UNIVERSIDADE FEDERAL DE PERNAMBUCO

Centro de Biociências

Programa de Pós-Graduação em Inovação Terapêutica

LUIZ FELIPE GOMES REBELLO FERREIRA

Novas estratégias para métodos *in silico* na inovação terapêutica utilizando computação distribuída: GriDoMol

Recife

2017

LUIZ FELIPE GOMES REBELLO FERREIRA

Novas estratégias para métodos *in silico* na inovação terapêutica utilizando computação distribuída: GriDoMol

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Inovação Terapêutica da Universidade Federal de Pernambuco, para a obtenção do Título de Doutor em Inovação Terapêutica

Orientador: Prof. Dr. Marcelo Zaldini Hernandes

Recife

2017

Catalogação na fonte
Elaine Barroso
CRB 1728

Ferreira, Luiz Felipe Gomes Rebello
Novas estratégias para métodos *in silico* na inovação terapêutica utilizando computação distribuída: GriDoMol / Luiz Felipe Gomes Rebello Ferreira- Recife: O Autor, 2017.

157 folhas: il., fig., tab.
Orientador: Marcelo Zaldini Hernandes
Tese (doutorado) – Universidade Federal de Pernambuco.
Centro de Biociências. Inovação Terapêutica , 2017.
Inclui referências e apêndices

1. Bioinformática 2. Grid computacional 3. Moléculas I. Hernandes, Marcelo Zaldini (orientador) II. Título

660.6 CDD (22.ed.) UFPE/CCB-2017-160

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE BIOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM INOVAÇÃO TERAPÊUTICA

Recife, 17 de fevereiro de 2017

Tese de Doutorado defendida e **APROVADA** em 17 de fevereiro de 2017, cuja Banca Examinadora foi constituída pelos seguintes professores:

PRESIDENTE E EXAMINADOR INTERNO: Prof. Dr. Marcelo Zaldini Hernandes

(Departamento de Ciências Farmacêuticas - Universidade Federal de Pernambuco)

Assinatura:_____

PRIMEIRO EXAMINADOR EXTERNO: Prof. Dr. João Bosco Paraíso da Silva

(Departamento de Química Fundamental - Universidade Federal de Pernambuco)

Assinatura:_____

SEGUNDO EXAMINADOR EXTERNO: Prof. Dr. Carlos Henrique Madeiros Castelletti

(Instituto Agronômico de Pernambuco)

Assinatura:_____

TERCEIRO EXAMINADOR EXTERNO: Prof. Dr. Ramiro Brito Willmersdorf

(Departamento de Engenharia Mecânica - Universidade Federal de Pernambuco)

Assinatura:_____

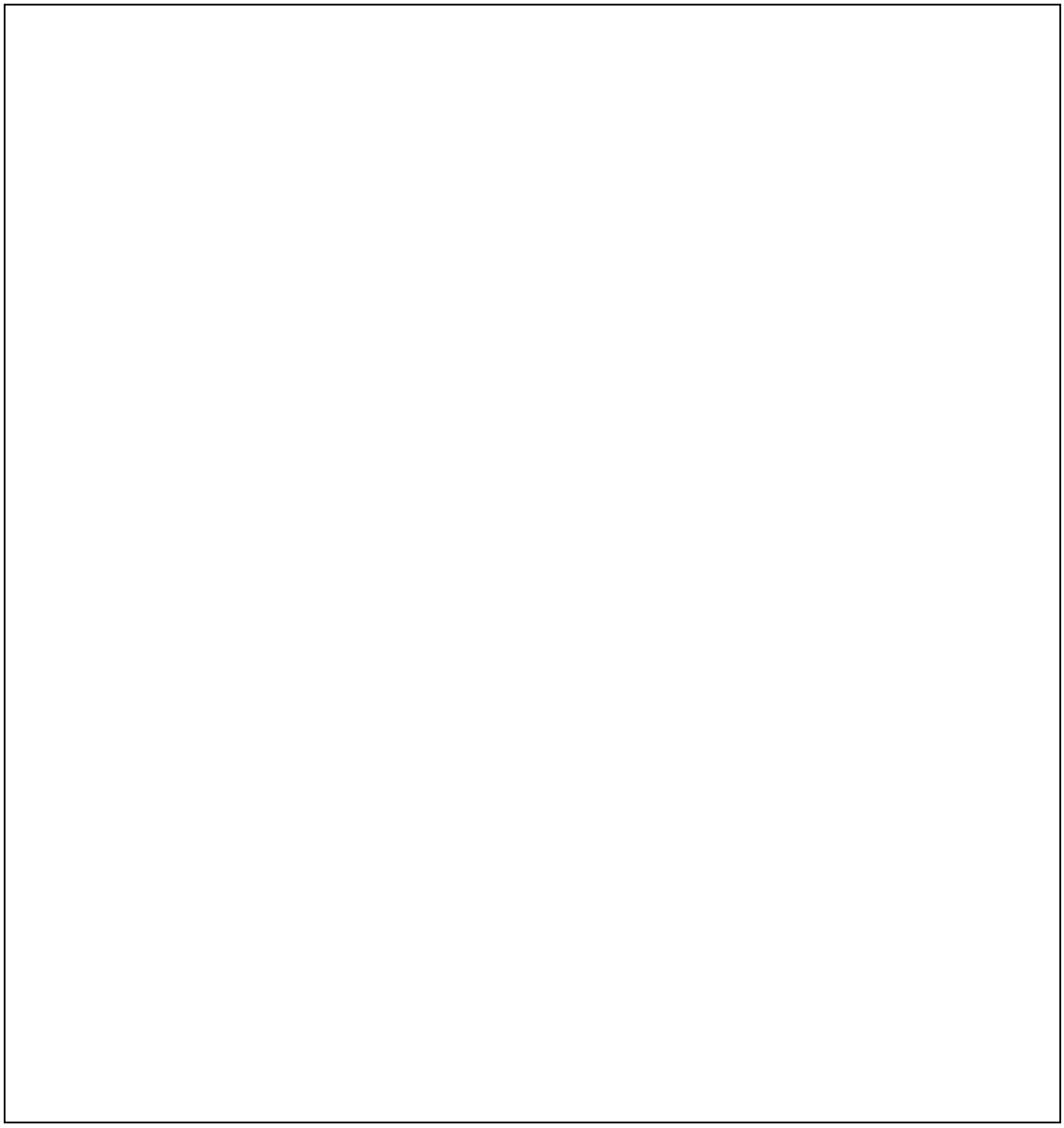
QUARTO EXAMINADOR EXTERNO: Prof. Dr. Antônio Mauro Rezende

(Centro de Pesquisa Aggeu Magalhães)

Assinatura:_____

FERREIRA, L.F.G.R.	Novas estratégias para métodos *in silico* na inovação terapêutica utilizando computação distribuída: GriDoMol	2,5 cm espaço reservado para etiqueta de localização	Doutorado PPGITUFPE 2017

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.



UNIVERSIDADE FEDERAL DE PERNAMBUCO

Programa de Pós-Graduação em Inovação Terapêutica

REITOR

Prof. Dr. Anísio Brasileiro de Freitas Dourado

VICE-REITOR

Profª. Drª. Florisbela de Arruda Camara e Siqueira Campos

PRÓ-REITOR PARA ASSUNTOS DE PESQUISA E PÓS-GRADUAÇÃO

Prof. Dr. Ernani Rodrigues de Carvalho Neto

DIRETORA DO CENTRO DE BIOCIÊNCIAS

Profª. Drª. Maria Eduarda Lacerda de Larrazábal da Silva

VICE-DIRETORA DO CENTRO DE BIOCIÊNCIAS

Profª. Drª. Oliane Maria Correia Magalhães

COORDENADOR DO PROGRAMA DE PÓS-GRADUAÇÃO EM INOVAÇÃO TERAPÊUTICA

Profª. Drª. Maíra Galdino da Rocha Pitta

VICE- COORDENADOR DO PROGRAMA DE PÓS-GRADUAÇÃO EM INOVAÇÃO TERAPÊUTICA

Prof. Dr. Luiz Alberto Lira Soares

DEDICATÓRIA

Dedico este trabalho a minha família e amigos.

AGRADECIMENTOS

- Ao orientador Marcelo Zaldini pelo apoio, paciência e inúmeras sugestões dadas para esta tese.
- A minha família pelo apoio incondicional.
- Ao programa de pós-graduação em inovação terapêutica (PPGIT).
- A Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco (FACEPE) pelo auxilio financeiro a este trabalho através da concessão da bolsa de estudos.

RESUMO

FERREIRA, L. Novas estratégias para métodos *in silico* na inovação terapêutica utilizando computação distribuída: GriDoMol. 2017. Tese (Doutorado). Universidade Federal de Pernambuco, Recife, Pernambuco, Brasil.

Estima-se que o uso de métodos in silico pode reduzir os custos associados ao desenvolvimento de um novo fármaco em até 50%. Esta redução ocorre porque o número de moléculas que precisam ser testadas e sintetizadas experimentalmente passa a ser drasticamente reduzido devido a alta confiabilidade dos métodos computacionais. Porém, estes métodos podem apresentar uma alta demanda computacional quando o número de moléculas a ser testados é alto e quando se busca maior precisão nos resultados numéricos. Sendo assim, este trabalho apresenta o desenvolvimento do programa GriDoMol, uma plataforma unificada para realizar cálculos de docking molecular em um sistema distribuído, através de um grid computacional, com foco em alto desempenho e precisão. Utilizando o GriDoMol e configurações avançadas de docking, foi possível realizar o docking molecular de um conjunto de 213 complexos em um tempo até 9,91 vezes mais rápido e encontrando soluções de docking mais estáveis, com reduções de até 1,3 Kcal/mol, quando comparado com a execução sequencial deste mesmo conjunto em um único computador, utilizando apenas um núcleo de processamento e a configuração padrão de docking. O GriDoMol também oferece a opção de realizar estudos de vacinologia reversa permitindo cálculos de docking molecular entre candidatos a epítopos e alelos de MHCs de Classe I e II humanos com o intuito de encontrar epítopos que possuam uma boa afinidade por estes alelos, aumentando as chances de apresentar uma resposta imunológica significativa.

Palavras-chave: *Docking* Molecular. GriDoMol. Computação Distribuída.

ABSTRACT

FERREIRA, L. New strategies for *in silico* methods in therapeutical inovation using distributed computing: GriDoMol. 2017. Thesis. Universidade Federal de Pernambuco, Recife, Pernambuco, Brazil.

It is estimated that the use of in silico methods can reduce the costs spent at the development stage of a new drug by up to 50%. This happens because the number of molecules that need to be experimentally synthesized and tested becomes drastically reduced due to the high predictability and reliability of the computational methods. Nevertheless, these methods may present a high computational demand when the number of molecules to be tested is high and when it's seeking for a higher precision in the numerical results. Therefore, this work presents the development of the program GriDoMol, a unified platform for performing molecular docking calculations in a distributed system, through a computational grid environment, with focus in high performance and precision. By using GriDoMol and higher docking settings values, it was possible to execute the molecular docking of a set containing 213 complexes in a time up to 9.91 times faster and finding more stable complexes, with energy reduction by up to 1.3 Kcal/mol, when compared with the sequential execution of this same set on a single computer, using a single processor core and the default docking settings values. The program GriDoMol also offers a reverse vaccinology option, allowing the molecular docking of candidate epitopes on selected human MHC's Class I and II alleles, in order to find the most promising epitopes which have a good binding affinity on a large set of alleles and, thus, better chances of having a significant immunogenic response.

Keywords: Molecular Docking. GriDoMol. Distributed Computing.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
ADT	AutoDock Tools (Ferramentas para o AutoDock)
AIDS	Acquired immunodeficiency syndrome (Síndrome da imunodeficiência adquirida)
BoT	Bag of Tasks (Conjunto de tarefas)
CCB	Centro de Ciências Biológicas
CENAPAD	Centro Nacional de Processamento de Alto Desempenho
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
COCOMO	Construtive Cost Model (Modelo de estimativa de tempo de desenvolvimento)
CPU	Central Processing Unit (Unidade central de processamento)
DcFAR	Departamento de ciências Farmacêuticas
DPR	Diferença Percentual Relativa
DRE	Diferença em Relação ao Experimental
WeNMR	Worldwide e-Infrastructure for NMR and structural biology
FACEPE	Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco
FAQ	Frequently Asked Questions (Perguntas mais frequentes)
FDM	Fator de Desempenho por Máquinas
FDN	Fator de Desempenho por Núcleos (*cores*)
FLOPS	FLoating-point Operations Per Second (Operações de ponto flutuante por segundo)
GPU	Graphics Processing Unit (Unidade de processamento gráfico)
GSF	GriDoMol Settings File (Arquivo de configurações de docking do GriDoMol)
GUI	Graphical User Interface (Interface gráfica de usuário)
HIV	Human immunodeficiency virus (Vírus da imunodeficiência humana)
IDE	Integrated Development Environment (ambiente de desenvolvimento integrado)
JDF	Job Description File (Arquivo de descrição das tarefas)
JRE	Java Runtime Environment (Ambiente de execução Java)
JVM	Java Virtual Machine (Máquina Virtual Java)
LGA	Lamarckian Genetic Algorithm (Algoritmo genético lamarckiano)
LNCC	Laboratório Nacional de Computação Científica
LQTM	Laboratório de Química Teórica Medicinal
MHC	Major Histocompatibility Complex (Complexo Principal de Histocompatibilidade)
MPI	Message Passing Interface (Interface para trocas de mensagens)
NMR	Nuclear Magnetic Resonance (Ressonância magnética núclear)
PDB	Protein Data Bank (Banco de dados de proteínas)
PDBQT	Arquivo PDB acrescido de cargas parciais (Q) e número de torções (T)
PPGIT	Programa de Pós-Graduação em Inovação Terapêutica
RAM	Random Access Memory (Memória de acesso randômico)
RMSD	Root Mean Square Deviation (Desvio da raiz média quadrática)
SDF	Structure Data File (Arquivo de dados da estrutura)
SINAPAD	Sistema Nacional de Processamento de Alto Desempenho

UI	User Interface (Interface de usuário)
UFCG	Universidade Federal de Campina Grande
UFPB	Universidade Federal da Paraíba
UFPE	Universidade Federal de Pernambuco
XMPP	eXtensible Messaging and Presence Protocol (Protocolo extensível de presença e mensagens)
WCG	World Community Grid (Grid comunitário mundial)

SUMÁRIO

1 INTRODUÇÃO

Os métodos comumente empregados para obtenção de dados científicos na área de desenvolvimento de fármacos são aqueles classificados como *in vitro* e *in vivo*. O primeiro consiste em realizar estudos das atividades biológicas entre um ligante e uma biomacromolécula fora de um organismo vivo, tipicamente em tubos de ensaio em ambientes controlados, enquanto que o segundo tipo de método está voltado aos experimentos realizados diretamente em modelos experimentais com organismos vivos. Ambos os métodos auxiliam na pesquisa e desenvolvimento de novas moléculas com atividade farmacológica (fármacos em potencial). Porém, se a quantidade de moléculas envolvidas nestes experimentos for elevada, este processo pode se tornar muito oneroso em tempo e custos financeiros.

Com o aumento da capacidade de processamento dos computadores, observado nestes últimos anos, uma nova modalidade de obtenção de dados científicos está ganhando atenção. A modelagem molecular de fármacos está associada ao estudo *in silico* de estruturas e propriedades químicas de moléculas de interesse farmacêutico, de forma a permitir, por exemplo, uma análise entre atividade biológica e propriedades físico-químicas através de procedimentos computacionais. Estes métodos *in silico* estão sendo utilizados de forma crescente como apoio pré-screening aos testes laboratoriais convencionais *in vitro* e *in vivo*, uma vez que através deste tipo de método é possível prever, de forma teórica, o comportamento da estrutura e propriedades moleculares para um grande número de moléculas, antes mesmo delas serem testadas fisicamente.

A importância dos métodos *in silico* já vem sendo percebida pela comunidade científica de todo o mundo e pela grande indústria farmacêutica (“big pharma”). Estima-se que o uso destas metodologias pode reduzir o tempo do projeto e os custos de desenvolvimento de um novo fármaco em até 50% (GELDENHUYS *et. al.*, 2006). Este elevado percentual de economia explica porque grandes indústrias farmacêuticas estão

investindo cada vez mais recursos na aquisição de máquinas poderosas em seus centros de pesquisas, assim como no desenvolvimento de software voltado para este tipo de pesquisa (McGee, 2005). Esta redução de custos e tempo ocorre porque muitas vezes o número de moléculas que precisam ser de fato sintetizadas e testadas experimentalmente pode ser drasticamente reduzido em função do poder de preditividade e confiabilidade dos métodos computacionais ("*in silico*"), abreviando assim o tempo de desenvolvimento de um novo fármaco.

O procedimento de *docking* molecular é um método da modelagem molecular que busca determinar se há interação energética favorável entre duas moléculas (ligante e alvo biológico), no intuito de elucidar as razões moleculares responsáveis pela potência farmacológica destes ligantes. O procedimento de *docking* busca pela posição e orientação que maximiza essas interações intermoleculares entre o ligante e o alvo biológico e, assim, formando um complexo por complementaridade estrutural e por estabilização energética. Um dos principais objetivos dos estudos na área de planejamento *in silico* de fármacos é prever a intensidade e a especificidade com que pequenas moléculas, normalmente denominadas de ligantes (drogas ou fármacos, em potencial) se ligam ao sítio ativo de um receptor alvo, tipicamente uma biomacromolécula (alvo farmacológico), modificando assim o seu ciclo bioquímico/farmacológico. O estudo de docking molecular geralmente ocorre na situação de se conhecer apenas o alvo biológico (receptor), e para descobrir o melhor ligante para um dado receptor é tipicamente necessária a realização de cálculos de docking molecular entre o receptor e um grande número de pequenas moléculas (ligantes) para descobrir quais delas tem maior afinidade para interagir de maneira a formar um complexo estável com o receptor. Desta forma, este método *in silico* pode apresentar, por vezes, uma alta demanda computacional quando o número de ligantes e alvos a serem testados é alto e quando se busca alta precisão nos resultados numéricos obtidos.

Entretanto, como cada cálculo de *docking* molecular entre um ligante e um receptor possui independência lógica e temporal em relação aos outros cálculos de *docking*, pode-se utilizar uma plataforma de computação distribuída, como, por exemplo, grids computacionais, para acelerar a execução do conjunto de cálculos de *docking* molecular ao permitir a execução simultânea (em paralelo) de cálculos individuais de *docking* molecular ao longo de um conjunto de computadores integrantes de um ambiente de *grid* computacional.

A ideia de aplicar tecnologias de computação distribuída para a formação de verdadeiros "laboratórios virtuais" é atualmente muito explorada mundo afora em projetos de diversas áreas do conhecimento humano, inclusive em modelagem molecular. Neste contexto, é possível encontrar iniciativas de estudos de *docking* molecular acelerados por poderosas infraestruturas computacionais (BUYYA et al., 2002; ABREU et al., 2010; ZHANG et al., 2008; JIANG et al., 2008; DOMINGUEZ; BOELENS; BONVIN, 2003; DE VRIES; VAN DIJK; BONVIN, 2010). Desta forma, o tempo total necessário para obtenção dos resultados de aplicações desta natureza pode ser significativamente reduzido, uma vez que cada execução de *docking* molecular será distribuído nas máquinas pertencentes a um *grid* computacional, de forma abstraída para o usuário final, ou seja, o usuário apenas submete os *jobs* (aplicações paralelas contendo as tarefas individuais), enquanto o ambiente de *grid* computacional se encarrega de distribuí-los e executá-los de maneira otimizada através das máquinas que constituem o sistema integrado, passando a impressão que toda a execução está sendo realizada na máquina do usuário.

O objetivo deste trabalho é apresentar a implementação das versões 2.0 (em C++) e 3.0 (em Java) do programa GriDoMol, *software* desenvolvido no Laboratório de Química Teórica Medicinal (LQTM), situado no Departamento de Ciências Farmacêuticas (DCFar) da Universidade Federal de Pernambuco (UFPE), que utiliza a plataforma de *grid* computacional OurGrid (ANDRADE et al., 2003) e os programas de *docking* molecular

AutoDock (MORRIS et al., 2009) e AutoDock Vina (TROTT; OLSON, 2010), com o intuito de disponibilizar uma plataforma unificada com uma interface gráfica fácil e amigável para preparação e realização de *Virtual Screening* com alta performance.

Uma das principais vantagens do programa GriDoMol quando comparado com outras iniciativas de execução de docking molecular de forma distribuída é a utilização do ambiente de grid computacional. O grid computacional permite maior flexibilidade uma vez que os computadores que pertencem a este tipo de arquitetura distribuída não precisam ser dedicados e, por isso, podem ser usados para outros fins. Além disto, o grid computacional permite distribuição geográfica destes computadores, facilitando a criação de poderosas infraestruturas computacionais entre varias instituições de ensino e pesquisa.

Além de permitir o virtual screening tradicional, onde pequenas moléculas candidatas a fármacos são testados em receptors biológicos, também foi desenvolvida no programa GriDoMol uma metodologia para permitir estudos de vacinologia reversa utilizando o framework Rosetta (LEAVER-FAY *et al.*, 2011). Esta metodologia foi criada em nosso laboratório e utilizada inicialmente com o intuito de encontrar candidatos a epítopos com o potencial de gerar uma resposta imunológica significante contra diferentes espécies de Leishmania. A eficácia destes epítopos preditos foi testada e validada através de testes experimentais, por exemplo, com pacientes que foram curados desta doença (E SILVA et al., 2016).

2 REVISÃO DE LITERATURA

2.1 Computação Distribuída

A computação distribuída, ou em paralelo, consiste na utilização de técnicas para unir o poder computacional de vários processadores ou núcleos de processamento para a execução de uma tarefa complexa, que esteja relacionada a uma demanda computacional elevada. Pode-se aplicar paralelismo, essencialmente, de duas formas distintas: unindo o poder computacional entre dois ou mais computadores, ou utilizando o paralelismo interno disponível em um mesmo computador, por causa de seus vários núcleos de processamento.

Os principais tipos de paralelismo que são largamente utilizados são o c*luster e o grid c*omputacional. Embora ambos formem uma arquitetura distribuída unindo diversos computadores para resolução de problemas de alta demanda computacional, há algumas características que os diferenciam.

O ambiente distribuído do tipo *cluster*, comumente denominado supercomputador, geralmente é formado por computadores homogêneos, com mesma configuração de hardware e software, interligados por uma rede local de alta velocidade. Todo o poder de processamento desta plataforma é dedicado a um computador central (denominado *front-*end), responsável por gerenciar a execução dos cálculos ao longo dos computadores disponíveis no *cluster* (SADASHIV; KUMAR, 2011).

O ambiente distribuído do tipo *grid* computacional pode ser formado por computadores heterogêneos, com diferente configuração de *software* e *hardware*, interligados através de uma rede local e/ou internet. Ao contrário do cluster, apenas a parte ociosa do poder de processamento dos computadores que integram o ambiente de grid computacional é utilizado. Geralmente, os ambientes de *grid* computacionais são formados em colaboração voluntária entre varias instituições, onde cada uma delas cede

o poder de processamento que não estiver sendo utilizado no momento (ocioso), para outros membros do grid computacional poderem utilizá-lo de forma simples e automática. Desta forma, pode-se formar uma poderosa infraestrutura computacional por uma fração do custo de manter um *cluster* dedicado (SADASHIV; KUMAR, 2011).

Um dos ambientes de grid computacional mais conhecidos e utilizados no mundo é o BOINC (BOINC, 2016), desenvolvido pela Universidade de Berkeley. Por se tratar de um *grid* computacional voluntário, seu poder de processamento varia bastante dependendo do dia e hora, situando-se atualmente (novembro de 2016) em torno 13 petaFLOPS (quadrilhões de operações de ponto flutuante por segundo). Este poder de processamento está acima do obtido pelo quinto supercomputador (*cluster*) mais rápido do mundo, o *K Computer* (TOP500 Supercomputers, 2016). Este supercomputador, que utiliza processadores SPARC64 da Fujitsu, possui 705.024 núcleos de processamento e tem o poder computacional de 10,5 petaFLOPS (TOP500 Supercomputers, 2016). Em uma análise mais atualizada, o supercomputador mais rápido do mundo, nos dias de hoje, é o Sunway TaihuLight (TOP500 Supercomputers, 2016). Este supercomputador, que utiliza processadores Sunway, de fabricação chinesa, possui 10.649.600 núcleos de processamento e seu poder computacional é de aproximadamente 93 petaFLOPS (TOP500 Supercomputers, 2016), ou seja, mais ou menos a capacidade computacional de 7 grids computacionais como o BOINC.

