o uso de Fast Fourier Transform associado à docagem de proteinas rigidas [6], existem três formas de se classificar um algoritmo de docagem, isto em relação às caracteristicas das proteinas em si, e não sobre o modelo matemático ou metodologias necessárias para o implementar:

- **Docagem Flexivel** Em que ambos os complexos receptor e ligando são considerados como sendo corpos flexiveis e adaptáveis sendo, no entanto, a mesma flexibidade interpretada pelo algoritmo de forma simplificada ou limitada, e por consequência pode-se aplicar um modelo através de simulações de docagem.

- **Docagem Semi-Flexivel** Um dos corpos é considerado rigido e o outro não, em situações normais este tipo de algoritmos trata o ligando como se fosse a proteina flexivel, já que esta é mais pequena que a sua parceira, tendo assim uma maior probabilidade de mudar a sua forma, outra justificação tem a ver com os custos de computação serem mais baixos do que se considerarmos os receptores como flexiveis.

- **Docagem Rigida** O par é considerado como sendo flexivel na sua integridade, sendo também considerado que na docagem entre os dois corpos uma das proteinas irá

acabar por penetrar a outra o que leva a que se tenha de adaptar o conjunto de soluções para o problema em seis dimensões, 3 para a rotação e 3 para a translação.

Estes três conceitos são necessários para que se possa entender melhor os algoritmos nas secções seguintes, já que as suas complexidades dependem em certa forma da maneira como tratam a flexibilidade dos pares de proteinas e por consequência do número de dimensões que o espaço solução irá ter assim como a qualidade do conjunto solução encontrado.

2.2 Algoritmos de correlação espacial

2.3 Fast Fourier Transform

Representa o método a partir do qual ocorreram adaptações que levaram origem à criação do BiGGER [4], a maior parte do software de docagem de proteinas funciona com ele. A origem deste método remonta ao artigo de Katchalski Katzir et al [2], considera ambos os pares como corpos rigidos, o que no âmbito do ponto 2.1 pode-se assumir como docagem rigida.

2.4 O algoritmo BiGGER para docagem de proteinas

Este algoritmo (acrónimo para *Bimolecular complex Generation with Global Evaluation and Ranking*) [2] de acordo com a classificação referida no ponto 2.1 enquadra-se no paradigma da docagem flexivel[4] permite resolver o problema de conseguir prever o ajuntamento dos complexos proteicos, consistindo essencialmente em dois passos, o primeiro efectua uma redução de possiveis configurações resultantes de passos de translação e rotação numa magnitude de cerca de 10^{15} configurações para poucos milhares das anteriores, isto através do algoritmo BoGIE relatado no ponto 2.5. A segunda fase do algoritmo consiste em aplicar metodologias de aprendizagem automática de modo a que se possa prever qual das configurações resultantes corresponde ao melhor ajuste entre os dois complexos, isto é, a que tem o score mais elevado. Em termos de complexidade temporal, este algoritmo assume valores mais optimais ($O(N^{(}2.8))$[4]) do que os algoritmos que recorrem ao Fast Fourier Transform, estes últimos assumem como complexidade $O(N^3 * log_2(N^3))$[2] e estes por sua vez têm uma melhor complexidade do que a dos algoritmos de correlação espacial ($O(N^6)$). O motivo pelo qual o das 3 especies de algoritmos o BiGGER assume-se com maior eficiencia em termos de computação deve-se ao facto de este ter sido implementado com diversas optimizações face aos algoritmos FFT, uma delas consiste no uso de uma heuristica mais eficiente no passo da eliminação de possibilidades: descarta situações em que existem sobreposições entre cores ou até mesmo situações que não cumprem com os limites impostos nas restrições [4]. O tempo de computação, segundo os autores do algoritmo estava situado entre as 2H e as 8H, dependendo do par de proteinas, em

contraste com o tempo de execução dos algoritmos FTT que ronda as 6H, com um CPU do ano de 2000 (Intel Pentium II 450 MHz dispoe apenas 1 core)[4]. Segundo a lei de moore, o número de transistors presentes num CPU duplica a cada 2 anos, e por consequencia a capacidade computacional. Num computador actual o tempo de execução do BiGGER provavelmente será menor, demorando entre 1H e 4H por exemplo.