Como a infraestrutura computacional de um ambiente de *grid* computacional é geralmente proporcionada por doações voluntárias dos ciclos de processamento ociosos das máquinas dos voluntários, em suas casas ou ambientes de trabalho, os custos de manutenção da infra-estrutura e consumo energético destas máquinas são arcados por estes próprios voluntários, tornando o acesso a este poder computacional incomparavelmente muito mais barato do que manter um supercomputador em instalações próprias, com todos os gastos envolvidos neste tipo de infra-estrutura

(incluindo refrigeração com ar-condicionado, no-breaks, geradores, etc.). Vale a pena ressaltar que, na maioria dos casos, *grids* computacionais são acessíveis a qualquer pessoa ou usuário que tenha interesse em participar deste tipo de ambiente distribuído, ao contrário de muitos supercomputadores que são de uso restrito de empresas ou outras instituições públicas e privadas.

O principal tipo de paralelismo encontrado no nível interno ao computador é o *multithreading*. Programas que foram desenvolvidos para utilizarem esta estratégia de paralelismo separam suas rotinas computacionais em vários módulos, os quais podem ser executados simultaneamente fazendo-se uso dos vários núcleos de processamento (CPUs) disponíveis em processadores modernos (*dual-core*, *quad-core*, octa-core, etc.). Um bom exemplo, que utiliza esta estratégia em processadores modernos para aumentar o desempenho computacional da aplicação, é o programa AutoDock Vina (TROTT; OLSON, 2010), detalhado na seção 2.1.1.2 deste capítulo.

Uma abordagem similar ocorre com as placas gráficas atuais. Observa-se uma tendência nas placas gráficas atuais em utilizar vários núcleos de processamento gráfico, comumente denominado GPUs, com o intuito de acelerar a reprodução de vídeos e jogos em alta resolução. Acontece que também é possível obter acesso ao poder de processamento deste tipo de arquitetura distribuída para processamento numérico de alto desempenho. Ao utilizar bibliotecas especificas disponíveis para algumas placas de vídeo, como, por exemplo, CUDA (CUDA, 2016) e o OpenCL (OpenCL, 2016), é possível desenvolver programas que utilizem os núcleos de processamento das placas gráficas para processamento sem ser, necessariamente, processamento gráfico. Para exemplificar o poder computacional deste tipo de plataforma distribuída, o supercomputador Titan, atualmente situado na terceira posição como supercomputador mais poderoso do mundo em operação, utiliza GPUs em sua composição. Sua arquitetura híbrida conta com 18.688 nós (computadores), onde cada um deles contém um

processador AMD Opteron e uma placa de vídeo Nvidia Tesla K20, totalizando 560.640 núcleos. O Titan já foi o supercomputador mais poderoso do mundo em operação em novembro do ano de 2012 (TOP500 Supercomputers, 2016). Na seção 2.3.3 deste capítulo, serão apresentadas diversas iniciativas que utilizam este tipo de abordagem para acelerar a execução de programas voltados à área de pesquisa relacionada com modelagem molecular.

Devido a sua relevância neste presente trabalho, iremos detalhar os conceitos sobre *grid* computacional e sua aplicabilidade no meio científico.

2.1.1 GRIDS COMPUTACIONAIS

As principais características que diferem um ambiente de grid computacional de outras arquiteturas distribuídas são sua heterogeneidade (capacidade de interagir entre diversos sistemas operacionais e arquiteturas de *hardware* diferentes), e sua possibilidade de dispersão geográfica, permitindo que seus recursos possam estar distribuídos em todo mundo (CIRNE, 2002; SMITH, 2004). A idéia por traz da utilização de ambientes de *grid* computacional é realizar tarefas complexas que são inviáveis de serem executadas em apenas uma única máquina. Desta forma, é possível obter um poder computacional muito grande, de forma simplificada para o usuário do *grid* computacional.

Empresas e instituições que fazem parte de um ambiente de *grid* computacional doam ciclos de processamento de seus computadores em período ocioso para, quando precisarem, poderem ter acesso a mais recursos dos outros membros do *grid* computacional. Apesar do interesse principal no compartilhamento de capacidade de processamento no grid computacional, podem existir ainda outras modalidades de compartilhamento de recursos, como por exemplo, recursos de armazenamento (*storage*) ou uso de periféricos, como impressoras, ploters, sensores, entre outros (CIRNE, 2002). Uma das principais características de um *grid* computacional é abstrair, principalmente

para o usuário mais leigo, os detalhes técnicos quando este utilizar os recursos do *grid* que podem estar, inclusive, distribuídos geograficamente. Outra característica diferenciada do *grid* computacional é a ausência de um controle central para todo o ambiente de *grid* computacional. Um exemplo que expressa a idéia de um *grid* computacional é a própria internet, onde são utilizados recursos de servidores externos para a obtenção de dados ou processamento (CIRNE, 2002). Não é preciso saber o caminho feito a cada roteador e nem a localização física de cada servidor para usufruir das funcionalidades destes vários servidores na internet. Além disto, como dito no começo do capitulo, a utilização de *grids* computacionais pode significar uma redução significativa de custos, quando comparado com outras arquiteturas distribuídas, uma vez que a manutenção dos computadores pertencentes às instituições que participam deste ambiente distribuído é custeada pelas próprias instituições, distribuindo, assim, o custo de manter este tipo de infraestrutura em operação contínua.

Entretanto, a utilização de grids computacionais também apresenta algumas desvantagens quando comparado com outras arquiteturas distribuídas. Uma destas desvantagens é que, como as máquinas podem estar distribuídas geograficamente, não há garantias de que a tarefa enviada para ser executada remotamente, fora do ambiente de grid computacional local, será realizada uma vez que pode haver problemas de conexão com a internet ou desligamento da máquina remota, situação que não acontece em arquiteturas distribuídas locais, onde todas as máquinas estão no mesmo ambiente físico e, portanto, não há a necessidade de conectar as máquinas do grid computacional através da internet, além de ser mais fácil identificar com rapidez casos onde uma máquina tenha sido desligada. Além disto, como o grid computacional utiliza o poder de processamento ocioso das máquinas que compõe o grid computacional, nem sempre as tarefas enviadas para as máquinas do grid computacional estarão sendo executadas utilizando 100% do poder computacional uma vez que estas máquinas não são dedicadas

e, se algum usuário estiver utilizando a máquina para outros fins no momento que a tarefa é enviada, apenas o poder computacional ocioso será utilizado, situação que não ocorre em outros sistemas distribuídos tal como, clusters, por exemplo, onde todas as máquinas dedicam 100% de seu poder computacional à máquina principal. Outra desvantagem é relacionada à segurança uma vez que os arquivos armazenados nas máquinas remota, que podem estar em ambientes físico de outras instituições, podem ter o conteúdo facilmente identificado, inclusive os resultados da execução da tarefa. Porém, estas desvantagens são poucas quando comparadas com as vantagens de utilizar o ambiente de grid computacional para suprir a crescente demanda computacional, principalmente quando há confiança entre as instituições que integram o ambiente de grid computacional.

2.1.1.1 OurGrid

O projeto OurGrid é uma iniciativa nacional de código aberto (*open source*) de implementação de um ambiente de *grid* computacional, desenvolvido na Universidade Federal de Campina Grande (UFCG), desde 2003 (ANDRADE et al., 2003) e é bastante utilizado por outras iniciativas de computação distribuída encontradas na comunidade científica e de possuir uma ampla gama de aplicabilidade (IRTAZA; JAFFAR; MAHMOOD, 2014; SILVA et al., 2014; ANGLANO; CANONICO; GUAZZONE, 2010). Os componentes da plataforma OurGrid possuem versões para os principais sistemas operacionais, como Windows, Mac OS e Linux. Os três componentes responsáveis pela gestão e comunicação do ambiente OurGrid são os componentes Broker, Peer e Worker (ver Tabela 1). Para realizar a comunicação entre os componentes da plataforma OurGrid, é necessário utilizar o protocolo XMPP. Este protocolo detecta a presença destes componentes e possibilita a troca de mensagens entre eles.

Tabela 1 – Descrição dos principais componentes do OurGrid utilizados na formação de um ambiente de grid computacional típico.

Componente	Descrição
Broker	Responsável por distribuir as tarefas (*tasks*) ao longo dos computadores disponíveis no ambiente de *grid* computacional. Principal forma de interação entre o usuário e o *grid* computacional.
Peer	Responsável por gerenciar os computadores pertencentes ao *grid* computacional de certa empresa ou instituição. Este componente identifica e disponibiliza os computadores para a execução remota de uma dada tarefa. Desta forma, ele é responsável por autorizar e realizar a comunicação entre *Brokers* e *Workers*.
Worker	Responsável pela execução, propriamente dita, das tarefas recebidas remotamente. Para isso, este componente recebe os arquivos de entrada e o programa que será utilizado para realizar o processamento, e posteriormente retorna o resultado deste processamento de volta para a máquina do usuário que requisitou a execução. Este componente está presente em cada computador integrante do *grid* computacional que esteja apto para ceder seus recursos computacionais.

Um exemplo de como se forma um ambiente de grid computacional entre várias instituições utilizando a plataforma OurGrid pode ser observado na Figura 1.

O modelo de execução de tarefas na plataforma OurGrid é o *Bag of Tasks* (*BoT*). Este modelo consiste na distribuição de um conjunto de tarefas (*jobs*) ao longo dos computadores pertencentes ao grid computacional, onde cada tarefa (*task*) não possui qualquer tipo de dependência lógica ou temporal em relação às outras tarefas do conjunto. Desta forma, a execução de cada tarefa não depende da comunicação ou estado de outra tarefa do conjunto.

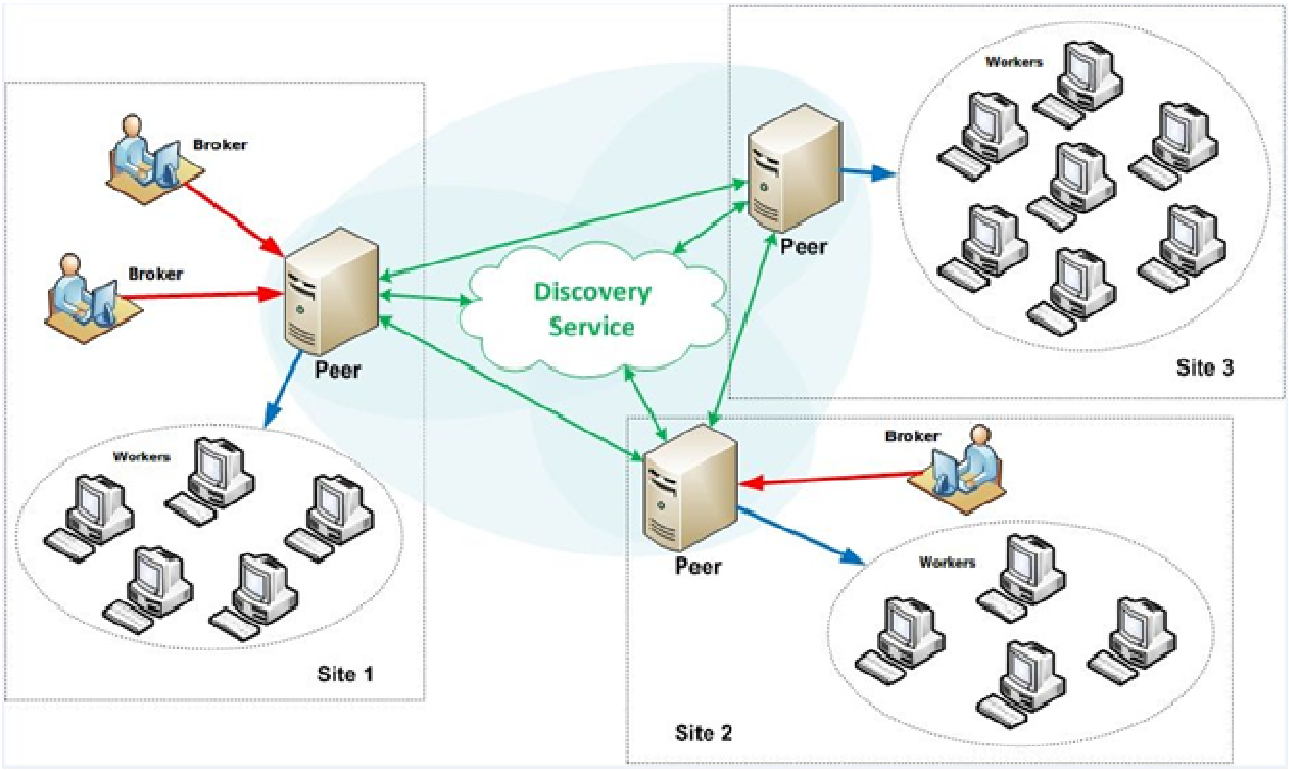


Figura 1 – Exemplo de um ambiente de grid computacional formado pela plataforma OurGrid. Neste exemplo, há três ambientes de grid computacionais locais (Sites 1, 2 e 3) que se comunicam entre si através de seus respectivos Peers com o intuito de formar um grid computacional unificado. extraído de http://www.isgtw.org/feature/grid-cooperative. Acessado em Fevereiro de 2017.

Para realizar a execução destes conjuntos de cálculos (*jobs*), o OurGrid utiliza uma estrutura de arquivo chamada *Job Description File* (JDF), na qual são definidos os arquivos de entrada (*input*) e os de saída (*output*), bem como a linha de comando de execução, e os parâmetros que serão utilizados. Basicamente, cada *task* é subdividida em três partes (*init*, *remote* e *final)*, as quais indicam à máquina remota como proceder de acordo com a fase em que a tarefa (*task*) se encontra (ver Tabela 2).

Antes de definir as tarefas que serão realizadas no ambiente de *grid* computacional, é possível indicar os requisitos mínimos necessários que o computador remoto precisa dispor para realizar a execução do conjunto de tarefas, como por exemplo: qual o sistema operacional necessário, qual a quantidade de memória RAM demandada, qual o espaço requisitado em disco, dentre outros.

Tabela 2 – Ciclo de execução de uma tarefa na plataforma OurGrid.

Fase	Descrição
Init	Fase inicial da tarefa. Nela, os arquivos necessários para a execução de uma determinada tarefa são transferidos para o computador remoto.
Remote	Fase de execução da tarefa. Consiste na execução do programa e seus parâmetros de entrada.
Final	Fase final da tarefa. Nesta fase, os arquivos de saída gerados pelo processamento da etapa anterior (*remote*) são transferidos para o computador do usuário que requisitou a tarefa.

2.1.1.2 Boinc

A plataforma de computação voluntária BOINC (BOINC, 2016) é um dos ambientes de grid computacional mais conhecidos no mundo, com mais de 230 mil voluntários ativos, que permite aos voluntários doarem o poder de processamento ocioso de seus computadores, *smartphones* e *tablets* a uma lista de projetos científicos disponíveis na plataforma BOINC. Além da possibilidade de fornecer o poder computacional voluntário, há também a opção de criar novos projetos dentro da plataforma BOINC e, desta forma, se beneficiar do poder computacional de outros usuários.

A escolha entre quais projetos científicos receberão o poder computacional doado fica a critério do próprio usuário. Ao instalar o programa em seus dispositivos, uma lista contendo os detalhes de cada projeto é exibida ao usuário. Além da estratégia de paralelismo através da utilização de *grids* computacionais, alguns projetos disponíveis na plataforma BOINC utilizam mais uma estratégia de paralelismo através da utilização dos núcleos de processamento gráfico (GPUs). Por exemplo, na Figura 2 é possível observar que o *World Community Grid* (WCG, 2016), um dos projetos científicos mais populares desta plataforma, possui suporte aos sistemas operacionais Windows, Mac, Linux e Android (*smartphones* e *tablets*), além de oferecer suporte à tecnologias de processamento em paralelo nas unidades gráficas de processamento disponíveis em alguns modelos de placas de vídeo das empresas Nvidia e ATI. O *World Community Grid*

e outros projetos que utilizam modelagem molecular em *grids* computacionais serão aprofundados na seção 2.3.2 deste capítulo.

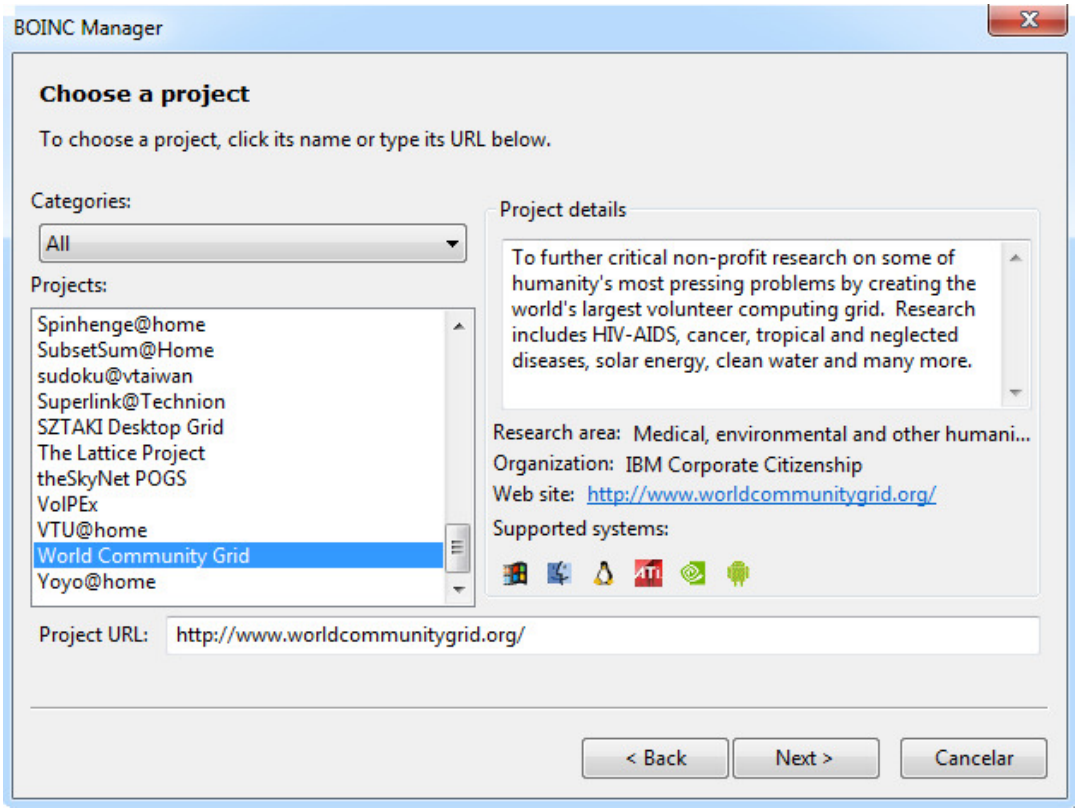


Figura 2 – Tela do programa BOINC para seleção de projetos que receberão os ciclos de processamento ociosos do computador do usuário.

Para incentivar a colaboração, é disponibilizado um *website* com uma tabela de pontuação e classificação dos usuários, indicando quais colaboraram mais ativamente no desempenho do *grid* computacional e seus respectivos projetos.

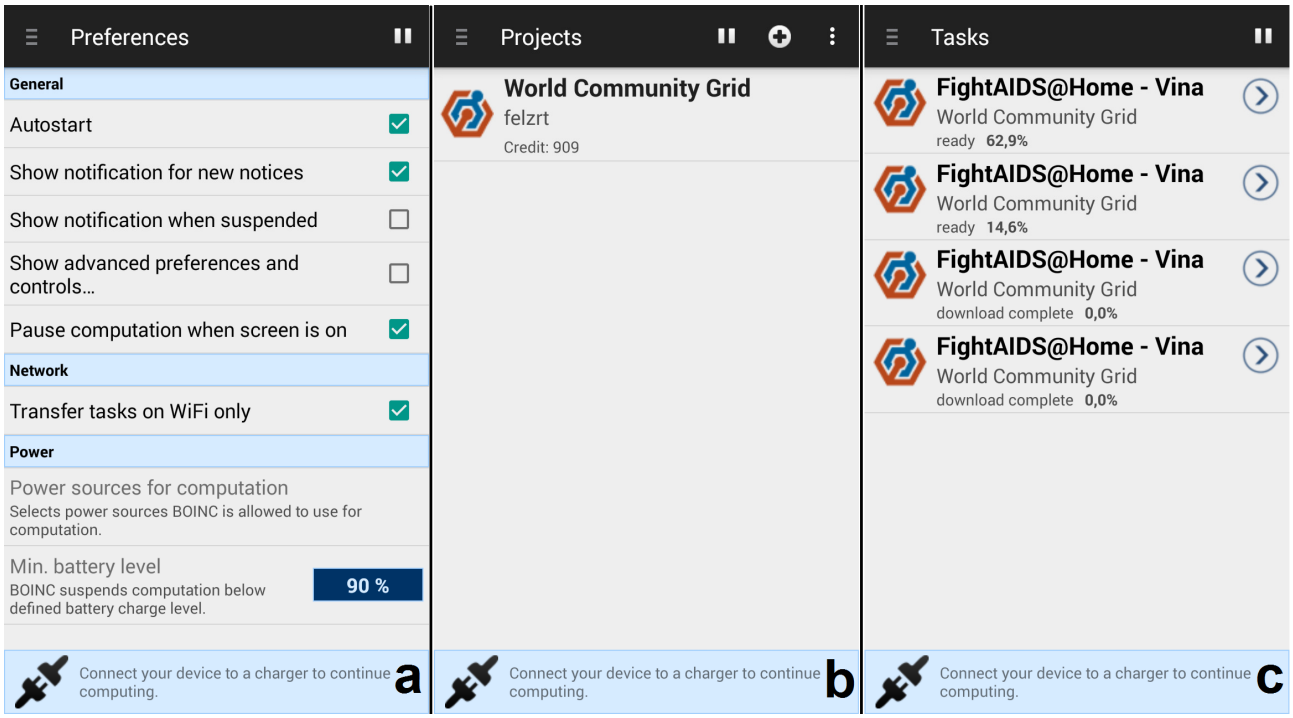


Figura 3 - Telas do aplicativo BOINC oficial disponível para dispositivos que utilizam o sistema operacional Android. a) Tela de configuração. b) Tela contendo os projetos. c) Tela contendo a progressão atual das tarefas.

O aplicativo BOINC para o sistema operacional Android (ver Figura 3) permite que o voluntário doe o poder computacional de seus *smartphones* e *tablets* durante o período em que os mesmos estiverem ociosos. O voluntário pode ajustar as condições para que as tarefas sejam executadas (ver Figura 3a) tal como, por exemplo, o nível mínimo de bateria em que é permitido o início da execução das tarefas neste dispositivo. Além disto, através deste aplicativo móvel o usuário pode acompanhar sua pontuação (*credit*, ver Figura 3b) obtida ao doar poder de processamento de seus computadores e dispositivos Android para cada projeto e também a progressão individual de cada tarefa enviada a este dispositivo Android (ver Figura 3c). A ideia por traz deste aplicativo é utilizar o poder de processamento ocioso durante o período noturno, em que o *smartphone* está sendo carregado, para receber e processar pequenos pacotes de dados científicos e enviar os resultados para os seus respectivos projetos. A utilização destes aplicativos não reduz de

forma significativa o tempo de vida útil da bateria (Android FAQ BOINC, 2016) e aproveita o poder computacional que, de outra forma, seria desperdiçado.

2.2 Modelagem Molecular

Uma das formas de definir modelagem molecular é o estudo das estruturas e propriedades químicas de sistemas atômico-moleculares. Suas aplicações vão desde a criação/edição (HANWELL et al., 2012; BIENFAIT; ERTL, 2013; MolCalc, 2016; Marvin JS, 2016) e visualização (DELANO, 2004;2009; SAYLE; MILNERWHITE, 1995; STIERAND; RAREY, 2010) de moléculas, até a otimização de geometrias (PURANEN; VAINIO; JOHNSON, 2010; HESS et al., 2008) e interações entre alvos biológicos e seus potenciais ligantes, como é o caso do *docking* molecular (GOODSELL; MORRIS; OLSON, 1996; MORRIS et al., 2009; TROTT; OLSON, 2010), por exemplo.

A utilização destes métodos podem reduzir significativamente o tempo e os custos necessários para o desenvolvimento de novos fármacos (GELDENHUYS et al., 2006). Um exemplo da importância destes estudos é que a utilização desta abordagem ajudou a encontrar inibidores do HIV (WLODAWER, 2002; DESJARLAIS et al., 1990), do gene responsável por um tipo de câncer, o LSD1 (VENKATASWAMY et al., 2013) e inibidores da proteína matricial VP40 do vírus Ebola (TAMILVANAN; HOPPER, 2013).

2.2.1 DOCKING MOLECULAR

O *docking* molecular é uma das abordagens da modelagem molecular que é útil para determinar se há interação energética favorável entre duas moléculas (tipicamente um ligante e um receptor), no intuito de elucidar as razões moleculares responsáveis pela potência farmacológica destes ligantes ou fármacos em potencial. O procedimento de *docking* molecular busca pela posição e orientação que maximiza essas interações intermoleculares. Assim, o ligante e o alvo biológico formam um complexo por

complementaridade estrutural e por estabilização energética.

Um dos principais objetivos dos estudos na área de planejamento "*in silico*" de fármacos é prever a intensidade e a especificidade com que pequenas e médias moléculas, normalmente denominadas de ligantes (drogas ou fármacos, em potencial) se liguem ao sítio ativo de um alvo receptor, tipicamente uma biomacromolécula (alvo farmacológico), modificando assim o seu ciclo bioquímico/farmacológico (LENGAUER; RAREY, 1996). Em termos gerais, esta modificação pode ser tipicamente de dois tipos: i) agonista, ativando este receptor ou alvo farmacológico, para que este desempenhe sua atividade biológica; ii) antagonista, de forma que inibe a ação do receptor farmacológico, evitando que ele desempenhe sua atividade biológica, como, por exemplo, bloquear o sitio ativo de uma enzima com o intuito de evitar que esta interaja com seu substrato natural. A decisão sobre inibir ou ativar uma enzima que é eleita como potencial alvo farmacológico, por exemplo, vai depender do mecanismo farmacológico e do propósito terapêutico.

A busca pela complementaridade estrutural ligante-receptor é amplamente associada à ideia da "chave-fechadura" (FISCHER, 1894), onde apenas a chave apropriada está apta a se encaixar na fechadura (ver Figura 4). Um avanço do modelo "chave-fechadura", de Fischer, foi proposto por Koshland (KOSHLAND, 1958), que considera um "ajuste induzido" do receptor (enzima, por exemplo), causado pela presença de uma molécula diferente do seu substrato natural, tipicamente um modulador artificial candidato a fármaco, por exemplo.

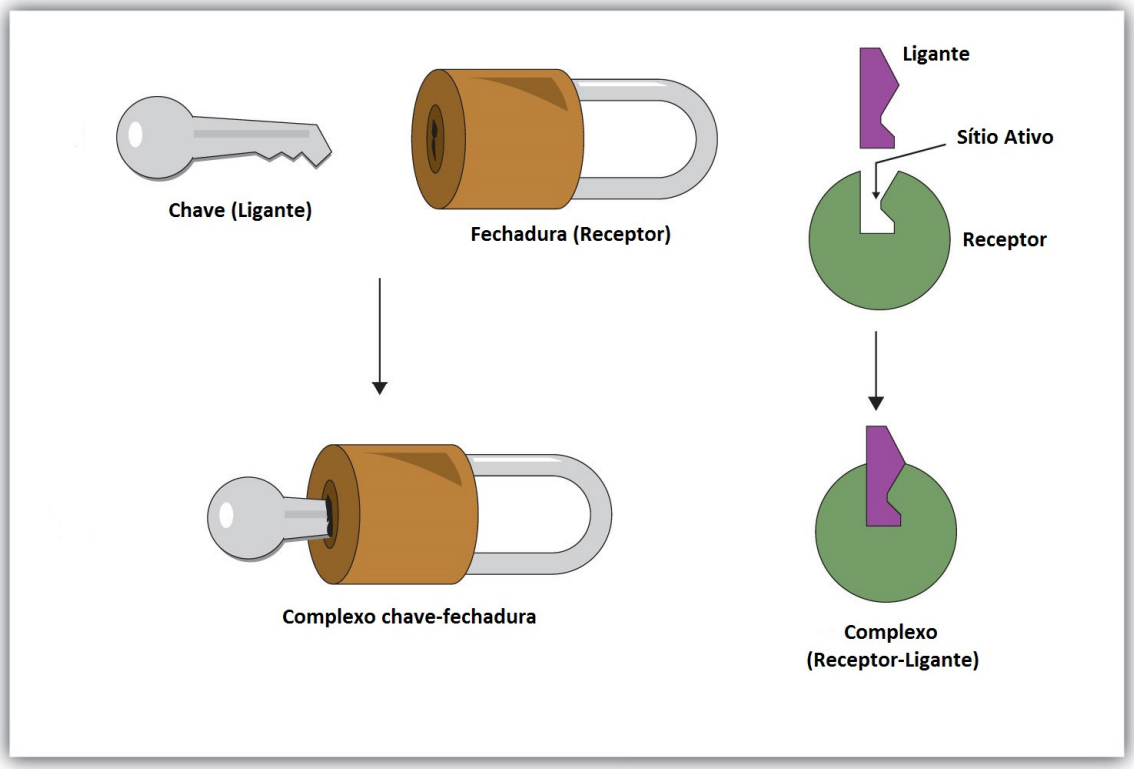


Figura 4 – Exemplo do modelo chave-fechadura proposto por Emil Fischer. Adaptado de https://biochem1362blog.wordpress.com/2014/02/22/162/. Acessado em Fevereiro de 2017.

Para determinar as melhores soluções, os programas de *docking* utilizam uma função de pontuação (score). Nos programas de *docking* molecular que utilizam uma função de pontuação baseada em campos de força (*force field*), como é o caso dos programas AutoDock e AutoDock Vina, tal estabilidade do complexo receptor-ligante é medida pela intensidade da energia de interação, através do mapeamento das forças intermoleculares envolvidas na ligação entre os átomos do ligante e os átomos do sítio ativo do receptor. Neste caso, quanto menor (mais negativa) for a energia de interação, maior as forças atrativas que estão agindo entre o receptor e o ligante e, consequentemente, maior é a estabilidade do complexo (receptor + ligante), e normalmente mais promissor é o candidato a fármaco (ligante).

O *docking* molecular é um excelente método para avaliar os possíveis ligantes de um alvo biológico de interesse. Ao determinar quais moléculas demonstram mais

afinidade pelo alvo farmacológico, é possível reduzir drasticamente o número de moléculas que serão sintetizadas e testadas (*in vitro* e *in vivo*) em laboratório, economizando tempo e recursos financeiros (GELDENHUYS et al., 2006).

2.2.1.1 AutoDock

O AutoDock é um programa de *docking* molecular de código aberto (*open source*), baseado em campos de força (*force fields*), que busca obter a energia de interação mínima para o complexo ligante-receptor, explorando os principais graus de liberdade das moléculas de interesse, incluindo a flexibilidade dos ligantes e dos aminoácidos localizados na região do sítio ativo do alvo biológico (GOODSELL; MORRIS; OLSON, 1996; MORRIS et al., 2009). O procedimento de *docking* molecular com o programa AutoDock é dividido em duas etapas, realizadas por módulos distintos, sendo eles o AutoGrid e o AutoDock.

Antes de realizar o cálculo de *docking* molecular com o AutoDock, é necessário pré-calcular os mapas de energia de interação para os tipos de átomos que compõe os ligantes de interesse (candidatos à fármacos), através da utilização do programa AutoGrid (que faz parte do pacote de programas do AutoDock). Este procedimento consiste em criar uma grade tridimensional com pontos em coordenadas cartesianas específicas ao redor da região delimitada como sitio ativo do alvo farmacológico (receptor), e fazer a sondagem dos tipos de átomos do ligante em cada um destes pontos, com a finalidade de conseguir determinar quais são as interações entre aquele tipo de átomo do ligante e os átomos presentes no sitio ativo do receptor. Por exemplo, se o ligante possui um átomo de carbono do tipo aromático e um oxigênio aceitador de ligação de hidrogênio, estes átomos serão usados na sondagem realizada através da varredura tridimensional pela região do sitio ativo, e as energias de interação de cada um destes átomos com os átomos do sitio ativo do receptor serão mapeadas (gerando os arquivos A.map e OA.map,

respectivamente). O mapeamento *in silico* para o potencial eletrostático também é realizado através do módulo AutoGrid. O objetivo deste pré-cálculo com o módulo AutoGrid é tornar o cálculo de *docking* mais rápido e eficiente como um todo, uma vez que não haverá a necessidade de recalcular as energias de interação dos átomos envolvidos durante o processo de *docking, que é* realizado efetivamente, em seguida, pelo módulo AutoDock.

O programa AutoDock utiliza algoritmos genéticos na busca pelo complexo mais estável, sendo o mais conhecido o LGA ("Lamarckian Genetic Algorithm"). Ao modificar os valores atribuídos aos parâmetros do algoritmo genético utilizado pelo AutoDock, é possível, para o usuário, tornar os cálculos de *docking* mais robustos e precisos, porém com uma maior demanda computacional para cada cálculo individual de *docking* molecular, chegando a impactar o tempo necessário para o término do processo em até algumas ordens de grandeza a mais.

2.2.1.2 AutoDock Vina

O programa AutoDock Vina (TROTT; OLSON, 2010) é uma versão adaptada do programa de *docking* molecular AutoDock (GOODSELL; MORRIS; OLSON, 1996; MORRIS et al., 2009), desenvolvido no mesmo instituto (The Scripps Research, 2016), com o objetivo de ser simples, rápido e com alta preditividade. Assim como o AutoDock, a função de pontuação do AutoDock Vina é baseada em campos de força (*force fields*) e permite levar em conta no processo de docking importantes graus de liberdade no ligante e em resíduos pré-selecionados do receptor.

O processo de *docking* molecular deste programa se tornou mais simples, automático e intuitivo, em comparação ao do AutoDock, uma vez que requer apenas as estruturas das moléculas que serão usadas no cálculo de *docking* molecular, dispensando

a etapa do cálculo com o módulo AutoGrid.

A estratégia de paralelismo utilizada pelo AutoDock Vina consiste em distribuir o número de buscas que serão realizadas pelos seus algoritmos genéticos nos diversos núcleos de processamento do computador ("multithreading"). Assim, cada núcleo de processamento realiza a evolução das populações em uma determinada região específica do domínio de buscas e no final, os complexos (receptor + ligante) mais estáveis encontrados durante a evolução das populações geradas pelo algoritmo genético, em cada subárea, são reunidos e ordenados de forma crescente em relação aos valores de energias de interação calculados. Esta estratégia de paralelismo torna o programa AutoDock Vina consideravelmente rápido, obtendo um desempenho até 62 vezes mais rápido quando comparado com o AutoDock (TROTT; OLSON, 2010).

Várias das iniciativas do projeto *World Community Grid* (WCG, 2016) usam o programa AutoDock Vina na busca por novos fármacos, por exemplo, para combater doenças como a AIDS e Ebola (ver seção 2.3.2.1).

Os programas de docking molecular AutoDock e AutoDock Via, embora desenvolvidos pelo mesmo instituto de pesquisa, diferem em suas características. Uma das principais vantagens do programa AutoDock é que este programa permite que o usuário avançado realize customizações avançadas em seu algoritmo de buscas, com o intuito de aumentar as chances de obter uma melhor solução de docking ao custo de uma demanda computacional mais elevada. Já o programa AutoDock Vina é uma versão modificada do programa AutoDock com o intuito de ser mais amigável aos usuários menos experientes, uma vez que necessita apenas das estruturas das moléculas que serão utilizadas (ligante e alvo), além das coordenadas do centro e as dimensões do sítio ativo. Além disso, o programa AutoDock Vina possui capacidade interna, já implementada, de utilizar paralelamente os diversos núcleos de processamento dos processadores (CPUs) em seu algoritmo genético, dividindo o espaço de busca de soluções entre os

núcleos existentes, com o objetivo de acelerar a realização dos cálculos de *docking* molecular. Desta forma, o programa AutoDock Vina é tipicamente mais rápido quando comparado com outras soluções de *docking* molecular, inclusive com o próprio AutoDock.

2.3 Computação Distribuída Aplicada à Modelagem Molecular

2.3.1 INFRAESTRUTURAS BASEADAS EM CLUSTERS

2.3.1.1 Mola

O sistema Mola (ABREU et al., 2010) é uma plataforma para execução de *docking* molecular em larga escala através da utilização de um pequeno ambiente computacional de *cluster* heterogêneo.

A integração entre os computadores para formar o ambiente *cluster* ocorre através da execução de uma distribuição específica do sistema operacional Linux, disponível em um live-CD. Sistemas operacionais disponibilizados em live-CD não necessitam de instalação, facilitando a integração e, posteriormente, separação dos computadores neste ambiente *cluster,* não havendo necessidade de qualquer modificação no sistema operacional original previamente instalado em cada um dos computadores utilizados no ambiente *cluster.* Este sistema operacional, que é executado no computador central que será usado para controlar os outros integrantes do ambiente *cluster*, gerencia os recursos disponíveis através dos protocolos LAM-MPI (descontinuado), Local Area Multicomputer MPI, e o MPICH (MPICH, 2016), MPI CHameleon, ambos baseados no MPI (*Message Parsing Interface*), um padrão para comunicação de dados em sistemas distribuídos.

Os programas de *docking* molecular disponíveis neste sistema são o AutoDock (GOODSELL; MORRIS; OLSON, 1996) e o AutoDock Vina (TROTT; OLSON, 2010), ambos desenvolvidos pelo *The Scripps Research Institute* (The Scripps Research, 2016). É possível configurar os ajustes (*setups*) de *docking* molecular específicos de cada

programa, tais como os parâmetros do algoritmo genético do programa AutoDock e a variável *exhaustiveness*, responsável pelo número de execuções da etapa de refinamento parcial dos cálculos individuais, no caso do programa AutoDock Vina.

Com o intuito de aproveitar todos os núcleos de processamento dos computadores durante a execução do programa AutoDock, o qual não possui em seu código fonte funções para fazer uso do paralelismo a nível de CPUs, durante a execução dos cálculos de *docking* molecular, o sistema Mola envia múltiplas instâncias de execução para cada computador, sendo uma instância de execução do programa AutoDock para cada CPU. No caso do AutoDock Vina, apenas uma instância é enviada por computador, uma vez que este programa é otimizado para utilizar todos os núcleos de processamento ("multithreading") disponíveis em cada computador. Novas tarefas são atribuídas aos computadores apenas quando as execuções anteriores finalizam, evitando-se assim, sobrecarregar computadores mais lentos e mantendo o *cluster* estável.

Ao utilizar 5 máquinas *dual-core*, totalizando 10 CPUs, este sistema foi capaz de realizar o *docking* molecular de 237 ligantes por dia para o programa AutoDock e 575 ligantes por dia para o programa AutoDock Vina.

2.3.1.2 Haddock

O sistema Haddock (DOMINGUEZ; BOELENS; BONVIN, 2003; VAN ZUNDERT *et al*., 2016) é um servidor web para *docking* molecular cuja função de pontuação ("score") é baseada no conhecimento de dados experimentais (*knowledge based*) das moléculas. Seu método de *docking* molecular consiste na minimização de energia do corpo rígido, com refinamento semi-flexível dos ângulos de torção, além de um refinamento final com solvente explicito. Todo o processamento é feito no ambiente *cluster* dedicado a este projeto e conta também com acesso ao ambiente de *grid* computacional europeu We-NMR (We-NMR, 2016), caso necessite de mais poder computacional. Atualmente, o

servidor HADDOCK conta com mais de 6.000 usuários e já executou mais de 100.000 conjunto de cálculos de docking (VAN ZUNDERT *et al*., 2016).

Há três interfaces gráficas distintas disponíveis para a interação entre o usuário e o servidor. A interface “Fácil” é a mais simples disponível. Nela, o usuário precisa apenas fornecer os arquivos contendo as estruturas moleculares e a lista dos resíduos do sitio ativo que interagem com o ligante. A interface “Especialista” permite definir segmentos flexíveis, distâncias entre os átomos e o estado de protonação dos resíduos. A interface “Guru” permite alterações em configurações avançadas que podem afetar de forma significativa os resultados dos cálculos de *docking* molecular, tal como constantes de força, temperatura e os pesos ajustados para os termos individuais da função de pontuação.

2.3.1.3 Dovis

O Dovis (ZHANG *et al*., 2008) é um sistema que utiliza um ambiente *cluster* para realizar *docking* molecular em larga escala, através de uma interface gráfica. O programa de *docking* molecular utilizado neste estudo foi o AutoDock (GOODSELL; MORRIS; OLSON, 1996).

Em sua versão 2.0 (JIANG *et al*., 2008), com o intuito de aumentar a performance, o Dovis teve seu algoritmo de escalonamento modificado para dinâmico, permitindo o balanceamento do *cluster* em tempo real. Além disso, o código fonte do programa AutoDock também foi modificado para reduzir os acessos ao disco rígido para a escrita dos arquivos de resultados durante o procedimento de *docking* molecular com este programa, uma vez que o *cluster* em questão utilizava um disco rígido comum a todas as máquinas, e o excesso de escrita simultânea neste disco sobrecarregava-o, tornando-o mais lento, e consequentemente os cálculos de *docking* molecular demoravam mais.

Com o intuito de classificar as soluções de *docking* molecular mais favoráveis

dentre as encontradas nos cálculos, os programas de *docking* molecular utilizam uma função de pontuação, comumente conhecida como *score*, e que, geralmente, é única para cada programa. O Dovis 2.0 permite que o usuário utilize uma função de pontuação diferente da empregada no programa de *docking* molecular selecionado, procedimento este conhecido pelo nome de *re-score.* Com isso, pode-se gerar uma nova classificação para os resultados obtidos. Além disto, existe a opção de converter os resultados finais para o formato SDF, formato amplamente utilizado por programas populares de visualização e interpretação de resultados, na área de modelagem molecular de fármacos (JIANG *et al*., 2008).

Os testes de validação do desempenho do Dovis 1.0 com o banco de dados ZINC, utilizando gradualmente até 128 núcleos de processamento (CPUs), demonstrou uma progressão ou escalonamento quase linear no ganho de desempenho obtido. O cruzamento do banco de dados ZINC com a cadeia A da proteína Ricina, utilizando 256 CPUs, conseguiu realizar o *docking* de aproximadamente 700 ligantes por CPU, ao dia (ZHANG *et al*., 2008).

2.3.1.4 DockThor

O projeto DockThor (DockThor, 2016) é uma iniciativa do Laboratório Nacional de Computação Cientifica (LNCC, 2016) para a realização de *docking* molecular através de um servidor Web.

Possui uma interface gráfica amigável ao usuário, que guia os procedimentos em cada etapa durante a preparação do cálculo de *docking* molecular. Para realizar um cálculo de *docking* molecular nesta plataforma, o usuário submete os arquivos com as estruturas da macromolécula (alvo biológico), ligante (candidato a fármaco) e cofatores (caso necessário) para, posteriormente, definir a região do sitio ativo da macromolécula, que será o local onde ocorrera a busca pela conformação mais estável para o complexo

(receptor + ligante). O resultado do cálculo de *docking* molecular é então enviado ao usuário via e-mail.

A infraestrutura utilizada para executar os cálculos *docking* molecular é a do Sistema Nacional de Alto Desempenho (SINAPAD, 2016) que conta com mais de 1,4 petaFLOPS de pode computacional distribuídos ao longo de nove *clusters* (denominados CENAPAD) em todo o Brasil, incluindo o super computador (cluster) Santos Dumont.

2.3.2 INFRAESTRUTURAS BASEADAS EM GRIDS COMPUTACIONAIS

2.3.2.1 World Community Grid

*O World Community Grid (*WCG*,* 2016) é uma iniciativa desenvolvida pela IBM que utiliza o ambiente *de grid* computacional da plataforma BOINC (BOINC, 2016) para apoiar projetos científicos voltados ao beneficio humanitário.

Alguns destes projetos são voltados para a área da saúde e executam os programas AutoDock (GOODSELL; MORRIS; OLSON, 1996) e AutoDock Vina (TROTT; OLSON, 2010) na máquina do voluntário, na tentativa de encontrar um potencial fármaco para algumas das doenças estudadas. Entre os projetos ativos que utilizam esta abordagem estão o Outsmart Ebola Together (Outsmart Ebola Together, 2016), FightAIDS@Home (FightAIDS@Home, 2016) e OpenZika (OpenZika, 2016). Outros projetos de pesquisa, já finalizados, que também utilizaram os programas AutoDock ou AutoDock Vina no *World Community Grid*, são: i) *Say No to Schistosoma* (Say No to Schistosoma, 2016); ii) GO Fight Against Malaria (GO Fight Against Malaria, 2016); iii) Drug Search for Leishmaniasis (Drug Search for Leishmaniasis, 2016); iv) Discovering Dengue Drugs – Together (Discovering Dengue Drugs, 2016); v) Help Fight Childhood Cancer (Help Fight Childhood Cancer, 2016); vi) Influenza Antiviral Drug Search (Influenza Antiviral Drug Search, 2016).

Este projeto possui mais de 720 mil voluntários, que disponibilizam mais de 3.200.000 dispositivos (WCG, 2016) para atender à demanda computacional. O tempo ganho com o poder de processamento doado pelos usuários deste projeto em seus dispositivos, desde o início do projeto em 2004 até os dias de hoje, equivale a incríveis mais de 1.300.000 (um milhão e trezentos mil) anos de processamento, caso este fosse feito em um único computador comum (WCG, 2016). Diariamente, o conjunto de dispositivos voluntários deste projeto processa dados que levaria mais de 480 anos em um único computador (WCG, 2016).

2.3.2.2 Rosetta@Home

O projeto Rosetta@Home (Rosetta@Home, 2016) utiliza o ambiente de *grid* computacional da plataforma BOINC (BOINC, 2016) para estudar como ocorre o enovelamento ("folding") de proteínas.

A metodologia deste programa para modelar o enovelamento consiste em duas fases. A primeira, começa com a proteína completamente esticada e, a partir deste estado inicial, gerar perturbações estruturais em uma parte da cadeia da proteína. Quando uma perturbação estrutural reduz a energia, ou seja, torna a estrutura mais estável, a perturbação é aceita, caso contrario a alteração é rejeitada (Rosetta@Home, 2016). Na segunda fase, denominada fase de relaxamento, são realizadas perturbações menores com o intuito de realocar os amino ácidos para uma posição mais estável, levando em consideração todos os átomos da proteína (Rosetta@Home, 2016).

O objetivo principal de se realizar o enovelamento de proteínas é tentar entender e predizer a conformação em que dada proteína aparece na natureza. Como a conformação de uma proteína pode afetar sua função e a forma como ela interage com outras moléculas, descobrir a conformação mais estável de uma proteína pode ajudar no combate à doenças como Malaria, Anthrax e HIV (Rosetta@Home, 2016), dentre outras.

Acontece que proteínas são estruturas grandes, muitas vezes contendo centenas ou até milhares de aminoácidos, o que torna a tentativa de enovelamento de proteínas um caso de alta demanda computacional. Este projeto possui mais de 1.200.000 voluntários (BOINC Stats, 2016), doando mais de 226.000 teraFLOPS (BOINC Stats, 2016) de processamento ao longo de mais de 2.150.000 computadores (BOINC Stats, 2016).

2.3.2.3 Folding@Home

Desenvolvido pela Universidade de Stanford, o programa Folding@Home utiliza o poder de processamento ocioso de voluntários para formar um ambiente de *grid* computacional com o intuito de descobrir os mecanismos por traz do enovelamento de proteínas (Folding@Home, 2016). O objetivo do projeto é descobrir quais fatores podem causar um enovelamento errôneo de uma proteína, causando doenças como Alzheimer, Huntington e alguns tipos de câncer (Folding@Home, 2016).

Ao contrário de outras iniciativas que utilizam *grid* computacionais, como, por exemplo, *World Community Grid* (WCG, 2016) e Rosetta@Home (Rosetta@Home, 2016) que utilizam a plataforma BOINC (BOINC, 2016), o projeto Folding@Home tem sua própria plataforma *grid* computacional dedicada ao projeto. O projeto Folding@Home possui cerca de 100 teraFLOPS de poder de processamento, vindo de mais de 110.000 computadores, e oferece suporte a vários tipos de processamento utilizando o paralelismo interno do computador em múltiplos núcleos de processamento de um computador (CPUs) e unidades de processamento gráfico (GPUs) (Folding@Home, 2016).

2.3.2.4 Foldit

O projeto Foldit (COOPER et al., 2010) consiste em um espécie de jogo online com múltiplos jogadores, que utiliza o esforço colaborativo de milhares de usuários, no intuito de elucidar aspectos importantes no processo de enovelamento de estruturas

tridimensionais de proteínas.