2.5 BoGIE

Acrónimo para *Boolean Geometric Interaction Evaluation*[2] [4], é um algoritmo de pesquisa em grelha utilizado na primeira fase do BiGGER, que é referido no ponto seguinte, mais precisamente na amostragem da população total de configurações possiveis para milhares, existem dois processos principais a considerar, sendo o primeiro a definição de uma matriz tridimensional de booleanos em que cada posição da matriz representa uma parcela da forma que o complexo assume, assume o valor1 se a celula corresponde a uma parcela da proteina cujo centro se encontra a uma distancia tri-dimensional que se designa por esfera de van der waals de qualquer outro átomo pertencente a outra proteina, as células com o valor 0 correspondem a frações do complexo que são consideradas como externas. O segundo passo gera duas matrizes de boolean semelhantes às anteriores para cada um dos elementos do par: a matriz de superficie (*surface matrix*) e a matriz central (*core matrix*) tal como está ilustrado graficamente na figura 2.1. Os elementos celulares que ocupam a matriz de superficie são aqueles que na matriz inicial do passo anterior assumiram valor 1 mas tinham vizinhos com valor 0, ou seja, pretende-se os pontos de fronteira. Na segunda matriz constam as células em que quer o seu valor, quer o das suas células vizinhas assumem valor verdadeiro o que corresponde a posições em que o seu centro está próximo do centro do complexo proteico ou podendo até mesmo coincidir. A forma de garantir que se consegue obter a superficie molecular da proteina é através da operação logica XOR (OU exclusivo), que terá como output 1 apenas nos pontos da fronteira, pois é aqui que o resultado do XOR associado aos dois pontos selecionados dá o valor verdadeiro, já que os valores entre as duas células são diferentes e falso se forem iguais. Sendo assim a complexidade deste algoritmo está associada mais com o primeiro passo do que com o segundo, já que este ultimo depende do output da matriz resultante do primeiro passo e apenas executa um conjunto de operações XOR o que não é assim tão custoso em termos de memória e tempo comparando com a medição para cada célula de uma distancia.

De notar, no entanto, que ambos os passos podem ser optimizados recorrendo ao GPU, no capitulo 3 serão detalhadas possiveis abordagens à paralelização desta etapa do BiGGER, podendo trazer melhorias para além do uso do XOR.

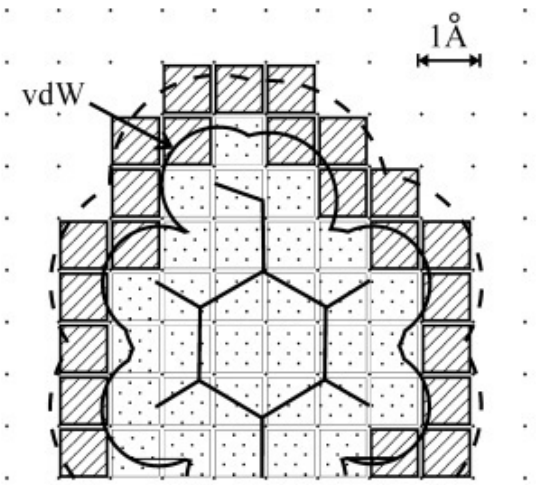

Figura 2.1: Representação em 2D das matrizes resultantes do segundo passo do BoGIE, as células preenchidas a tracejado diagonal correspondem à matriz de superficie, as com pontos correspondem à matriz *core* e a núvem com tracejado continuo representa o corte associado à esfera de van der waals com a proteina localizada ao centro [4].

2.6 O uso de GPU como ferramenta auxiliar

GPU(*Graphical Processor Unit*) consiste na unidade de processamento gráfico que coexiste na placa gráfica presente em qualquer computador, sendo especializada em processamento gráfico que por norma é intensivo demais para o CPU, esta componente consegue ser mais poderosa a executar instruções com custos de complexidade temporal elevadas do que a unidade de processamento central (CPU), já que o CPU têm um número de cores limitado, que pode variar entre entre 2 cores para computadores fabricados até 2010 e 16 cores para computadores mais recentes. Quem quiser implementar uma paralelização para a versão sequêncial de um dado programa, recorrendo ao CPU necessita de conhecimentos de programação com threads assim como gerir o acesso exclusivo a variáveis partilhadas entre as mesmas, tal pode ser feito por gestão de locks ou por instruções atómicas, resultando num programa paralelizado mas com um speedup limitado comparando com o mesmo associado a uma paralelização através de aceleradores na GPU, esta última acaba por ser uma ferramenta mais potente para paralelizar programas já que a sua arquitetura é consistente em uma quantidade variável de multi-processadores (*streaming multi-processors*) [5] que por si só têm um conjunto de processadores escalares (SPs) que são também conhecidos como os cores da GPU, podendo uma GPU ter até 512 cores no total, isto se a arquitetura em questão for a da espécie Fermi. Na arquitetura de uma GPU está também presente uma zona de memória global que pode variar entre MBs e GBs, que pode ser partilhada entre os diversos SPs. Porém continuamos a necessitar do CPU para comunicar com o GPU, em que este último funciona como *device* e o CPU funciona como *host* numa interface de comunicação de dados entre os 2, uma analogia para facilitar a compreensão ao leitor: Imagine que está a fazer mudanças em sua casa, e

quer mover um móvel muito pesado de uma divisão para a outra. Você tem um ajudante consigo que é mais forte fisicamente - Ele representa o GPU, então os dois unem-se para desempenhar a tarefa de mover o móvel de um sitio para o outro, sendo o seu colega quem tem a tarefa mais custosa de arrastar o móvel e o leitor de dar as instruções para ele o conseguir fazer. Sendo assim é essencial usar o GPU para utensilio na paralelização de algoritmos que façam computações muito pesadas, como é o caso das etapas presentes nos algoritmos destacados nas secções anteriores.