Cada proteína é apresentada como um quebra-cabeça (*puzzle*), onde o jogador pode escolher se deseja enovelar a proteína de maneira individual (*soloist*), ou em conjunto com um grupo (*groups*) de usuários. As regras do jogo são: i) Empacotar as proteínas no espaço mais compacto possível; ii) Evitar que os resíduos hidrofóbicos fiquem na superfície da molécula; iii) Evitar que os átomos da molécula se choquem. Assim, quanto mais compacta estiver a proteína, com o menor número possível de resíduos hidrofóbicos em sua superfície e evitando que átomos fiquem muito próximos de outros átomos, melhor será a pontuação do jogador.

O que torna esta iniciativa mais eficiente do que um método computacional tradicional para a previsão do enovelamento de proteínas é que, como cada jogador utiliza uma estratégia diferente para completar o jogo, pode-se utilizar diversas metodologias para solução dos quebra-cabeças, cada uma delas aplicada por um jogador diferente, comparado com a utilização de apenas uma metodologia aplicada em um algoritmo específico (COOPER et al., 2010). Em um teste realizado as cegas, utilizando-se 10 quebra-cabeças (proteínas) diferentes, e não disponíveis em banco de dados público, os jogadores conseguiram encontrar cinco soluções mais próximas da estrutura nativa de cada proteína, comparado-as com as soluções encontradas pelo método de predição de estrutura conhecido como Rosetta (ROHL et al., 2004). Além disso, estes jogadores encontraram a mesma solução que o programa Rosetta em três destes quebra-cabeças.

Um quebra-cabeça especial, disponível apenas por 21 dias, com uma enzima do vírus M-PMV, responsável pela AIDS em macacos, teve seu enovelamento solucionado pelos jogadores do Foldit em apenas 10 dias, encontrando-se a mesma solução observada experimentalmente (KHATIB et al., 2011), sendo este um problema que não havia sido solucionado em 15 anos (FOLDIT Portal, 2016).

2.3.3 INFRAESTRUTURAS BASEADAS EM UNIDADES DE PROCESSAMENTO GRÁFICO

2.3.3.1 MolDock

O MolDock (THOMSEN; CHRISTENSEN, 2006) é um programa de *docking* molecular baseado em campos de força, que permite ao usuário levar em conta vários graus de liberdade do ligante. Embora o programa utilize toda a superfície da macromolécula alvo, seu algoritmo de busca é um algoritmo genético que limita a zona de busca à apenas as cavidades (possíveis sítios ativos). Martin Simonsen e sua equipe (SIMONSEN et al., 2013) adaptaram o programa MolDock para que as rotinas computacionais que mais demandam esforço computacional, neste caso, os algoritmos de busca e pontuação, fossem realizadas na unidade de processamento gráfico (GPU) dos computadores.

O ganho de desempenho (tempo) obtido ao realizar parte do procedimento de *docking* molecular na placa de vídeo Geforce 8800 GT (112 núcleos) variou entre 7,7 à 37,5 vezes mais rápido, quando comparado com a versão original do programa, que não utiliza este paralelismo. É importante salientar que foi obtido praticamente o mesmo nível de precisão (SIMONSEN et al., 2013). A média de ganho de desempenho (tempo) foi de 27,4 em 10 execuções e 33,1 em 20 execuções.

2.3.3.2 Bristol University Docking Engine (BUDE)

A versão atual do programa de *docking* molecular BUDE (McIntosh-Smith et al., 2014), o qual é baseado em campos de força, combina técnicas de dinâmica molecular e algoritmos genéticos para buscar a melhor conformação entre o receptor (alvo biológico) e o ligante (candidato a fármaco), de forma rápida, ao realizar o *docking* molecular de forma

distribuída utilizando as unidades de processamento gráfico (GPU). Como um dos objetivos do trabalho em questão era possibilitar que o programa funcionasse na maior variedade possível de GPUs, utilizou-se a linguagem OpenCL (OpenCL, 2015), que oferece suporte aos produtos das principais empresas do mercado como por exemplo: Intel, AMD, ATI e Nvidia (McIntosh-Smith et al., 2014) .

Duas métricas foram utilizadas para mensurar o ganho de desempenho desta abordagem distribuida. A primeira métrica definida foi o número de interações entre pares atómicos por segundo. Nesta métrica, o desempenho obtido pelo programa em sua versão não paralelizada em GPU foi de 11,7 bilhões de cálculos de interações entre pares de átomos por segundo, enquanto a versão paralelizada em GPU conseguiu 44 bilhões de cálculos de interações por segundo, representando um ganho de desempenho em aproximadamente 3,8 vezes mais interações calculadas por segundo, em relação ao programa BUDE não paralelizado (McIntosh-Smith et al., 2014).

A segunda métrica utilizada focou em quantos cálculos de ponto flutuantes por segundo (FLOPs) são realizados por cada abordagem. Esta métrica é bastante utilizada para quantificar o poder de processamento em sistemas computacionais de alto desempenho, sendo a principal métrica usada na compilação da lista dos 500 supercomputadores mais poderosos do mundo (Top500 Supercomputers, 2015). O desempenho obtido nesta métrica, pela versão não paralelizada do programa, foi de 0,14 teraFLOPs (trilhões de cálculos de ponto flutuante), enquanto a versão paralelizada em GPU conseguiu atingir 1,43 teraFLOPs, acarretando um ganho de desempenho de 10,21 vezes em relação ao programa BUDE original não paralelizado (McIntosh-Smith et al., 2014) .

2.3.3.3 BindSurf

O programa de *docking* molecular BINDSURF (SANCHEZ-LINARES, 2012) utiliza

um novo paradigma na realização de cálculos de *virtual screening* ou *docking molecular*. Essencialmente, *virtual screening* ocorre quando temos uma macromolécula alvo e procura-se em um banco de dados de ligantes (candidatos a fármacos) quais deles teriam a maior afinidade pelo alvo biológico. Geralmente, o sitio ativo escolhido para posicionar os ligantes é o mesmo onde se encontra o substrato natural tipicamente co-cristalizado na estrutura cristalográfica.

Porém, por acreditar que diferentes ligantes poderiam ter afinidade por diferentes sítios ativos de uma mesma macromolécula, a metodologia do programa BINDSURF tenta posicionar cada ligante (fármaco em potencial) ao longo de toda superfície da macromolécula alvo, estando a procura de outros sítios ativos (SANCHEZ-LINARES, 2012). As regiões da macromolécula identificadas como possíveis sítios ativos são utilizadas para posicionar o ligante em busca da conformação mais estável. Estas buscas pela melhor conformação em vários sítios ativo são realizadas simultaneamente através do paralelismo *multithreading,* utilizando-se as unidades de processamento gráfico disponíveis nas placas de vídeo modernas (SANCHEZ-LINARES, 2012).

2.3.3.4 Piper

Bharat Sukhwan e Martin C. Herbordt (SUKHWANI, HERBORDT, 2009) modificaram o programa de *docking* macromolecular PIPER (KOZAKOV et al., 2006), especializado em realizar *docking* entre proteínas, para adicionar suporte a processamento em paralelo. Foram desenvolvidas duas versões com suporte a processamento em paralelo do programa PIPER: uma versão que utiliza o paralelismo em nível de CPU, neste caso um processador quad-core (4 núcleos de processamento), e outra versão que utiliza paralelismo em nível de GPU.

O desempenho da versão que utiliza paralelismo em nível de GPU conseguiu ser 17,7 vezes mais rápido quando comparado com a versão original não paralelizada do

programa, e 6,1 vezes mais rápido quando comparado com a versão paralelizada em nível de CPU em um computador com processador quad-core.

2.3.4 COMPARAÇÃO COM O PROGRAMA GRIDOMOL

Como o programa GriDoMol, desenvolvido nesta tese de doutorado, possui o mesmo objetivo que algumas das ferramentas citadas nesta revisão de literatura, ou seja, realizar cálculos de docking molecular em larga escala utilizando um elevado número de moléculas de forma acelerada utilizando vários computadores interligados através de arquiteturas distribuídas, optou-se por comparar as funcionalidades destas ferramentas em relação ao programa GriDoMol (ver Tabela 3). Como pode ser observado na Tabela 3, além do programa GriDoMol ser a única destas ferramentas a possuir uma metodologia para aplicações em vacinologia reversa, o programa GriDoMol contempla uma ampla gama de funcionalidades.

Tabela 3 - Comparação entre algumas das principais características do programa GriDoMol e outras soluções para a realização de docking molecular de forma distribuída.

Funcionalidades	GriDoMol	Mola	Haddock	Dovis	DockThor	World Community Grid
Permite criar uma infraestrutura utilizando somente os computadores locais	Sim	Sim	Não	Sim	Não	Não
Permite utilizar o poder computacional de *Smartphones* e *Tablets*	Sim	Não	Não	Não	Não	Sim
Permite submeter cálculos de docking molecular diretamente do navegador	Não	Não	Sim	Não	Sim	Não
Metodologia para realizar cálculos de docking molecular voltados à vacinologia reversa	Sim	Não	Não	Não	Não	Não

3 OBJETIVOS

3.1 Geral

- Disponibilizar uma plataforma unificada para realização de cálculos de docking molecular em larga escala.

3.2 Específicos

- Projetar e desenvolver uma nova versão do programa GriDoMol (2.0), na linguagem de programação C++, para *realização dos testes de performance*.
- Projetar e desenvolver uma nova versão do programa GriDoMol (3.0), na linguagem de programação Java, para possibilitar a integração entre as versões do programa GriDoMol para *Desktop* e, futuramente, versões *Web* e *Mobile*.
- Desenvolver uma interface gráfica de usuário (GUI) amigável e intuitiva, com o intuito de aumentar a interatividade com o usuário.
- Validar o ganho de desempenho computacional obtido ao empregar a plataforma de grid computacional através da análise de desempenho (“benchmark”).
- Utilizar um conjunto de testes para verificar o nível de precisão das soluções de docking molecular, em relação aos seus respectivos valores experimentais de energia livre de ligação (“binding”), e RMSD em relação à estrutura cristalográfica, ao se utilizar ajustes de cálculo mais robustos para os procedimentos de docking molecular.
- Desenvolver uma metodologia para utilizar o programa GriDoMol em estudos de vacinologia reversa, permitindo a realização de docking molecular entre candidatos a epítopos e alelos de MHC Classe I e II humanos.

4 METODOLOGIA

4.1 GriDoMol

4.1.1 GRIDOMOL 2.0 (EM C++)

Uma nova versão (2.0) do programa GriDoMol (GRid para DOcking MOLecular), que começou a ser criado em um estudo anterior (FERREIRA, 2013), foi criada com o intuito de adicionar funcionalidades importantes que não faziam parte da versão anterior. As principais funcionalidades adicionadas foram: i) A capacidade de selecionar individualmente qual programa de *docking* molecular realizará cada um dos cálculos de *docking* molecular, em um mesmo *job* (conjunto de cálculos); ii) Contemplar as principais opções avançadas utilizadas pelos algoritmos de busca nos programas de *docking* molecular AutoDock e AutoDock Vina; iii) Capacidade de alteração da ordem com que os cálculos de *docking* são executados. A principal motivação para realizar a implementação desta versão atualização em C++ do programa GriDoMol foi a possibilidade de obter uma versão intermediaria, utilizando pouco esforço de desenvolvimento, uma vez que já existia uma versão em C++ operacional do programa GriDoMol, que pudesse ser utilizada enquanto a versão definitivo do programa GriDoMol (em Java), principal objetivo desta tese, estava sendo projetada e desenvolvida.

4.1.2 GRIDOMOL 3.0 (EM JAVA)

A versão mais moderna e atual do programa GriDoMol foi desenvolvida utilizando-se a linguagem de programação Java (Java, 2016), amplamente utilizada no mundo acadêmico e comercial, por causa do seu alto grau de portabilidade. A linguagem de programação Java é multiplataforma e, uma vez codificada a aplicação, ela pode ser executada facilmente nos diversos sistemas operacionais que possuem uma instância funcional da *Java Virtual Machine* (JVM). Desta forma, foi possível, por exemplo,

desenvolver versões do programa GriDoMol para plataformas Linux e Windows, com poucas modificações necessárias no código fonte.

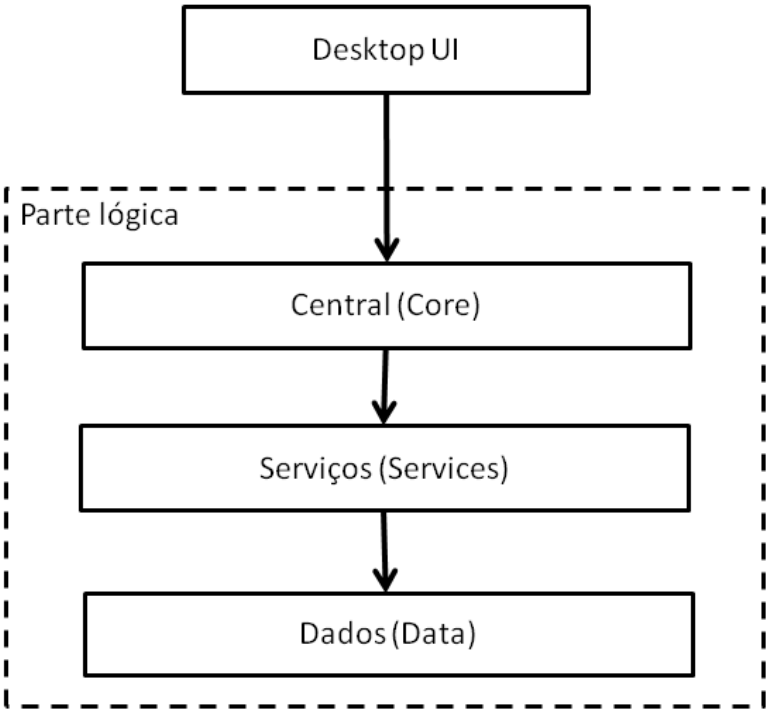


Figura 5 – Diagrama do desenvolvimento em camadas aplicado ao programa GriDoMol.

Com o intuito de facilitar a integração entre a parte lógica do programa GriDoMol (que são suas funcionalidades) e a interface gráfica do usuário desktop (pensando, também, a posteriori, nas versões Web e Mobile), o programa GriDoMol foi desenvolvido em camadas (ver Figura 5). A camada mais interna, *Data*, contem as definições das estruturas de dados utilizadas pelo programa GriDoMol (*Job, Task, Worker, Ligand, Receptor*, etc.). Nesta camada, por exemplo, define-se que um objeto do tipo *Receptor* (Macromolécula alvo) possui atributos como coordenadas tridimensionais do centro de seu sitio ativo, bem como também o tamanho do mesmo, dentre outros atributos. A camada *Services* é responsável por realizar as operações envolvidas na utilização das estruturas de dados definidas na camada *Data*, tal como, por exemplo, adicionar um

Receptor (alvo biológico) e *Ligand* (candidato a fármaco) a uma *Task* (Tarefa) e, posteriormente, adicionar esta *Task* a um *Job* (Conjunto de tarefas). A camada *Core* é responsável pela intermediação dos comandos vindos de uma interface gráfica (camada *UI*) e a camada *Services*, desta forma, podendo prover um acesso unificado e simplificado para o sistema GriDoMol.

A interface de desenvolvimento (IDE) utilizada para implementação foi o Eclipse Luna (Eclipse, 2016). A escolha desta IDE se deve ao fato de que a mesma possui versões multiplataformas, facilitando o desenvolvimento de códigos fontes nestes sistemas operacionais, sem a necessidade de importar ou modificar os arquivos do projeto, em si, para se adequarem a outra IDE. Além disto, esta IDE "*open source*" possui diversos módulos para auxiliar o desenvolvimento de aplicações Java, como por exemplo, o modulo WindowBuilder (WindowBuilder, 2016), desenvolvido pela empresa Google com o intuito de facilitar a criação de interfaces gráficas. A versão do interpretador Java (JRE) utilizada durante o desenvolvimento do programa GriDoMol foi a versão 7.0_45.

Embora boa parte do código utilize as funções próprias da linguagem de programação Java, houve algumas partes do código fonte que foram implementados como comandos específicos para o sistema operacional. A linguagem de programação Java possui um comando chamado "***System.getProperty()***" onde é possível obter informações do sistema que estiver executando um programa desenvolvido em Java, como nome e versão do sistema operacional, por exemplo. Sendo assim, utilizou-se os recursos como "if System.getProperty("os.name").contains(**nome_do_sistema**)", onde "**nome_do_sistema**" é substituído por Windows ou Linux, para indicar qual parte do código deve ser executada em determinada plataforma alvo e, desta forma, poder disponibilizar o programa GriDoMol para estas duas plataformas utilizando um único código fonte e executável (arquivo .JAR), com vantagens operacionais.

4.2 OurGrid

Para permitir o acesso aos componentes do OurGrid através de linha de comando no sistema operacional Windows, houve a necessidade de editar o arquivo de lote (.bat) de cada componente do OurGrid. Houve uma modificação na parte “(...) org.ourgrid.broker.ui.**async**.Main” para “(...) org.ourgrid.broker.ui.**sync**.Main **%1 %2**”, no componente Broker, e de “(...) org.ourgrid.peer.ui.**async**.Main **start** (...)” para “(...) org.ourgrid.peer.ui.**sync**.Main **%1 %2** (...)”, no componente Peer. O comando “**start /b javaw**” foi substituído por apenas “**java**” no início de cada um destes arquivo de lote. Estas alterações permitiram que os componentes do OurGrid pudessem iniciar em modo de linha de comando ("**sync**"), um requisito essencial para possibilitar a execução de comandos vindo do programa GriDoMol, além de permitir especificar quais comandos serão executados, através da passagem de argumentos ("**%1**" e "**%2**").

Por se tratar de um ambiente de grid computacional local, formado com computadores do mesmo laboratório e sem acesso a internet, e com o intuito de facilitar a integração entre os computadores pertencentes ao ambiente de grid computacional, adicionou-se dois parâmetros ao arquivo de configuração do componente Peer do OurGrid. O primeiro parâmetro utilizado foi o “peer.voluntary=yes", cujo objetivo é permitir que o componente Peer aceite qualquer computador apto para realizar uma task (neste caso, cálculos de docking molecular) que peça para integrar o ambiente de grid computacional. Já o parâmetro “peer.register.ondemand=yes” permite que o componente Peer aceite pedidos de submissão de jobs de qualquer computador com o componente Broker. É importante ressaltar que, por razões de segurança, em ambientes de grid computacionais multi institucionais, onde os componentes estão distribuídos geograficamente e estão conectados entre si através da internet, estes parâmetros adicionais do componente Peer não devem ser utilizados. Ao invés disto, recomenda-se a adição manual de cada componente, utilizando o componente Peer para assinar os

certificados digitais de cada um destes componentes do OurGrid que estão autorizados a acessar o ambiente de grid computacional para que, somente eles, possam acessar estes recursos computacionais.

Em cada computador apto a realizar cálculos de docking molecular, foram configurados o mesmo número de componentes Workers quanto o número de núcleos de processamento que o computador possui. Esta abordagem foi feita por se tratar da recomendação oficial feita pelo instalador deste componente durante o processo de instalação. Além disto, esta abordagem permite que programas que não possuem paralelismo interno em nível de CPUs, como é o caso do programa AutoDock, possa enviar várias instâncias de execução para um mesmo computador de forma a acelerar a execução do conjunto de cálculos de docking molecular como um todo. Para evitar o uso de múltiplos componentes Workers quando submete-se cálculos de docking molecular com o programa AutoDock Vina, uma vez que este programa já utiliza paralelismo interno em nível de CPUs, foi criado um requisito (vina=true) no primeiro componente Worker de cada computador de forma a permitir que apenas um Worker por computador seja usado neste caso onde o programa já utiliza, nativamente, todos os núcleos de processamento do computador.

4.3 OpenFire

Para realizar a comunicação entre os componentes do ambiente de *grid* computacional denominado OurGrid, é necessário utilizar o protocolo XMPP (eXtensive Messaging and Presence Protocol). Este protocolo detecta a presença dos componentes do OurGrid nos computadores e possibilita a troca de mensagens entre eles. O servidor XMPP utilizado neste trabalho foi o OpenFire 3.9.3 (OpenFire, 2016), por se tratar de uma solução "*open source*" fácil de usar, seguro e com uma ótima escalabilidade, suportando até 50.000 acesso simultâneos a partir de sua versão 3.2 (OpenFire, 2016).

4.4 Docking Molecular

Para realizar os cálculos de *docking* molecular, foram utilizados os programas AutoDock 4.2.6 (GOODSELL; MORRIS; OLSON, 1996; MORRIS et al., 2009) e o AutoDock Vina 1.1.2 (TROTT; OLSON, 2010), ambos desenvolvidos pelo mesmo grupo (The Scripps Research, 2016).

4.5 Testes de Desempenho

Uma vez que a versão 3.0 (em JAVA) do programa GriDoMol ainda não possuía a capacidade de criar, enviar e gerenciar os cálculos de *docking* molecular, particularmente no momento em que os testes de desempenho foram iniciados nesta tese de doutorado, optou-se por iniciar a execução dos testes utilizando a versão 2.0 (em C++) do programa GriDoMol, a qual já estava apta para realizar a execução dos *jobs* (conjunto de cálculos individuais de *docking*). Posteriormente, com a versão 3.0 do programa GriDoMol finalizada, realizou-se a execução de jobs nesta nova versão utilizando o mesmo conjunto de moléculas e os mesmos computadores do grid computacional, reavaliando-se o desempenho desta nova versão do programa, escrito em Java.

O conjunto de moléculas escolhido para os testes foi o mesmo utilizado pelo grupo de bioquímica da universidade Zhejiang (China), em um estudo recente sobre a otimização de funções de pontuação de programas de *docking* molecular (WANG et al., 2013). Este conjunto de moléculas é um subconjunto selecionado a partir do banco de moléculas PDBBind (LIU et al., 2015; WANG et al., 2004), no qual consta os valores experimentais das constantes de inibição (K) para alguns dos complexos disponíveis no banco de moléculas PDB (ROSE et al., 2014; BERMAN et al., 2000). Ao utilizar um conjunto de moléculas em que os valores de energia da constante de inibição de cada

complexo já foi determinado experimentalmente (in vitro), pode-se comparar estes valores de energia reais diretamente com as predições obtidas pelos cálculos de docking molecular (in silico), uma vez que as soluções de docking molecular que são obtidas através dos programas AutoDock e AutoDock Vina também estão em valores de energia. Em outras palavras, ao utilizar este conjunto de testes, pode-se avaliar a capacidade dos programas de docking em prever o comportamento de um grande número de moléculas ao comparar estes resultados preditos com os resultados reais obtidos em testes experimentais.

Além disto, outro motivo para a escolha deste conjunto de moléculas foi sua diversidade. Em uma situação real de *virtual screening*, geralmente utiliza-se um conjunto de moléculas bastante diferente entre si e, ao utilizar um conjunto heterogêneo como o proposto acima, esperamos conseguir chegar mais próximo de um caso de uso real para a plataforma GriDoMol.

Um conjunto de 213 (duzentos e treze) receptores, e seus respectivos ligantes co-cristalizados, foi selecionado para a realização de *re-docking* (ver Tabela 5). O *re-docking* acontece quando se retira o ligante que está co-cristalizado junto com o seu respectivo alvo biológico, e o programa de *docking* molecular tenta reinseri-lo na macromolécula. Esta abordagem é muito utilizada para fins de avaliação da capacidade preditiva dos programas de *docking* molecular.

Tabela 5 – Tabela com o conjunto de 213 moléculas utilizadas durante os testes de desempenho realizados com o GriDoMol.