2.6.1 APIs para computação acelerada

Em termos de programação em GPUs existe como API o CUDA (*Compute Unified Device Architecture*) que foi implementado pela NVIDIA para que o programador consiga aplicar de certa forma a analogia presente no ponto anterior. O CPU executa a parte sequêncial do programa enquanto que o GPU, com a sua quantidade numerosa de cores, executa a parte que requer computação mais intensiva como por exemplo multiplicação de matrizes (matmult), através de invocações especiais que se denominam *kernels*, após as alocações respetivas de memória do host para o device, que são feitas por parte do CPU. Se alguém quiser utilizar CUDA para paralelizar o seu programa, é necessário transferir a toolkit respetiva ao sistema operativo disponivel na página da NVIDIA. No entanto também existem mais APIs/frameworks para o efeito, como por OpenCL(*Open computing language*) que é adequado para programação paralela em Free pascal enquanto que o CUDA é mais utilizado para C/C++ ou até mesmo Python (pyCUDA), ambos são semelhantes no sentido de implementar o protocolo de comunicações entre o CPU e o GPU, e de permitir invocar kernels para este último computar a parte intensiva.

2.6.2 Optimizações dos algoritmos existentes utlizando a GPU

Já existem vários softwares e artigos dedicados à paralelização da abordagem usando FFT [5] o que muitos têm em comum é o facto de utilizarem CUDA, neste caso utilizam CuFFT, que é uma versão própria que fornece as implementações necessários para que a perfomance do algoritmo seja aumentada em 10x face às versões que utilizam o CPU para o processamento respetivo[1] . O MEGADOCK 4.0 é um software de docagem de proteinas de origem japonesa [3], este programa é adequado para máquinas que têm muitos cores de GPU e CPU á disposição, caracteristicas tipicas de supercomputadores, não sendo, por observação própria, adequado para utilização em computadores pessoais de forma a obter os resultados optimais. Foi implementado para GPU não só por CUDA como também por MPI e OpenMP. Detalhando em termos técnicos a implementação,

Capítulo 3

Plano de trabalhos para a Elaboração da dissertação

3.1 Possibilidades de optimização

O programa que serve de interface gráfica ao BiGGER, que se pode fazer a clonagem do repositório em *https://github.com/lkrippahl/Open-Chemera* encontra-se implementado em Free Pascal, 97.6% do código total, ao acordo com o que foi abordado previamente, o trabalho a realizar na elaboração incide sobre o código respetivo às unidades bogie.pas e dockdomains.pas, que implementam as etapas definidas nos pontos 2.4 e 2.5. Face à possibilidade de não existir nenhuma versão do CUDA para programar paralelizações em Free Pascalpõe-se de lado esta última alternativa em troca de uma outra que recorra à framework OpenCL, que é suportada pelo Lazarus (IDE a utilizar durante a Elaboração). Poderão vir a ser implementados para a paralelização das duas unidades, kernels que executam operações de mapeamento, em que o espaço de possibilidades a tratar na primeira fase do BiGGER poderá ser associado a uma estrutura de dados, dividida por blocos em que estes últimos serão constituidos por casas indexadas pelo ID da thread correspondente, pelo que a indexação geral estará associada a uma formula que envolva a posição da thread no bloco e o número de bloco, para cada uma das posições o core do GPU executará a função que determina se a configuração é aceite ou não.

3.2 Desafios

Os desafios incidem-se sobre dois pontos:

- **Criação das grelhas**
- **Pesquisa de sobreposições**

3.3 Plano de Trabalhos

Bibliografia

[1] *cuFFT*. 2018. URL: https://developer.nvidia.com/cufft.

[2] L. Krippahl. *Integrating protein structural information*. 2003.

[3] M. Ohue, T. Shimoda, S. Suzuki, Y. Matsuzaki, T. Ishida e Y. Akiyama. "MEGADOCK 4.0: an ultra–high-performance protein–protein docking software for heterogeneous supercomputers". Em: *Bioinformatics* 30.22 (2014), pp. 3281–3283. DOI: 10.1093/bioinformatics/btu532. eprint: /oup/backfile/content$_public/journal/bioinformatics/30/22/10.1093_bioinformatics_btu532/2/btu532.pdf. URL: http://dx.doi.org/10.1093/bioinformatics/btu532.

[4] P. N. Palma, L. Krippahl, J. E. Wampler e J. Moura. "BIGGER: A new (soft) docking algorithm for predicting protein interactions". Em: 39 (jun. de 2000), pp. 372–84.

[5] Ritchie, D. W., Venkatraman e Vishwesh. *Ultra-fast FFT protein docking on graphics processors | Bioinformatics | Oxford Academic*. 2010. URL: https://academic.oup.com/bioinformatics/article/26/19/2398/229220.

[6] H. Wenfan. *Rigid Body Protein Docking by Fast Fourier Transform*. 2005. URL: https://www.comp.nus.edu.sg/~leowwk/hyp/huangwenfan.pdf.