PDB	# de Amino-ácidos	MM ligante (g/mol)	-LogK	PDB	# de Amino-ácidos	MM ligante (g/mol)	-LogK	PDB	# de Amino-ácidos	MM ligante (g/mol)	-LogK	PDB	# de Amino-ácidos	MM ligante (g/mol)	-LogK
1A30	99	-	4.3	**1QI0**	305	342,3	2.35	**2JDY**	115	194,18	4.37	**3CKP**	412	545,67	6.35
1A94	99	833,05	7.85	**1R5Y**	386	176,18	6.46	**2OBF**	289	352,84	8.85	**3COW**	301	460,46	6.9
1ADL	131	304,47	5.36	**1RE8**	350	474,51	9.52	**2ON6**	289	266,25	6.76	**3D4Z**	1045	201,2	4.89
1AMW	214	427,2	4.54	**1SL3**	287	552,32	11.85	**2OSF**	259	142,18	3.83	**3DXG**	124	324,18	2.4
1B39	299	507,18	6.92	**1SLG**	135	-	3.9	**2P15**	258	442,51	10.3	**3E8R**	271	467,52	9.74
1B7H	517	-	8.02	**1SV3**	121	152,15	4.35	**2P4Y**	277	574,93	9	**3E93**	371	496,6	8.85
1B8O	284	266,25	10.64	**1SWR**	127	244,31	6.92	**2PGZ**	230	303,35	5.74	**3EHY**	158	259,28	5.85
1B9J	517	-	5.96	**1TRD**	250	171,05	5.4	**2POW**	260	319,28	7.2	**3EJR**	1045	347,45	8.57
1BCU	259	209,25	3.28	**1TSY**	316	308,18	4.96	**2PQ9**	427	458,15	8.11	**3F17**	158	304,32	8.63
1BGQ	225	364,78	8.57	**1TXR**	299	308,37	7.74	**2PU2**	358	355,3	4.43	**3F3T**	286	399,49	4.59
1BXO	323	638,69	10	**1U1B**	124	809,36	7.8	**2QBP**	299	609,51	8.4	**3F80**	322	176,17	4.22
1BXQ	323	626,68	7.38	**1UTO**	243	122,19	2.27	**2QE4**	248	326,39	7.96	**3FCQ**	316	194,18	2.77
1C1V	259	335,39	7.64	**1V48**	289	337,22	7.8	**2QFT**	427	170,08	5.26	**3FK1**	427	169,07	2.62
1C88	298	270,26	5.29	**1VSO**	257	322,25	4.72	**2QM9**	155	441,54	7.77	**3G2Z**	263	179,18	2.36
1D7J	107	88,11	3.3	**1W3K**	303	329,31	4.3	**2QMJ**	870	645,61	4.21	**3G32**	263	243,22	3.71
1DF8	127	244,31	9.7	**1W3L**	303	-	6.28	**2QWB**	388	309,27	2.74	**3G35**	263	283,26	4.68
1E66	543	298,81	9.89	**1X1Z**	252	340,18	11.06	**2QWD**	388	290,27	4.85	**3G5D**	286	488,01	7.96
1FKB	107	914,17	9.7	**1XGJ**	358	343,33	6	**2QWE**	388	334,33	7.48	**3GBA**	257	304,3	9.07
1FKI	107	437,57	7	**1XH6**	350	541,64	7.32	**2R23**	224	278,26	3.72	**3GE7**	386	355,44	8.7
1FLR	219	332,31	10	**1Y1Z**	292	115,13	3.08	**2RFH**	307	209,2	6.82	**3GNW**	581	613,7	9.1
1FTM	263	186,17	7.61	**1Y6Q**	242	293,39	11.7	**2RKM**	517	294,4	3.9	**3GV9**	358	185,2	2.12
1G2K	99	631,74	7.96	**1YC1**	264	368,34	6.17	**2STD**	172	334,67	9.85	**3GY4**	223	136,17	5.1
1G74	131	282,46	6.52	**1ZEA**	216	-	5.22	**2USN**	165	414,5	6.51	**3HEC**	348	493,6	4.47
1GPK	537	242,32	5.37	**2ARM**	121	289,37	7.68	**2UWO**	254	461,98	8.7	**3HZK**	226	516,45	6.7
1H23	543	494,71	8.35	**2AVQ**	99	770,98	4.39	**2V00**	330	215,25	3.66	**3HZM**	226	238,19	5.52
1HFS	160	705,86	8.7	**2B1V**	257	260,33	5.74	**2V2H**	242	156,03	3.77	**3IE3**	209	297,33	6.68
1HNN	282	212,27	6.24	**2B7D**	254	532,51	8.7	**2VL4**	846	16,17	6.01	**3IMC**	301	147,17	2.96
1IF7	259	371,45	10.52	**2BRB**	297	331,37	4.86	**2VO5**	846	279,33	4.89	**3IUB**	301	345,37	4.54
1J37	607	217,29	8.96	**2BZ6**	254	417,43	7.09	**2VOT**	846	306,34	7.14	**3IWW**	709	517,32	7.96
1JQ8	121	-	6	**2CET**	468	305,35	8.02	**2W66**	716	234,25	4.05	**3JVS**	271	406,43	6.54
1JYS	242	135,13	3.52	**2CGF**	225	338,78	6.41	**2WEC**	323	539,54	3.96	**3K8Q**	289	238,24	11.28
1KEL	218	346,27	7.28	**2CTC**	307	166,17	3.89	**2WN9**	228	294,35	8.52	**3KEJ**	167	432,41	7.12
1LBK	208	307,32	3.18	**2D1O**	171	462,63	7.7	**2X00**	228	509,72	11.33	**3KEK**	167	451,49	8.36
1LOL	229	365,21	6.39	**2D3U**	570	398,46	6.92	**2X91**	598	405,49	6.74	**3KME**	270	374,45	6.14
1LOQ	228	324,18	3.7	**2EXM**	298	203,24	4.11	**2X96**	598	617,63	4.03	**3KV2**	322	176,17	6.29
1LVU	289	300,21	5.32	**2F34**	258	353,35	6.89	**2XAB**	249	297,35	9.27	**3L4U**	875	345,39	7.52
1MQ6	233	568,86	11.15	**2FLR**	254	358,39	6.1	**2XDL**	249	223,27	3.1	**3L4W**	875	207,22	6
1N2V	386	208,22	4.08	**2G5U**	127	323,99	8.49	**2XEG**	709	746,72	10.7	**3L7A**	843	379,34	4.33
1N5R	543	414,59	5.66	**2G71**	289	340,34	7.46	**2XJ7**	716	230,26	6.66	**3L7D**	843	363,34	2.18
1NC1	242	296,35	6.12	**2G94**	389	658,85	9.52	**2XMY**	298	435,52	9.96	**3LKA**	158	187,22	2.82
1NJA	316	307,2	6.31	**2G9Q**	842	133,15	6.4	**2ZB0**	360	394,43	6.62	**3MFV**	322	184,2	2.52
1NJE	316	307,2	3.8	**2GSS**	209	303,14	4.94	**2ZCQ**	293	386,36	8.82	**3NEX**	116	376,45	6
1NVQ	289	482,53	8.25	**2HA3**	543	104,17	3.03	**2ZCR**	293	540,74	6.87	**3PCE**	238	125,15	2
1NWL	298	469,58	2.39	**2HB3**	99	560,7	11.35	**3ACW**	293	276,38	4.76	**3PCJ**	238	155,11	7.22
1O0M	124	324,18	5.15	**2I4J**	286	452,59	6.16	**3ADV**	287	176,22	3.03	**3PCN**	238	168,15	3.66
1O3F	223	327,38	7.96	**2I4X**	99	728,79	11.72	**3B3C**	291	167,14	3.98	**3STD**	165	364,44	11.11
1O5B	23	303,14	5.77	**2J47**	716	361,37	5.41	**3B7I**	291	167,14	5.82	**456C**	168	425,88	9.77
1OS0	316	494,52	6.03	**2J77**	468	163,17	4.89	**3B92**	259	284,39	8	**4ER2**	330	-	9.3
1P1Q	263	186,17	4.89	**2J78**	468	192,17	6.42	**3BFU**	263	189,19	6.27	**4TIM**	250	186,06	2.16
1PB8	292	105,09	5.15	**2J94**	254	488,97	6.27	**3BGS**	289	267,3	9.42	**4TMN**	316	519,53	10.17
1PBQ	292	258,06	6.27	**2JBJ**	707	226,12	9.7	**3BRA**	409	137,18	2.7	**5ER2**	330	948,2	6.57
1PS3	1045	232,19	2.28	**2JDN**	115	194,18	5.56	**3CFT**	118	223,25	4.19	**6CPA**	307	478,43	11.52
1Q84	580	661,86	11.05	**2JDU**	115	178,18	6.72	**3CJ2**	576	311,22	4.85	**6STD**	164	413,16	8.64
1Q8T	350	247,34	4.76												

Com o intuito de avaliar a diversidade química do conjunto de ligantes presentes em cada um dos 213 complexos (receptor + ligante) do conjunto de testes selecionado, utilizou-se o programa DataWarrior (SANDER et al., 2015) para criar gráficos mostrando a distribuição de algumas das propriedades destes ligantes ao longo de todo o conjunto (ver Figura 10 e 11).

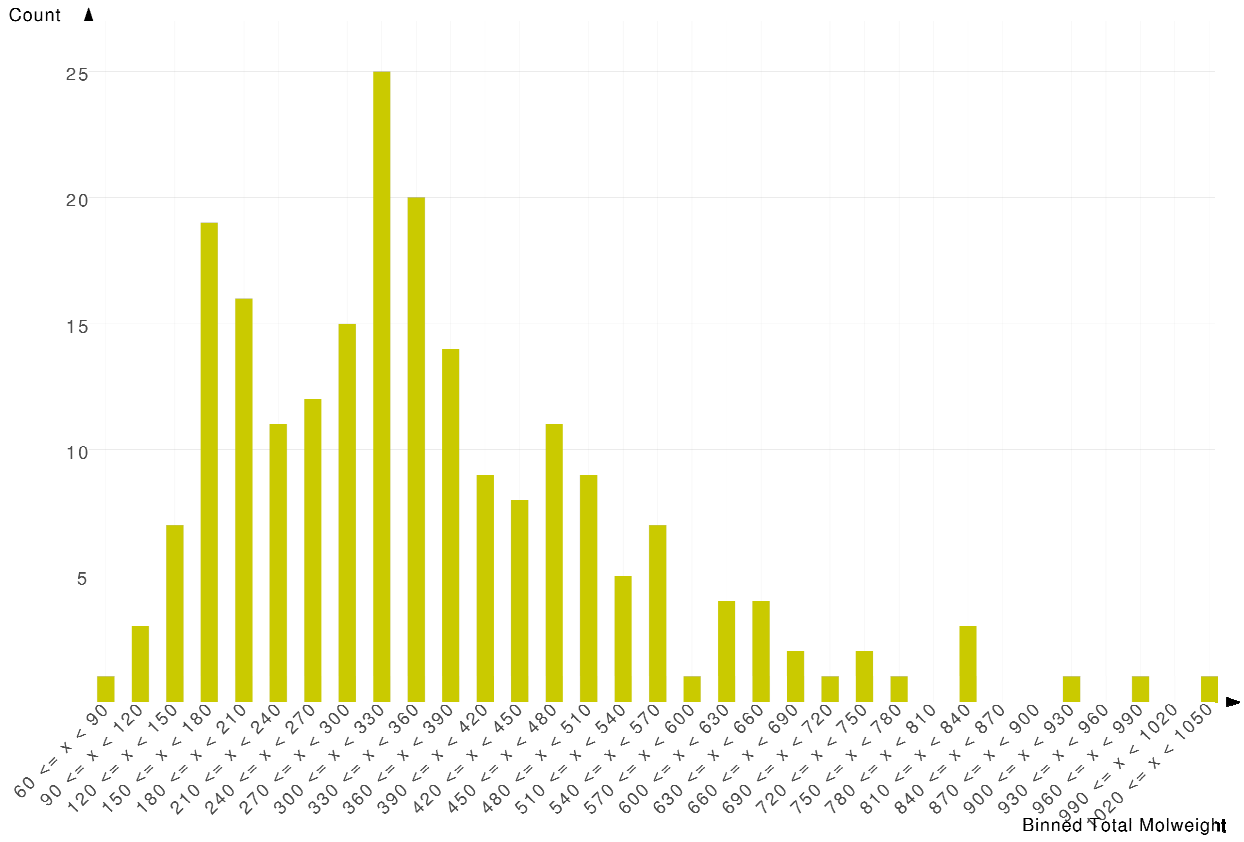


Figura 6 - Gráfico da distribuição do peso molecular dos 213 ligantes do conjunto de moléculas selecionado.

No gráfico da distribuição do peso molecular ao longo dos 213 ligantes (ver Figura 6), pode-se observar que as moléculas ligantes presentes neste conjunto possuem peso molecular situado entre 60 a 1050 g/mol, com uma diferença de até 990 g/mol entre o menor e o maior ligante deste conjunto. Entretanto, a maior parte destas moléculas dos ligantes se situa entre 120 a 570 g/mol, sendo que a boa parte dos ligantes se

concentram na faixa em torno de 300 a 330 g/mol. Esta diferença nas massas moleculares demonstra as diferenças significativas no número e tipos de átomos que cada uma destas moléculas de ligantes possui. Sendo assim, a demanda computacional (tempo) necessária para a realização dos cálculos individuais de *docking* molecular, neste conjunto tão diverso de moléculas, pode variar bastante.

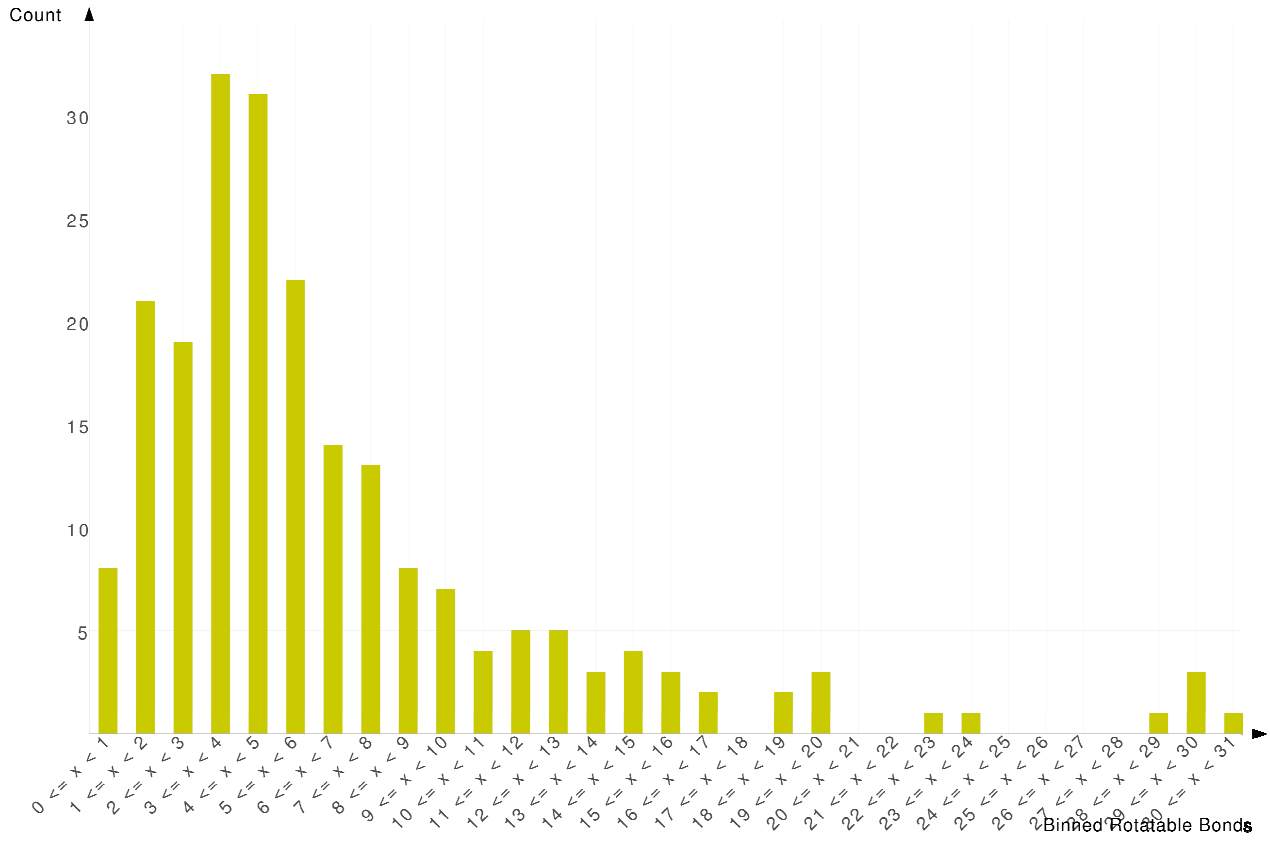


Figura 7 - Gráfico da distribuição do número de torções possíveis dos 213 ligantes do conjunto de moléculas selecionado.

Observa-se, no gráfico da distribuição do número de torções possíveis ao longo dos 213 ligantes (ver Figura 7), que o número de torções (ligações com possibilidade de rotação) apresentadas pelas moléculas deste conjunto varia entre 0 a 31 torções, sendo que a maior parte destas moléculas de ligantes possuem um número entre 0 e 10 torções. O número de torções apresentadas pelos ligantes pertencentes a este conjunto de

moléculas se concentra em 3 e 4 torções por ligante. Vale ressaltar que, como o programa AutoDock Vina leva em consideração os graus de liberdade da molécula de ligante durante os cálculos de *docking* molecular, ou seja, o número de torções ativas, quanto maior o número de torções que uma molécula ligante possui, maior será a demanda computacional do cálculo. Os demais gráficos obtidos através do programa DataWarrior podem ser encontrados na seção “Apêndice A” deste texto.

A preparação das moléculas, tal como a remoção das moléculas de água de cristalização, adição dos átomos de hidrogênio faltantes e adição das cargas dos átomos, foi realizada em nosso laboratório de forma não automatizada, através do uso da ferramenta computacional AutoDock Tools (ADT, 2016). A ferramenta AutoDock Tools, desenvolvida pelos mesmo grupo que desenvolveu os programas de *docking* molecular AutoDock e AutoDock Vina, foi utilizada por ser um programa gratuito e de código aberto “open source” e por ser a ferramenta recomendada para a preparação dos arquivos necessários para a realização de *docking* molecular pelos autores dos programas AutoDock e AutoDock Vina. Esta etapa inicial de preparação das moléculas é obrigatória para a realização dos cálculos de *docking* molecular com estes programas. Vale ressaltar que a versão 3.0 do programa GriDoMol realiza esta etapa de preparação das moléculas de forma automática, acessando algumas sequencias de comandos (scripts) disponíveis a partir da instalação do programa AutoDock Tools no computador do usuário. Sendo assim, o motivo pelo qual estas moléculas foram preparadas manualmente foi que o módulo do programa GriDoMol encarregado de automatizar esta etapa de preparação ainda não havia sido desenvolvido na época em que estes primeiros testes foram realizados.

Além dos parâmetros padrões necessários para a realização dos cálculos de *docking* molecular em si, outros parâmetros foram utilizados para melhorar os resultados obtidos, e também para tentar padronizar o tempo que cada cálculo individual de *docking* é realizado e, assim, viabilizar a reprodução dos testes de desempenho.

Tabela 4 – Tabela dos cálculos de docking no programa AutoDock com o complexo 1A1B ao utilizar diferentes configurações do algoritmo genético deste programa.

# Configuração	Parâmetros	Energia (Kcal/mol)	Tempo	Ligações de Hidrogênio
1	Padrão	-9,18	16m10s	6
2	ga_num_evals 25000000 ga_num_generations 10000 ga_run 50	-11,42	12h8m30s	5
3	ga_num_evals 25000000	-10	2h38m13s	5
4	ga_num_generations 10000	-9,18	16m15s	6
5	ga_run 50	-9,18	1h15m40s	6
6	ga_num_evals 25000000 ga_num_generations 10000	-10,57	2h25m51s	5
7	ga_num_evals 25000000 ga_run 50	-11,14	12h29m33s	4
8	ga_num_generations 10000 ga_run 50	-9,18	1h15m45s	6
9	ga_num_evals 12500000	-9,71	1h15m16s	5
10	ga_run 25	-9,18	37m54s	6
11	ga_num_evals 12500000 ga_run 25	-10,11	3h7m11s	4
12	ga_num_evals 12500000 ga_num_generations 10000 ga_run 25	-10,11	3h16m46s	4
13	ga_num_evals 12500000 ga_num_generations 10000	-9,71	1h18m43s	5
14	ga_num_evals 12500000 ga_run 50	-10,72	6h24m5s	6
15	ga_num_evals 25000000 ga_run 25	-10,81	6h33m46s	5
16	ga_num_generations 10000 ga_run 25	-9,18	37m59s	6
17	ga_num_evals 12500000 ga_num_generations 10000 ga_run 50	-10,72	6h33m53s	6
18	ga_num_evals 25000000 ga_num_generations 10000 ga_run 25	-11,08	6h23m24s	7
19	ga_num_evals 5000000 ga_run 20	-10,21	1h3m18s	7

Para o programa AutoDock, dois tipos de configurações para os cálculos de docking molecular foram utilizados: o cálculo padrão e o cálculo avançado. No cálculo padrão, nenhum dos valores do algoritmo genético foram alterados e, assim, utilizou-se apenas os valores definidos como padrão pelo desenvolvedor do programa AutoDock. Com o intuito de selecionar uma boa configuração de docking para utilizar em um cálculo avançado, realizou-se 19 cálculos de docking molecular com o mesmo complexo (PDB: 1A1B) utilizando diferentes configurações de docking para o algoritmo genético do programa AutoDock, com o objetivo de verificar a melhoria nas soluções de docking encontradas e seu custo em demanda computacional (Tabela 4). A configuração #18 foi escolhida por se tratar de uma configuração que obteve um excelente custo beneficio ao

encontrar uma solução de docking bastante estável (-11,08 kcal/mol) quando comparado com o cálculo padrão (-9,18 kcal/mol), levando quase a metade do tempo (6h23m24s) do que o cálculo com a configuração #2 que obteve a melhor solução de docking (12h8m30s) com uma diferença de -0,34 de energia livre de ligação (-11,42 kcal/mol).

Um dos principais parâmetros utilizado pelo algoritmo de busca do programa AutoDock Vina é o *exhaustiveness*. Este parâmetro é responsável por determinar a quantidade de buscas por uma conformação mais estável em cada etapa de um cálculo de *docking* molecular com este programa. Para os testes de desempenho computacional, utilizou-se dois valores distintos durante os testes realizados neste estudo, sendo eles o valor 8 (oito), o padrão do programa, e o valor 32 (trinta e dois), quatro vezes mais buscas quando comparado com o valor padrão. Já para os testes de comparação com os resultados experimentais, este valor foi alternado em progressão geométrica, em 1, 2, 4, 8, 16, 32 e 64. Ao aumentar este valor, aumenta-se também as chances de obter soluções de *docking* molecular mais estáveis ao custo de demanda computacional (tempo) para a realização dos cálculos de *docking* molecular.

Também foi modificado o valor do parâmetro *seed*, que indica a semente inicial para o gerador de números aleatórios, por onde o algoritmo genético começa sua busca, tornando-o fixo (número = -457186518) ao invés de aleatório. Este valor foi atribuído arbitrariamente, utilizando um dos valores gerados automaticamente em uma execução a priori da realização dos testes. Desta forma, tentou-se padronizar o processo de execução de cada cálculo de *docking*, com o intuito de tornar o procedimento reprodutível, uma vez que o algoritmo irá sempre começar do mesmo ponto.

4.5.1 DESEMPENHO COMPUTACIONAL

Para a realização dos testes de desempenho, utilizou-se a infraestrutura de um *grid* computacional montado no Laboratório de Química Teórica Medicinal (LQTM), localizado no Depto. de Ciências Farmacêuticas da UFPE. A configuração do computador *front-end*, responsável direto por executar efetivamente o programa GriDoMol para submeter os *jobs* para os demais computadores do ambiente de *grid* computacional, foi a seguinte: processador Intel dual core Xeon 2.80 GHz, contendo 2GB de memória RAM e executando o sistema operacional Windows 7 (GriDoMol 2.0) e AMD Athlon 64 X2 2.4 GHz, contendo 4GB de memória RAM e executando o sistema operacional Ubuntu 16.04 (GriDoMol 3.0).

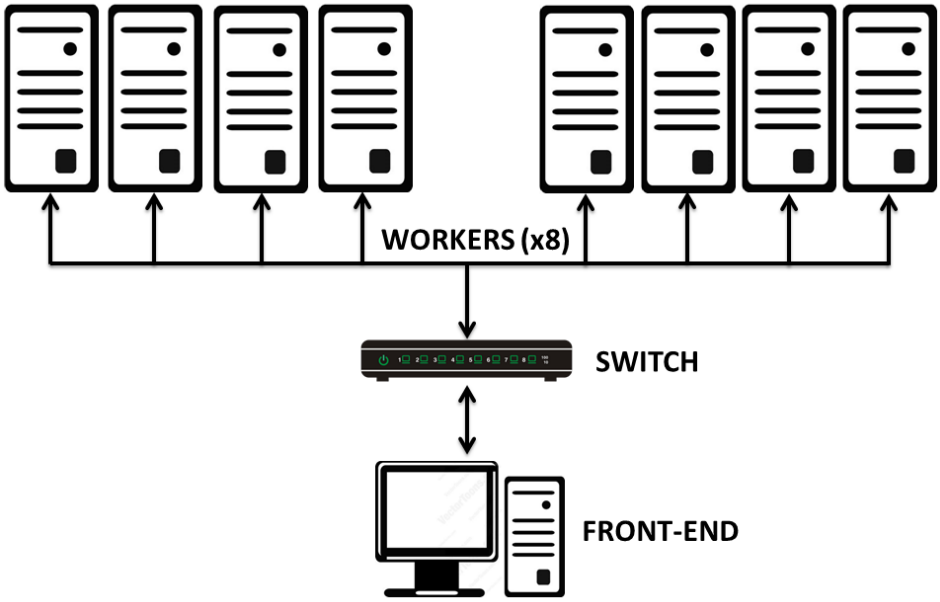


Figura 8 – Representação do ambiente de grid computacional formado por computadores homogêneos.

Os cálculos individuais de *docking* molecular foram executados em paralelo no ambiente de *grid* computacional homogêneo (ver Figura 8), contendo oito computadores

(com cada um contendo o componente *Worker* do OurGrid) de mesma configuração: dois processadores Intel Xeon e5410 quad-core 2.33 GHz (total de oito núcleos), com 16GB de memória RAM, executando o sistema operacional Linux Xubuntu 14.04. Todos os computadores e o *switch* se comunicaram utilizando interfaces de rede gigabit (1000Mb/s).

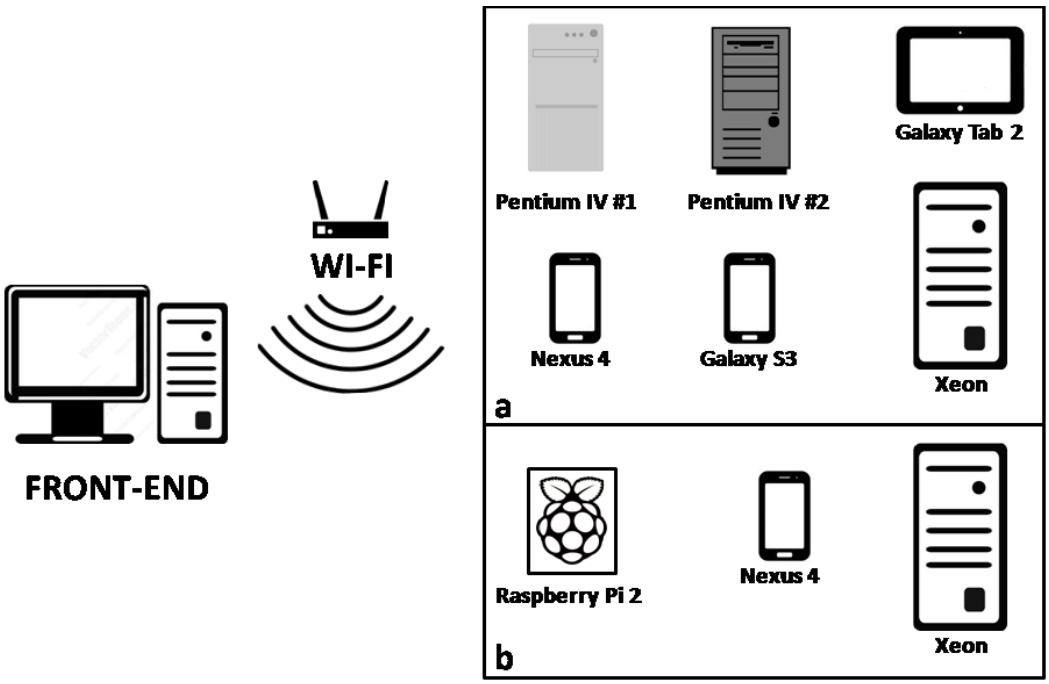


Figura 9 - Representação do ambiente de grid computacional formado por computadores heterogêneos para os cálculos com a versão 2.0 (a) e versão 3.0 (b) do programa GriDoMol.

Com o intuito de verificar o comportamento do ambiente de *grid* computacional em casos extremos de heterogeneidade, foram realizados testes de desempenho computacional utilizando dispositivos heterogêneos tais como, por exemplo, *smartphones* e *tablets*, para integrar o ambiente de *grid* computacional (ver Figura 9).

4.5.1.1 Ambiente de Grid Computacional Heterogêneo (GriDoMol 2.0)

Para os testes com a versão 2.0 do programa GriDoMol, o ambiente de grid

computacional heterogêneo foi formado por dois *smartphones* (LG Nexus 4 e Samsung Galaxy S3), um *tablet* (Samsung Galaxy Tab 2), dois computadores Pentium IV e um dos oito computadores Xeon já descritos anteriormente. As especificações técnicas de cada um destes dispositivos podem ser observadas na Tabela 5. Como os *smartphones* e o *tablet* utilizados neste estudo não possuem interface de rede ethernet, optou-se por utilizar uma rede sem fio para interligar todos os componentes do ambiente de *grid* computacional, incluindo os computadores, os quais utilizaram adaptadores de rede sem fio (wi-fi).

Tabela 5 – Especificações técnicas dos computadores, *smartphones* e *tablets* utilizados nos testes com o ambiente de grid computacional heterogêneo utilizando a versão 2.0 do programa GriDoMol.

Computador/Dispositivo	Número de núcleos de processamento utilizados	Clock do processador (GHz)	Memória RAM (GB)
Smartphone Nexus 4	3	1,5	2
Smartphone Galaxy S3	3	1,4	1
Tablet Galaxy Tab 2	1	1,0	1
Computadores Pentium IV	1	3,0	2
Computador Xeon	8	2,33	16

Para instalar o componente *Worker* do OurGrid, responsável pela execução de cada *task* (cálculo de *docking*) em cada um dos dispositivos ou computadores do ambiente de *grid* computacional, foi necessária a instalação de um sistema operacional Linux em conjunto com o sistema Android, através da utilização do aplicativo Linux Deploy (Linux Deploy, 2016). O sistema operacional instalado em cada um dos dispositivos Android utilizados neste estudo foi o Linux Debian, versão Wheezy, para a arquitetura armhf. Como os componentes da plataforma OurGrid foram escritos na linguagem de programação Java e, portanto, necessitam de uma versão da máquina virtual Java (JVM) instalada em cada destes sistemas Linux para funcionar, instalou-se a versão OpenJdk7

da máquina virtual Java e, desta forma, foi possível instalar a versão Linux do componente *Worker* em cada um destes dispositivos Android. Além disto, como não havia uma versão para a arquitetura armhf do programa AutoDock Vina no início dos testes com a versão 2.0 do programa GriDoMol, foi necessário instalar nestes sistemas Linux o compilador g++ para compilar o código fonte do programa AutoDock Vina para a arquitetura armhf e, assim, dar a estes dispositivos a capacidade de realizarem os cálculos de *docking* molecular com o programa AutoDock Vina. Após compilar o programa AutoDock Vina, foram feitos testes utilizando os arquivos de demonstração disponível no site do programa AutoDock Vina (AutoDock Vina, 2016) para validar o correto funcionamento do programa.

Em testes preliminares, observou-se certa instabilidade ao se tentar utilizar todos os núcleos de processamento destes *smartphones* e *tablet.* Por isso, optou-se por deixar um núcleo de processamento livre em cada um destes dispositivos, para permitir um bom funcionamento das operações básicas de gerenciamento interno do próprio sistema operacional Android. Assim sendo, para limitar o uso dos núcleos de processamento nos dispositivos moveis utilizados neste estudo, optou-se por configurar um *worker* por cada núcleo de processamento habilitado a executar os cálculos de *docking* molecular, o qual é um a menos que o total de núcleos de processamento disponíveis em cada *smartphone* e *tablet* utilizado neste trabalho (Nexus 4, Galaxy S3 e Galaxy Tab). Como esta limitação de estabilidade ao utilizar todos os núcleos de processamento só foi observada nos *smartphones* e *tablets*, o número de *workers* instalados em cada computador foi igual ao número de núcleos de processamento presente no mesmo. Além disto, para evitar que os cálculos de *docking* com o programa AutoDock Vina utilize todos os núcleos de processamento de cada *smartphone* e *tablet*, os cálculos de *docking* molecular executados pelo programa AutoDock Vina durante os testes com o ambiente de *grid* computacional heterogêneo utilizaram o parâmetro interno do programa (CPU = 1), no

intuito de limitar o uso dos núcleos de processamento para apenas um núcleo por cálculo. Desta forma, pode-se utilizar parte do poder computacional ocioso destes *smartphones* e *tablets*, mantendo-se a estabilidade ao deixar um núcleo livre para as operações internas do sistema operacional Android.

4.5.1.2 Ambiente de Grid Computacional Heterogêneo (GriDoMol 3.0)

Para os testes de desempenho computacional utilizando um ambiente de *grid* computacional heterogêneo com a versão 3.0 do programa GriDoMol, foram utilizados um *smartphone* (Nexus 4), um mini computador (Raspberry PI 2) e um dos computadores utilizados no teste homogêneo (Xeon). As características destes dispositivos pode ser observada na Tabela 6. Optou-se por utilizar apenas 3 núcleos de processamento em cada um destes dispositivos com o intuito de permitir uma comparação direta entre o desempenho obtido por cada tipo de núcleo de processamento ao utilizar o mesmo número de núcleos em todos os dispositivos, além de possibilitar que os dispositivos Nexus 4 e Raspberry PI 2 pudessem realizar uma quantidade maior de cálculos de docking molecular ao reduzir de 8 para 3 o número de núcleos de processamento ativos no computador Xeon.

Tabela 6 – Especificações técnicas dos computadores, *smartphones* e *tablets* utilizados nos testes com o ambiente de grid computacional heterogêneo utilizando a versão 3.0 do programa GriDoMol.

Computador/Dispositivo	Número de núcleos de processamento utilizados	Clock do processador (GHz)	Memória RAM (GB)
Smartphone Nexus 4	3	1,5	2
Computador Xeon	3	2,33	16
Raspberry PI 2	3	0,9	1

O sistema operacional utilizado no dispositivo Raspberry PI 2 foi a versão 16.04 da distribuição Linux Ubuntu Mate. Foram instalados o Openjdk7 e o programa AutoDock Vina através do repositório oficial deste sistema, e o componente *Worker* do OurGrid através de pacotes de instalação DEB.

A instalação do sistema operacional Linux nos dispositivos Android, procedimento necessário para utilizar o componente *Worker* do OurGrid nestes dispositivos, foi realizada através do aplicativo Linux Deploy utilizando os mesmos parâmetros e configurações descritos na seção anterior. Embora a versão do sistema operacional que foi utilizada neste teste com a versão 3.0 do programa GriDoMol tenha sido a mesma que foi utilizada no teste com a versão 2.0 do programa, ou seja, Debian Wheezy, as versões de muitos dos componentes do sistema operacional estão mais atualizadas neste teste com a versão 3.0 do programa GriDoMol uma vez em que há uma distância cronológica de cerca de um ano entre a realização dos testes com estas duas versões do programa GriDoMol. Além disto, quando foi iniciado o teste com o ambiente de grid computacional heterogêneo com a versão 3.0 do programa GriDoMol, o programa AutoDock Vina já estava disponível no repositório do sistema Linux utilizado e, por isso, não houve a necessidade de realizar a compilação do código fonte do programa AutoDock Vina neste teste.

4.5.2 MÉTRICAS DE DESEMPENHO

Durante os testes, para determinar o impacto da quantidade de computadores no desempenho do *grid* computacional, foi alternada a quantidade de computadores envolvidos no dimensionamento do *grid* computacional, variando-se no intervalo de 1 a 8 computadores em funcionamento, paulatinamente.

Através do parâmetro interno do programa AutoDock Vina “CPU=X”, onde X é o

número de processadores a serem utilizados durante os cálculos com este programa, foi possível alternar também o número de núcleos de processamento utilizados durante cada cálculo de *docking* molecular com este programa. Para determinar o desempenho *multithreading* do programa AutoDock Vina, foi ajustada a quantidade de núcleos de processamento (CPUs) utilizados durante cada cálculo, variando-se no intervalo de 1 a 8 núcleos de processamento utilizados por cálculo de *docking* molecular.

Com o intuito de observar precisamente o ganho de desempenho computacional nos diferentes níveis de paralelização dos cálculos realizados, definiram-se duas métricas de desempenho computacional. Estas métricas foram o Fator de Desempenho por Núcleos (FDN) e o Fator de Desempenho por Máquinas (FDM).

O Fator de Desempenho por (*Cores)* Núcleos (FDN) é o coeficiente que determina o ganho de desempenho observado, ao se utilizar o nível de distribuição *multithreading*, ou seja, nos diversos núcleos de processamento. Esta métrica mede o ganho de desempenho obtido pelo paralelismo interno do programa AutoDock Vina, devido à variação da quantidade de núcleos de processamento utilizados em cada cálculo de *docking* molecular. Seu valor é determinado pela divisão do tempo necessário para a realização dos cálculos de *docking* molecular em computadores que utilizem apenas um núcleo de processamento, pelo tempo de cálculo gasto pelo mesmo número de computadores, porém com mais de um núcleo de processamento sendo utilizado. Seu formalismo matemático pode ser encontrado na seguinte equação:

$$FDN = T(MxC1)/T(MxCy)$$

onde o termo T(MxC1) é o tempo total gasto por “x” máquinas que utilizam apenas um núcleo de processamento, enquanto o termo T(MxCy) é o tempo total gasto pelas mesmas “x” máquinas, porém, desta vez, utilizando “y” núcleos de processamento por máquina, para realizar o mesmo conjunto de tarefas (*Job*). Por exemplo, o FDN previsto

de um cálculo que utiliza quatro máquinas e todos os oito núcleos de processamento (cada) deveria ser igual a 8 (oito), uma vez que tende a ser idealmente 8 (oito) vezes mais rápido do que o cálculo utilizando quatro máquinas onde apenas um dos oito núcleos de processamento, em cada, está sendo utilizado. É importante ressaltar que este recurso *multithreading* do programa AutoDock Vina é uma opção interna deste programa, tendo sido implementada por seu desenvolvedor original. Sendo assim, o ganho de desempenho proveniente deste paralelismo não é o foco central deste trabalho, que enfatiza primariamente o ganho de desempenho devido à paralelização das tarefas em um ambiente de grid computacional, comandado pelo programa GriDoMol.

O Fator de Desempenho por Máquinas (FDM) busca quantificar o ganho de desempenho obtido na arquitetura distribuída do ambiente de grid computacional e é calculado segundo a seguinte equação:

$$FDM = T(M1Cx)/T(MyCx)$$

onde o termo T(M1Cx) indica o tempo total gasto por uma máquina utilizando “x” núcleos de processamento, desta máquina, e o termo T(MyCx) é o tempo total gasto, para executar a mesma tarefa, usando-se “y” máquinas e a mesma quantidade “x” de núcleos de processamento, em cada máquina, que foi utilizado no termo anterior. Utilizando exemplo parecido para ilustrar o conceito de FDN, o FDM previsto, ao utilizar quatro máquinas e oito núcleos de processamento cada, deveria, em tese, ser igual a 4 (quatro), uma vez que é previsto que seja 4 (quatro) vezes mais rápido quando comparado com o cálculo utilizando apenas uma máquina com os mesmos oito núcleos de processamento, no caso do escalonamento de performance no grid ser observado como essencialmente linear.

Ao medir individualmente o ganho de desempenho adquirido no nível *multithreading*, através do FDN, e o ganho de desempenho adquirido no nível de

ambiente distribuído de grid computacional, através do FDM, espera-se observar o ganho de desempenho adquirido por cada estratégia de paralelismo e determinar seus possíveis gargalos e vantagens. Desta forma, estes dados poderão auxiliar um usuário do programa GriDoMol na construção de seu próprio ambiente de grid computacional, indicando para ele, por exemplo, até que ponto vale a pena investir em mais núcleos de processamento ou máquinas e, assim, evitar os gargalos inerentes destas arquiteturas distribuídas.

4.5.2.1 Comparação com os Resultados Experimentais

Com o intuito de se observar a precisão com que os cálculos de *docking* molecular, executados pelo programa AutoDock Vina, se aproximam dos valores de energia livre de ligação observados experimentalmente, definiram-se duas métricas: a Diferença em Relação ao Experimental (*DRE*) e a Diferença Percentual Relativa (*DPR*). Além disto, para medir a precisão com que o programa AutoDock Vina encontra soluções de *docking* mais próximas da posição experimental, observada na estrutura cristalizada do complexo (Receptor + Ligante), utilizou-se uma medida comumente usada para quantificar a similaridade entre estruturas química: o Desvio da Raiz Média Quadrática, comumente conhecido como RMSD na sigla em inglês (Root Mean Square Deviation).

A métrica denominada de "Diferença em Relação ao Experimental" (DRE) é o número que representa o módulo da diferença entre o valor de energia livre de ligação (*binding*) predito pelo programa AutoDock Vina, e o valor de energia livre obtido experimentalmente. Seu formalismo matemático pode ser encontrado na seguinte equação:

$$DRE = | E(exp) - E(X) |$$

onde o termo E(exp) é o valor de energia livre observado experimentalmente enquanto o

termo E(X) é o valor de energia livre obtido pelo programa AutoDock Vina ao utilizar o valor “X” para o parâmetro exhaustiveness. Quanto mais próximo de zero for o valor obtido na métrica DRE, menor será a diferença entre os valores de energia livre de ligação da solução de *docking* predita pelo programa AutoDock Vina ao utilizar o valor “X” para o parâmetro exhaustiveness e o valor experimental. Por exemplo, se o valor de DRE for igual a 2 (dois) isto quer dizer que a diferença entre o valor de energia observado experimentalmente e o obtido pela solução de *docking* do programa AutoDock Vina é de 2 Kcal/mol, para mais ou para menos (por causa do módulo usado na equação que o define). Desta forma, espera-se observar o quão próximo as soluções de *docking* molecular preditas pelo programa AutoDock Vina estão dos resultados observados experimentalmente, conforme se alterna o valor utilizado para o parâmetro *exhaustiveness*.

A métrica denominada "Diferença Percentual Relativa" (DPR) é o número que representa a estabilidade relativa de cada complexo (ligante+receptor), em relação a um complexo de referência. Esta métrica compara, percentualmente, quanto cada complexo é menos (ou mais) energeticamente estável do que o complexo de referência, quando os valores de energia livre de ambos os complexos tenham sido obtidos nas mesmas condições, de forma que as soluções de *docking* sejam provenientes da utilização do mesmo valor do parâmetro *exhaustiveness*. O complexo escolhido para ser a referência para os demais complexos nos cálculos de DPR foi o complexo mais estável observado nos testes experimentais (PDB: 1SL3). Esta métrica é calculada segundo a seguinte equação:

$$DPR = \{[E(x) - E(1SL3)] / E(1SL3)\} * 100$$

onde o termo E(x) é o valor de energia livre do complexo “x”, enquanto o termo E(1SL3) é o valor de energia livre do complexo 1SL3 no mesmo conjunto de cálculos que o

complexo “x” (mesmo valor de *exhaustiveness*). Por exemplo, o complexo 1A30, nos cálculos com o valor oito para o parâmetro *exhaustiveness*, possui um DPR de -34, o que indica que este complexo é 34% menos estável, em termos de energia livre de ligação, quando comparado com o complexo referência (PDB: 1SL3), nas mesmas configurações, que neste caso é o valor 8 (oito) para o parâmetro *exhaustiveness*. Espera-se, desta forma, comparar os valores de DPR calculados para cada um dos 213 complexos (ligante+receptor), e assim, verificar a capacidade do programa AutoDock Vina em prever com precisão o comportamento de um conjunto de moléculas de interesse ao se alternar o valor do parâmetro *exhaustiveness*.

A métrica denominada "Desvio da Raiz Média Quadrática" (RMSD) é bastante utilizada para medir a diferença nas coordenadas tridimensionais (X, Y e Z) entre duas geometrias/conformações de uma mesma molécula. Através desta métrica, é possível comparar, para cada complexo, a distância entre a posição do ligante encontrada pela solução de *docking* molecular com o programa AutoDock Vina, e a posição deste mesmo ligante observada na estrutura co-cristalizada (referência experimental). Para duas coordenadas diferentes *a* e *b* de uma mesma estrutura molecular, o valor de RMSD é calculado conforme a seguinte equação (TROTT; OLSON, 2010):

$$\mathrm{RMSD}_{ab} = \max(\mathrm{RMSD'}_{ab}, \mathrm{RMSD'}_{ba})$$

onde

$$\mathrm{RMSD}'_{ab} = \sqrt{\frac{1}{N}\sum_{i}\min_{j} r_{ij}^2},$$

Na equação acima, o componente *N* é o total de átomos pesados do ligante (ou seja, excluindo os átomos de hidrogênio), o componente *min* é a menor distância entre um átomo *j* da estrutura *b* e o átomo *i* da estrutura *a*, sendo ambos o mesmo tipo de átomo, e

o componente r^2_{ij} é a distancia ao quadrado entre o átomo *i* da estrutura *a* e o átomo *j* da estrutura *b*. Quanto mais próximo um RMSD estiver do valor 0 (zero), menor será a distância média entre cada par de átomos correspondentes e, consequentemente, mais sobrepostas estarão estas estruturas. Desta forma, espera-se avaliar o nível de precisão com que o método de *docking* molecular empregado pelo programa AutoDock Vina é capaz de predizer corretamente a posição cristalográfica (obtida experimentalmente), geralmente a mais estável, de cada complexo (receptor + ligante), depois dos cálculos de re-*docking*.

Como a hipótese deste trabalho foi de realizar cálculos de *docking* molecular mais rápidos e mais precisos em um ambiente distribuído de *grid* computacional, quando comparados em relação à execução sequencial em apenas uma máquina, é importante analisar a capacidade dos programas de *docking* molecular utilizados pela plataforma GriDoMol em se aproximar dos resultados experimentais. Desta forma, acredita-se que ao utilizar estas métricas para realizar comparações entre os resultados experimentais e os resultados obtidos utilizando diferentes valores para o parâmetro exhaustiveness do programa AutoDock Vina, espera-se poder identificar, quantitativamente, o aumento da precisão do programa de docking molecular AutoDock Vina em prever o comportamento do conjunto de moléculas em suas soluções de docking ao custo do aumento da demanda computacional (tempo).

5 RESULTADOS E DISCUSSÃO

5.1 GriDoMol 2.0 (em C++)

O programa utilizado para criar, enviar e gerenciar os cálculos de *docking* molecular no ambiente de *grid* computacional durante os testes realizados neste trabalho foi o programa GriDoMol, em sua versão 2.0. Está nova versão do programa foi desenvolvida com o intuito de melhorar o desempenho e acrescentar novas funções a versão anterior (1.0), desenvolvida no estudo anterior durante o período de mestrado (FERREIRA, 2013).

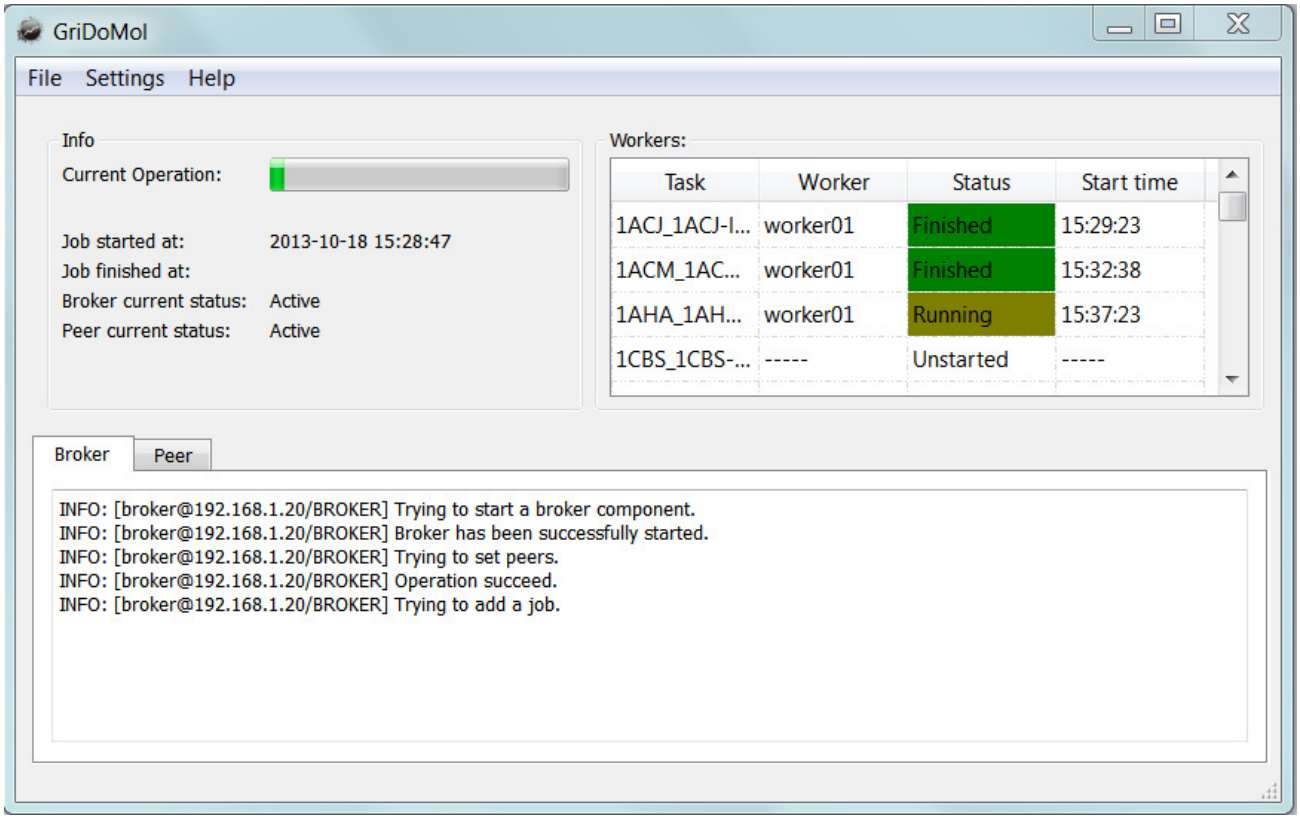


Figura 10 – Tela inicial do programa GriDoMol em sua versão 2.0.

A estrutura da tela inicial do programa GriDoMol, que pode ser observada na Figura 10, está dividida em três campos contendo as informações mais importantes para o acompanhamento do estado atual dos componentes do ambiente de *grid* computacional

e o andamento dos cálculos de *docking* submetidos. Estes campos estão divididos em *Info*, *Workers,* e as abas *Broker* e *Peer*.

É importante definir neste ponto alguns dos termos importantes que serão utilizados doravante no texto. Um *task* é uma tarefa individual encaminhada a uma máquina que integra o grid computacional. No caso específico deste trabalho com o GriDoMol, uma *task* é um cálculo de *docking* individual, ou uma instância de execução de um cálculo de *docking* molecular. Por outro lado, um *job* é um conjunto de tarefas (*tasks*) submetido no âmbito do grid computacional, onde cada *task* vai ser encaminhada a uma máquina ou núcleo livre do grid computacional, dependendo da configuração (*setup*) do cálculo de docking.

Ao se observar a Figura 10, no campo *Info* é possível visualizar algumas informações gerais sobre o andamento do *job* (conjunto de *tasks*), como a data e hora do início do *job*, data e hora do término do *job,* e uma barra de progresso contendo a porcentagem dos *tasks* (cálculos individuais de *docking* molecular) já concluídos neste *job*. Além disto, outra novidade da versão 2.0 do programa GriDoMol são as informações sobre o estado atual (ativo ou inativo) dos componentes *Broker* e *Peer* do OurGrid possibilitando-se, desta forma, um melhor controle de quais componentes estão em funcionamento ou parados.

O andamento de cada *task* (cálculo de *docking* molecular individual) pode ser observado no campo *Workers*. Neste campo há uma tabela contendo os detalhes de cada uma das *tasks* pertencentes a um *job* submetido ao ambiente de *grid* computacional. Para cada *task*, é possível observar informações, como qual é o cálculo de *docking*, usando uma nomenclatura no formato “Receptor_Ligante” (coluna *Task*), qual *Worker* está responsável por este cálculo (coluna *Worker*), o estado atual deste cálculo (coluna *Status*) e a hora em que o cálculo começou (coluna *Start time*). Para facilitar a visualização dos

estados atuais de cada *task*, na versão 2.0 do programa GriDoMol foram adicionadas cores distintas para cada um dos possíveis estados de uma *task*. A cor verde sinaliza que a *task* foi finalizada corretamente (*FINISHED*), amarelo para as *tasks* em execução (*RUNNING*), vermelho para os casos em que ocorreu uma falha com a *task* (*FAILED*) e a ausência de uma cor especial sinaliza que a *task* ainda não foi inicializada (UNSTARTED).

As abas *Broker* e *Peer* registram as atividades de cada um desses componentes do ambiente de grid computacional, exibindo informações de avisos, como por exemplo, quais máquinas estão disponíveis, as mensagens de erro e outras mensagens de comunicação de cada um destes componentes do sistema.

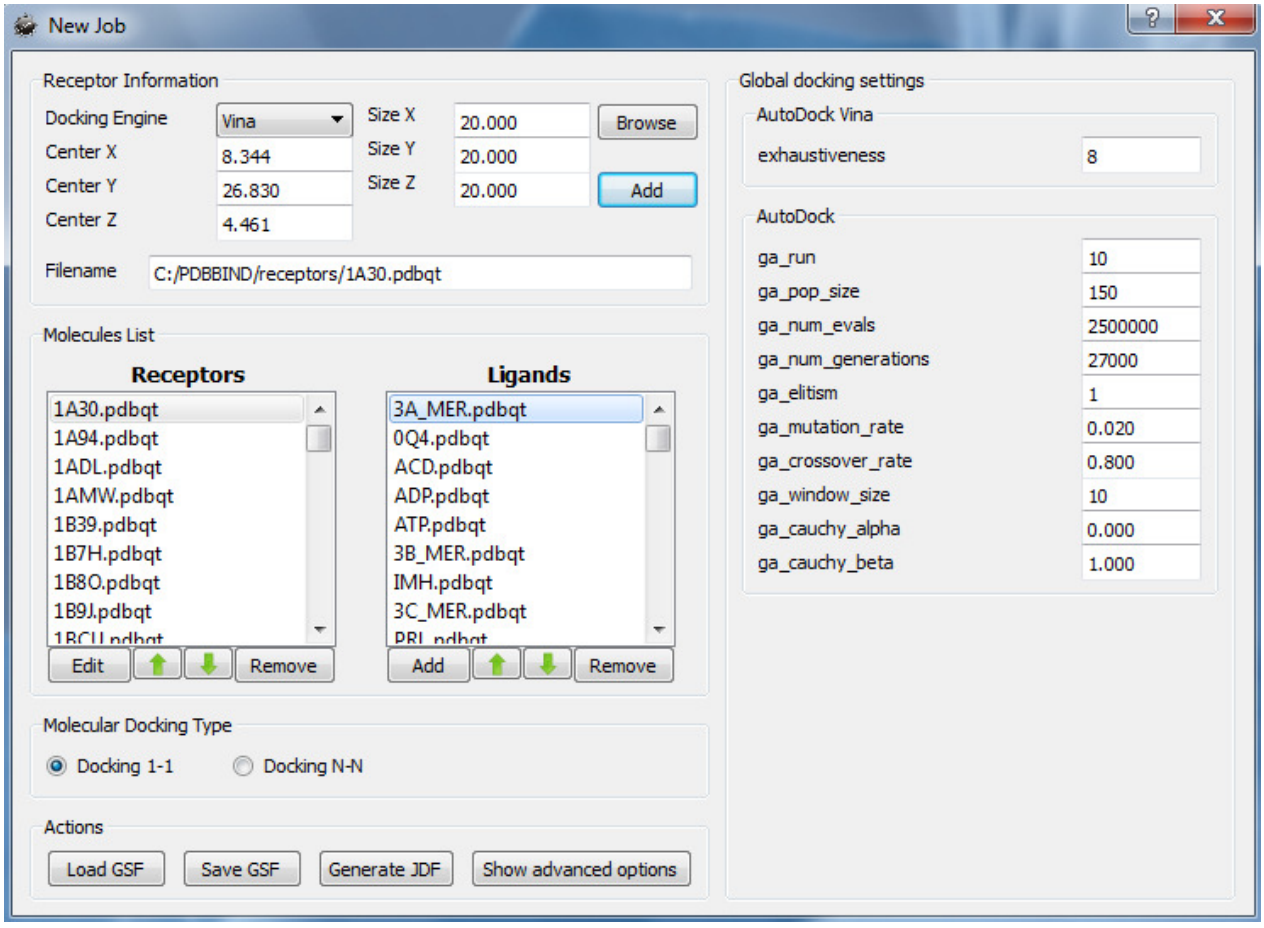


Figura 11 – Tela do programa GriDoMol para criação de um job. a) Tela solicitando as informações básicas e obrigatórias para criação de um *job* de *docking* molecular. b) Tela de criação do *job* ao ativar as opções avançadas para o *setup* dos cálculos de *docking* molecular.

O programa GriDoMol possui uma interface gráfica para a preparação dos arquivos necessários para os cálculos de *docking* molecular no ambiente de *grid* computacional (ver Figura 11). Nesta interface, são solicitados alguns dados de entrada para o usuário, sendo dividida entre os campos *Receptor Information*, *Molecules List, Molecular Docking Type*, *Actions* e *Global Docking Settings*.

No campo *Receptor Information* pode-se adicionar os receptores (alvos) que participarão nos cálculos de *docking* molecular. Neste campo é solicitado ao usuário o arquivo PDBQT contendo a estrutura do receptor e as coordenadas tridimensionais (x, y e z) do centro de seu sítio ativo e a área de busca a partir do centro do sítio ativo. Estas informações tridimensionais do centro do sítio ativo e a área de busca podem ser obtidas através do uso da ferramenta AutoDock Tools (ADT, 2015), que é totalmente compatível com os programas AutoDock e AutoDock Vina. Além disto, ao contrário da versão anterior do programa GriDoMol (1.0), onde era obrigatório que todos os cálculos fossem realizados pelo mesmo programa de *docking* molecular, nesta versão 2.0 é possível escolher, para cada receptor, qual programa de *docking* molecular (AutoDock ou AutoDock Vina) ficara encarregado de realizar os cálculos que envolvam este receptor.

No campo *Molecules List* consta a lista de moléculas que farão parte do *job* (conjunto de cálculos de *docking*). Neste campo é possível adicionar (ligantes), editar, remover ou reordenar (através das setas para cima ou para baixo, mudando-se a ordem de prioridade dos cálculos individuais) o conjunto de moléculas que participaram dos cálculos de *docking*.

No campo *Molecular Docking Type* determina-se qual a modalidade de cálculo de *docking* que será realizado, podendo-se escolher entre o *docking* N-N ou *docking* 1-1. Na modalidade *docking* N-N, o *docking* de todos os ligantes será realizado com todos os receptores, no que costuma ser chamado de *crossdocking*. Já na modalidade *docking* 1-1,

cada receptor estará envolvido apenas no cálculo com o ligante que estiver na mesma sequência (numeração linear) da lista, sendo este tipicamente o caso do *redocking*.

O campo *Actions* exibe as ações que o usuário pode tomar após preencher os dados do *job*. Nesta etapa, é possível carregar (*load*) ou salvar (*save*) a lista de moléculas e configurações no formato interno de arquivo GriDoMol Settings File (GSF), próprio do programa GriDoMol, para uma posterior reutilização deste conjunto de moléculas e configurações, ou mesmo, para o usuário guardar como *backup* um histórico do *setup* que foi utilizado nos cálculos deste *job*. Ao se carregar um arquivo no formato *GriDoMol Settings File* (GSF), é possível realizar alterações de dados, como: adicionar e/ou remover moléculas, modificar as coordenadas da localização do sitio ativo, o tamanho da área de busca em torno do sitio ativo, e reselecionar o programa de *docking* que irá realizar cada cálculo individual de *docking*. Ainda neste campo *Actions*, é possível gerar o arquivo *Job Definition File* (JDF). Este arquivo possui as definições do *job* (conjunto de cálculos individuais de *docking*) em um formato reconhecido pelo ambiente de *grid* computacional OurGrid, e é necessário para a realização dos cálculos de *docking* molecular neste ambiente de *grid* computacional. Além disto, nesta versão 2.0 do programa GriDoMol, adicionou-se a possibilidade de modificar algumas opções avançadas de cada um dos programas de *docking* molecular contemplados pelo sistema GriDoMol. Ao clicar no botão *show advanced options*, a tela de criação de *jobs* é expandida e os principais parâmetros dos programas AutoDock e AutoDock Vina são exibidos. Vale ressaltar que estas opções podem afetar significativamente a precisão e a demanda computacional dos cálculos de *docking* molecular com estes programas e que estes parâmetros não devem ser alterados, a não ser que o usuário tenha certeza do que está fazendo.

5.2 GriDoMol 3.0 (em Java)

As versões 1.0 e 2.0 do programa GriDoMol, ambas desenvolvidas utilizando-se a linguagem de programação C++ e a biblioteca de desenvolvimento QT (QT, 2016), haviam sido projetadas para a realização de cálculos de *docking* molecular de forma distribuída em um ambiente de *grid* computacional, acessado através de um programa *desktop* instalado no computador do usuário. Sendo assim, muitas de suas funções estavam intimamente ligadas à interface gráfica *Desktop*. Como o objetivo deste presente trabalho de pesquisa é contemplar acesso ao ambiente de *grid* computacional também de forma remota através de interfaces Web convencionais e dispositivos móveis (*smartphones* e *tablets*), além do acesso via programa *desktop* convencional, foi necessária uma recodificação completa do programa GriDoMol. Esta versão mais nova do programa GriDoMol (3.0), desenvolvida utilizando-se a linguagem de programação Java, foi projetada e desenvolvida de forma completamente nova, partindo-se do zero, para possibilitar a integração entre as versões do programa para *Desktop, Web* e *Web Mobile,* como, também, para facilitar a integração com outros projetos do Laboratório de Química Teórica Medicinal (LQTM).

A tela principal do programa foi reformulada para tornar o programa ainda mais amigável ao usuário final (ver Figura 12). Esta interface está subdividida em três campos, sendo eles: *General Information, Available Workers e Job*.

Observa-se, no campo *General Information,* as informações sobre o estado atual (iniciando, iniciado, parando e parado) dos componentes *Broker* e *Peer* da plataforma OurGrid, e o progresso, em porcentagem, dos cálculos (*tasks*) já concluídos do conjunto de cálculos de *docking* submetidos pela plataforma GriDoMol.

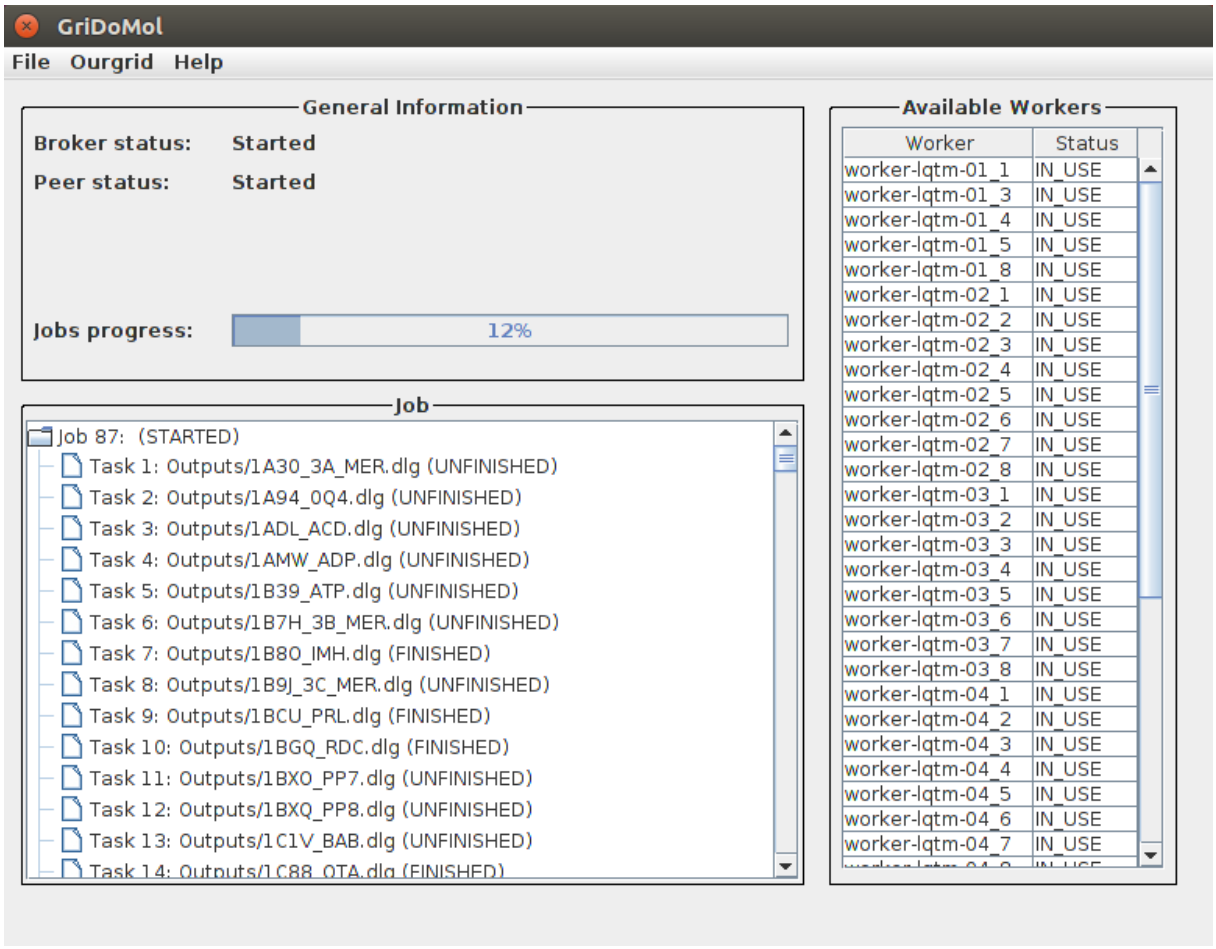


Figura 12 – Tela principal do programa GriDoMol (versão 3.0, em Java), executado à partir do sistema operacional Linux.

O campo *Available Workers* informa ao usuário o estado atual de cada um dos *workers* disponíveis no ambiente de *grid* computacional. Este componente está presente em cada um dos computadores e dispositivos com capacidade de efetivamente realizar os cálculos de *docking* molecular. Desta forma, é possível determinar o estado atual de cada um dos *Workers* presentes no ambiente de *grid* computacional. Geralmente, um worker pode estar executando um cálculo de *docking* molecular (status = IN_USE) ou ocioso (status = IDLE), por exemplo.

O usuário pode acompanhar o estado atual de cada um dos *jobs* (conjunto de cálculos) submetidos ao ambiente de *grid* computacional através do programa GriDoMol e

suas respectivas *tasks* (cálculos individuais) através do campo *Job*. Os estados possíveis, tanto para um *job* quanto para uma *task*, são: não iniciado (*UNSTARTED*) e finalizado (*FINISHED*). Além destes dois estados, há também o estado iniciado (*STARTED*), o qual é atribuído ao job quando este é submetido ao ambiente de grid computacional. Ao clicar-se com o botão direito em um *job*, aparece um menu onde é possível cancelar a execução do *job* escolhido. Ao contrario da versão 2.0 do programa GriDoMol, esta versão 3.0 tem a capacidade de adicionar múltiplos *jobs* em uma fila de espera e, desta forma, minimizar o gargalo típico da arquitetura distribuída do tipo grid computacional, uma vez que as máquinas que poderiam ficar ociosas esperando o último cálculo do *job* anterior terminar, podem agora começar a execução dos cálculos do próximo *job* da fila, maximizando-se assim a utilização da capacidade computacional do grid.

A submissão dos cálculos de docking molecular é realizada através da interface *Create/Continue a Job* (Figura 13). Com o intuito de permitir que cada usuário organize seus jobs em uma única pasta de trabalho própria, optou-se por implementar a opção de mudança de pasta de trabalho (*workspace*), através do botão *Change*, facilitando o gerenciamento destes jobs. Ao carregar um *workspace*, todos os jobs que estão dentro desta pasta serão automaticamente carregados na tabela, onde o usuário pode ver o tipo de cada *job*, *Virtual Screening* (VS) ou *Reverse Vaccinology* (RV), o nome do job e o progresso em que a execução do *job* foi interrompida. Ao submeter um *job* através do botão *Choose*, o programa GriDoMol verifica quais cálculos de docking molecular ainda não foram realizados e submete uma versão customizada deste *job* contendo apenas estes cálculos restantes ao ambiente de grid computacional. Esta abordagem foi elaborada de forma a permitir a continuação de *job* que foi iniciado anteriormente e foi interrompido por fatores diversos, evitando que todos os cálculos de docking molecular do job recomecem do início.

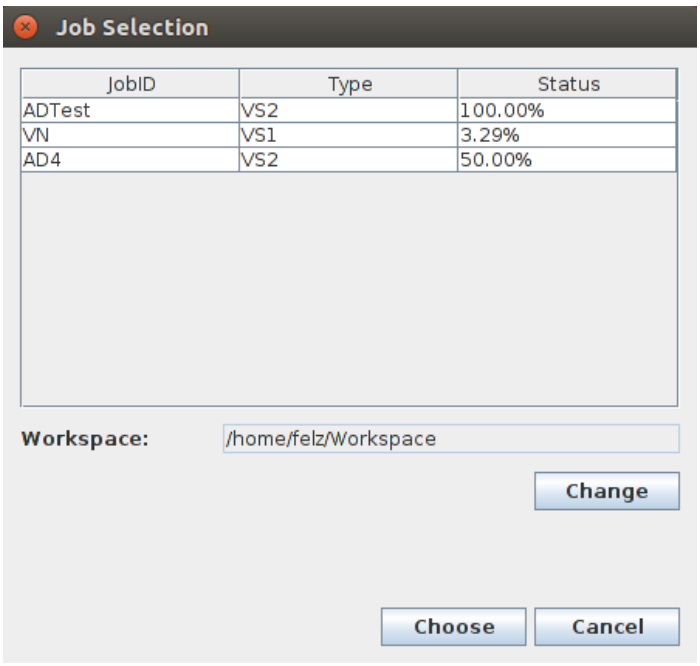


Figura 13 – Tela do programa GriDoMol responsável pela submissão de jobs ao ambiente de grid computacional.

A interface gráfica para a preparação dos arquivos necessários nos cálculos de *docking* em ambiente distribuído é acessada através da opção *Create a new Job*, disponível no menu *file* da tela principal. Com o intuito de tornar a interface mais amigável aos usuários com pouca experiência, e guia-los durante a criação do *job* (conjunto de cálculos), esta tela foi desenvolvida no estilo *wizard*, onde o seu conteúdo foi subdividido em várias etapas. A vantagem deste estilo (*wizard*) é que estas etapas ajudam o usuário a focar em poucos dados por vez, ao invés de colocar muitas informações na tela de uma só vez, facilitando a interação entre o usuário e o programa GriDoMol, principalmente no caso de usuários iniciantes. Além disto, cada etapa possui textos auto-explicativos indicando quais dados são esperados, efetivamente guiando o usuário durante a criação do *job*. Este *wizard* de criação de *jobs* foi subdividido em 6 etapas, sendo elas: Inicial (*Start*), Receptores (*Receptors*), Ligantes (*Ligands*), Lista de Tarefas (*List of Tasks*), Configurações Globais (*Global Settings*) e Finalização (*Finish*).

Na etapa *Start* (ver Figura 14a), é oferecida ao usuário a opção de criar um novo *job* desde o início ou editar outro *job* criado anteriormente. Caso o usuário opte por

continuar um *job* anterior, todas as moléculas e configurações utilizadas pelo *job* escolhido serão importadas e estarão disponíveis para a edição deste novo *job*. Esta opção é útil nos casos onde o usuário deseja apenas acrescentar ou modificar alguns dados a este *job* tal como, por exemplo, adicionar e remover moléculas, ou alterar as configurações globais utilizadas pelos programas de *docking* molecular.

Na etapa *Receptors* (ver Figura 14b) é possível adicionar todos os receptores (alvos) que serão utilizados durante os cálculos de *docking* molecular. Para adicionar um novo receptor, basta clicar no botão *Add* após preencher as informações solicitadas no campo *Receptor Information*, tais como informar a localização do arquivo contendo a estrutura tridimensional do alvo, preencher as coordenadas tridimensionais (x, y e z) do centro de seu sítio ativo, o tamanho da área de busca a partir do centro do sítio ativo (também em x, y e z) e, opcionalmente, selecionar o arquivo "flexres" contendo os resíduos de aminoácido do receptor que levará em conta os graus de liberdade (torções) das cadeias laterais destes aminoácidos. Estas informações podem ser obtidas através do uso de programas com interface gráfica de visualização do alvo biológico, como por exemplo, o próprio AutoDock Tools (ADT, 2016), que é a ferramenta mais recomendada para a preparação das moléculas que serão utilizadas nos cálculos de *docking* molecular com os programas AutoDock e AutoDock Vina. Além disto, ainda no painel *Receptor Information*, existe a opção de escolher qual programa de *docking* molecular (*docking engine*) será encarregado de realizar os cálculos de *docking* molecular com cada receptor, podendo escolher, atualmente, entre os programas AutoDock e AutoDock Vina. Para verificar ou editar as informações de um receptor previamente adicionado, basta selecioná-lo na lista *Receptors* que suas informações apareceram no campo *Receptor Information* para que o usuário possa modificar suas informações e readicionar este receptor. Para remover um receptor, basta selecionar o receptor na lista *Receptors* e clicar no botão *Remove Selection*.

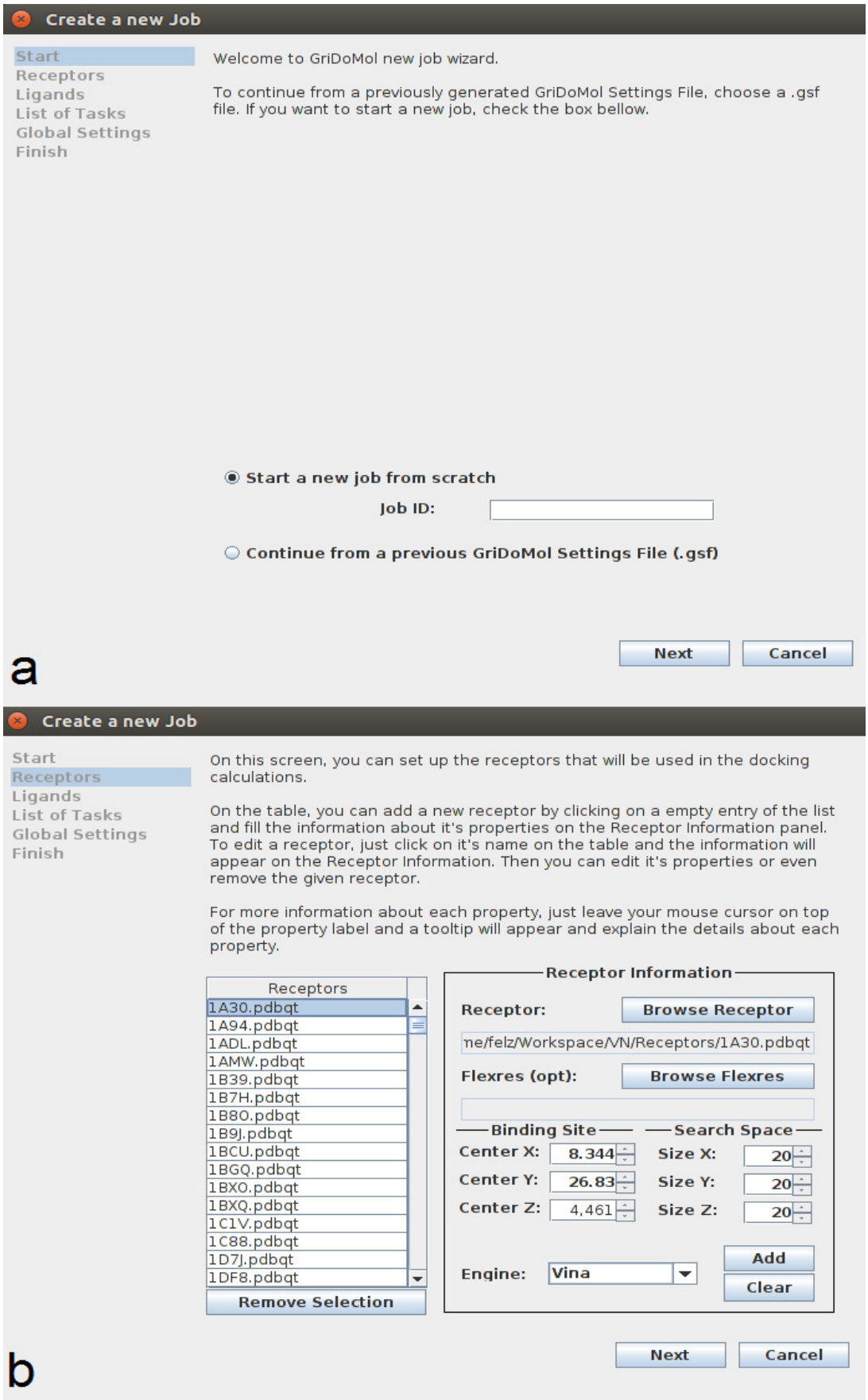


Figura 14 – Interface gráfica estilo *wizard* para criação de *jobs* no programa GriDoMol (versão 3.0, em Java). a) Etapa *Start.* b) Etapa *Receptors*.

A etapa *Ligands* (ver Figura 15a) é responsável pela adição das moléculas de ligantes. Nela, o usuário poderá escolher o local onde se encontram os arquivos com as estruturas tridimensionais do ligante através do botão *Add Ligands* e os ligantes são incluídos na lista *Ligands.* Para apagar um ligante previamente adicionado, basta selecioná-lo na lista *Ligands* e clicar no botão *Remove Selection*. No campo *Ligand Information* o usuário pode verificar o local onde se encontra o ligante selecionado na lista *Ligands*.

Durante a etapa *List of Tasks* (ver Figura 15b), é possível escolher como serão organizados os cálculos de *docking* molecular entre os receptores e ligantes adicionados nas etapas anteriores (*Receptors* e *Ligands*, respectivamente). Há basicamente duas modalidades de *docking* contempladas pelo programa GriDoMol, sendo elas o *docking* 1-1 (*Redocking*) e o *docking N-N* (*Crossdocking*). Na modalidade *docking* 1-1, para adicionar um novo cálculo (*task*), basta ir ao campo *Task Information* escolher qual conjunto (receptor + ligante) será usado no cálculo individual de *docking* e clicar no botão *Add*. Na modalidade *crossdocking,* o *docking* de todos os ligantes será realizado com todos os receptores. Neste caso, como não há a necessidade de formar os conjuntos de complexos (receptor + ligante) para os cálculos de *docking* nesta modalidade, a lista de cálculos é desabilitada uma vez que o programa GriDoMol irá criar, automaticamente, todas as combinações possíveis entre os receptores e ligantes disponíveis. O usuário pode optar por remover da lista um dos cálculo de docking entre um dado receptor e ligante, para isso, basta selecionar a linha contendo que deseja remover e clicar no botão *Remove Row*. Verificou-se durante os testes de desempenho computacional (seção 6.2.2) que a ordem com que os cálculos de docking molecular são realizados impacta bastante o desempenho geral do ambiente de grid computacional, deixando vários computadores e/ou núcleos de processamento ociosos. Com o intuito de tentar minimizar este gargalo computacional, colocando os cálculos com maior demanda computacional no início,

optou-se por desenvolver um módulo do programa GriDoMol capaz de prever a demanda computacional aproximada de um cálculo de docking molecular, baseando-se no tamanho da zona de busca em torno do sítio ativo (size x, y e z) e no número de torsões do ligante, em casos de empate no tamanho da zona de busca. Para permitir que o programa GriDoMol ordene a execução dos cálculos de docking molecular, basta marcar a opção *Sort tasks by predicted demand (descending)*. Vale ressaltar que este ordenamento dos cálculos de docking molecular baseado na demanda computacional predita realizado pelo programa GriDoMol é apenas uma aproximação, pois a única maneira de realizar um ordenamento exato desta natureza seria após observar a demanda computacional real ao termino da execução de cada um destes cálculos de docking molecular, o que não faria sentido para o usuário comum uma vez que, com as soluções de docking em mãos, não há uma necessidade de resubmeter estes cálculos.

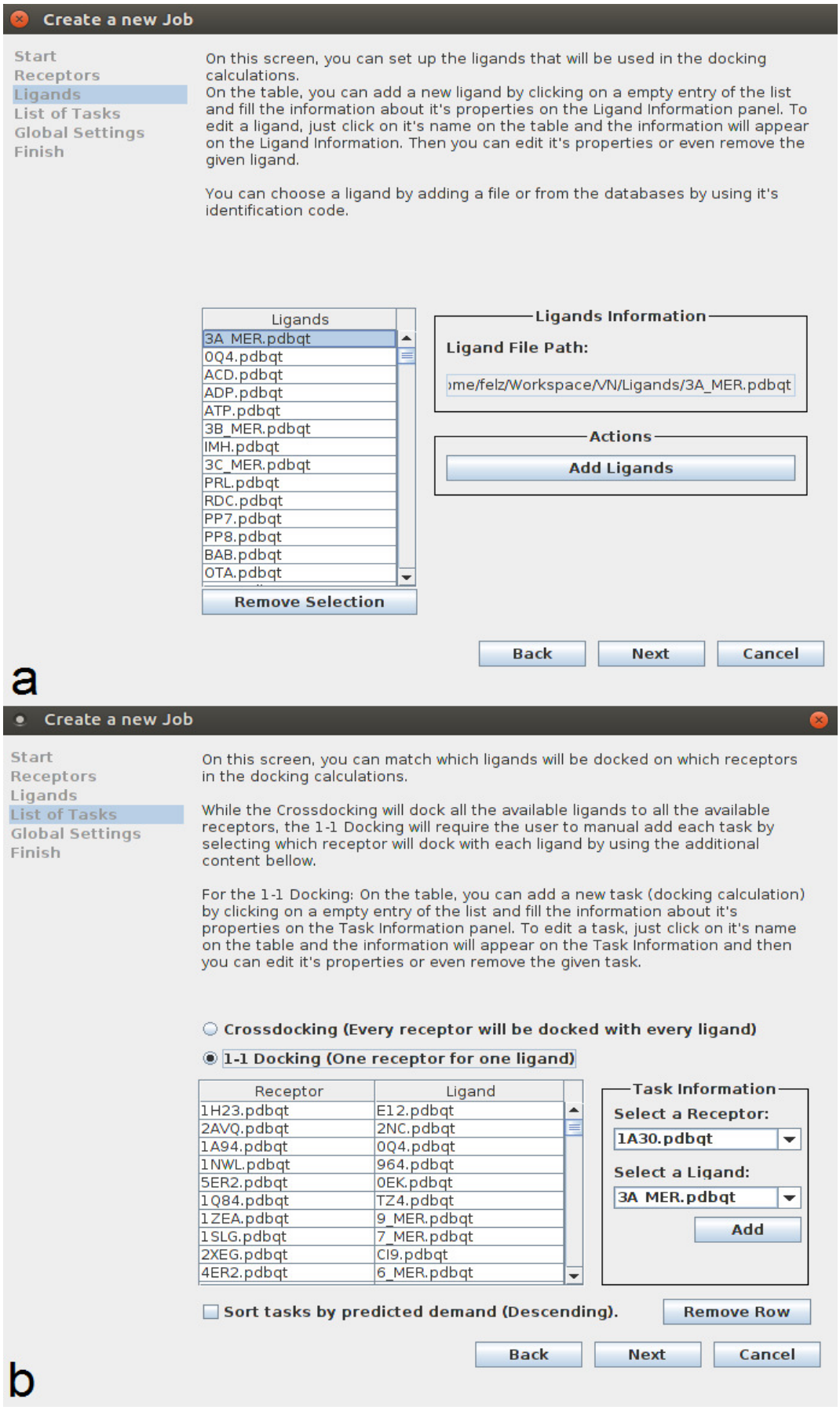


Figura 15 – Interface gráfica estilo *wizard* para criação de *jobs* no programa GriDoMol (versão 3.0, em Java). a) Etapa *Ligands*. b) Etapa *List of Tasks*.

Na etapa *Global Settings* (ver Figura 16a), o usuário pode personalizar alguns dos principais parâmetros utilizados pelos programas de *docking* molecular disponíveis. Vale ressaltar que os valores padrão destes parâmetros, em cada programa de *docking* molecular, foram amplamente testados e calibrados por seus desenvolvedores originais e se adéquam à maioria dos casos que se deseja fazer o cálculo de *docking* molecular. A alteração destes valores pode afetar significativamente a precisão e o tempo de cada cálculo realizado por estes programas, e estas modificações só devem ser realizadas por um usuário avançado.

A etapa *Finish* (ver Figura 16b) conclui o processo de criação de um *job* e permite que o usuário crie os arquivos *GriDoMol Settings File* (GSF) e *Job Definition File* (JDF). O formato de arquivo GSF foi implementado com o intuito de facilitar uma futura reutilização e/ou alteração do conjunto de moléculas selecionadas para os cálculos e seus parâmetros. No início (etapa *start*) da criação de um *job* é possível carregar e continuar a partir de um arquivo GSF já pré-existente. Por outro lado, o arquivo JDF contem a estrutura do *job* e suas *tasks* (cálculos individuais), no formato aceito pela plataforma OurGrid, e é necessário para submeter o conjunto de cálculos de *docking* molecular no ambiente de *grid* computacional.

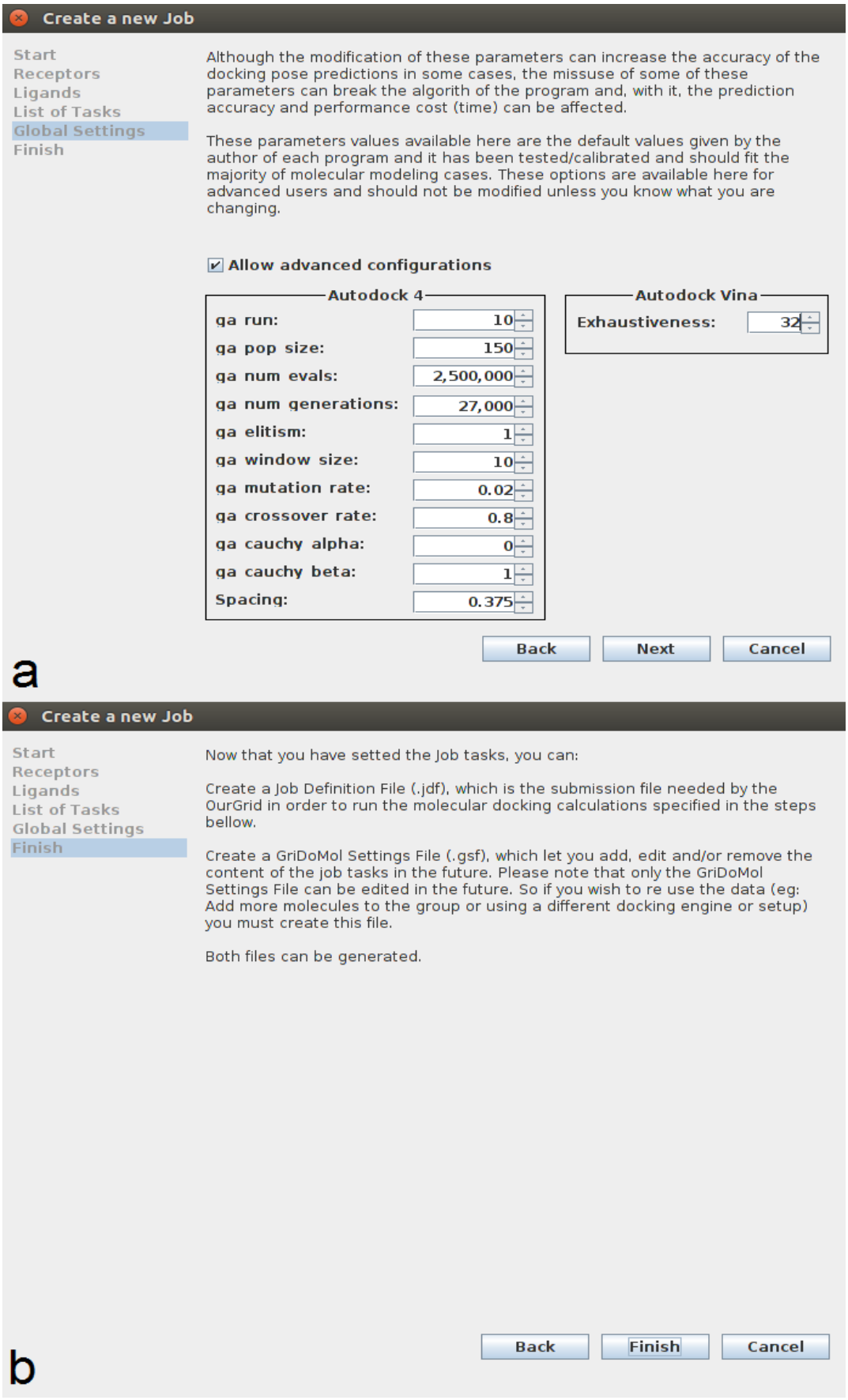


Figura 16 – Interface gráfica estilo *wizard* para criação de *jobs* no programa GriDoMol (versão 3.0, em Java). a) Etapa *Global Settings*. b) Etapa *Finish*.

Embora não exista, neste momento, uma tela para análises de resultados, algumas funções capazes de analisar os resultados de *docking* molecular já foram implementadas no código do programa *GriDoMol*. Estas funções foram utilizadas para coletar e análise dos valores de energia livre (*score*) e os desvios (*RMSD*) entre as soluções encontradas pelo programa AutoDock Vina. Estes valores foram utilizados durante os testes de comparação com os resultados experimentais.

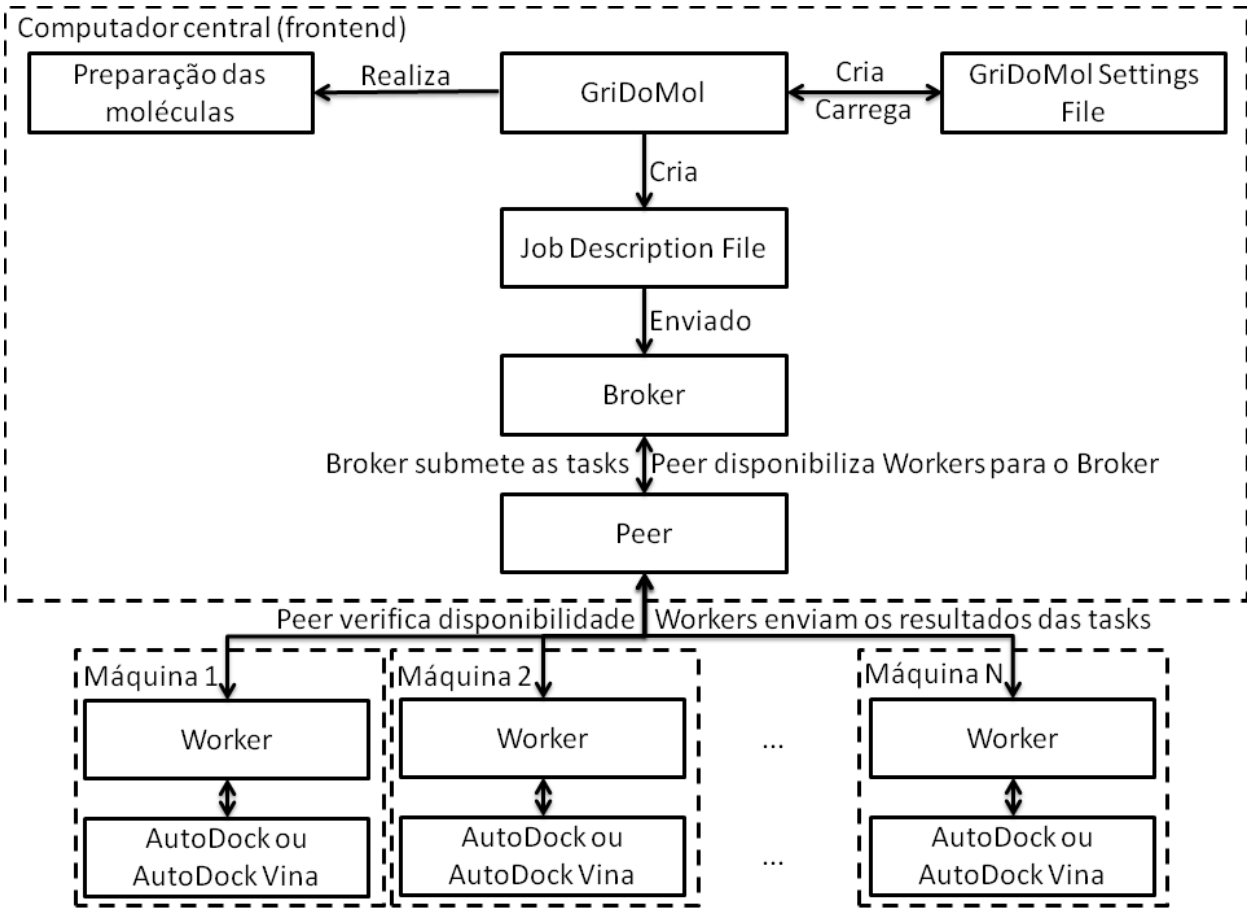


Figura 17 - Esquema de comunicação entre o programa *GriDoMol* e os demais programas utilizados neste trabalho.

A Figura 17 mostra o esquema de comunicação entre o programa *GriDoMol* e os demais programas utilizados neste trabalho. O Fluxo de execução começa pela preparação inicial das moléculas que serão utilizadas, como, por exemplo, adicionar

átomos de hidrogênio, remover as moléculas de água e converter os arquivos contendo as estruturas moleculares para o formato PDBQT. Esta preparação inicial pode ser realizada de forma automática com a versão 3.0 do programa GriDoMol, através da interface de criação do job para Virtual Screening. Além desta preparação inicial, o usuário pode, através do programa GriDoMol: i) Criar ou carregar os arquivos *GriDoMol Settings File* (GSF) contendo as informações sobre as moléculas que serão utilizadas; ii) Criar o arquivo *Job Description File* (JDF), utilizado para realizar os cálculos de *docking* molecular de forma distribuída, e submetê-lo para o componente *Broker*.

Após enviar o *job* (conjunto de cálculos de docking molecular) para o componente *Broker*, o mesmo se encarrega de distribuir as *tasks* (cálculos individuais de *docking*) de forma otimizada entre os *Workers* presentes em cada máquina encarregada de realizar cálculos de *docking* no ambiente de *grid* computacional. O componente *Peer* analisa a disponibilidade das máquinas em tempo real, e informa o componente *Broker* caso alguma máquina fique indisponível. Cada máquina que executa o componente *Worker* realiza cálculos de *docking* utilizando o programa AutoDock Vina. Quando o cálculo de *docking* termina, os resultados são enviados para o computador do usuário (*frontend*).

Para determinar o poder computacional do ambiente de *grid* computacional, controlado pelo GriDoMol, em FLOPS (operações de ponto flutuante por segundo) utilizou-se neste estudo a versão Intel da ferramenta Linpack (Linpack, 2016). A ferramenta Linpack é muito utilizada para determinar o poder computacional, incluindo testes de desempenho de supercomputadores (Top 500 Supercomputers, 2016). O resultado obtido para a execução padrão do programa Linpack em um dos computadores usados neste trabalho, em nosso laboratório, foi de 63,4 GFLOPS (1 GFLOP = 10^9 FLOPS) e, como os computadores que compõem o ambiente de *grid* computacional deste presente trabalho são homogêneos (ou seja, todos iguais entre si), o poder computacional estimado de todo o ambiente de grid computacional homogêneo contendo oito destes

computadores é de aproximadamente 507,2 GFLOPS.

5.3 Resultados dos Testes de Desempenho

Além dos testes de desempenho realizados utilizando-se os computadores homogêneos (Intel *Xeon*) do grid computacional, também foram realizados testes adicionais com um ambiente de grid computacional heterogêneo, formado por computadores e dispositivos diversificados, tais como computadores com processadores *Pentium* IV, *smartphones* (Nexus 4 e Galaxy S3), um computador do tipo Raspberry PI 2 e um *tablet* (Galaxy Tab), com o intuito de investigar o comportamento do ambiente de grid computacional em casos de heterogeneidade extremas.

É importante ressaltar que os resultados de desempenho computacional obtidos neste estudo representam o ganho de desempenho obtido em ambientes de grid computacional compostos por computadores que possuam as mesmas configurações de *hardware* e *software* que os computadores utilizados nestes estudo e, por isso, não representa o ganho de desempenho que pode-se obter em todos os ambientes de grid computacional. Um exemplo disto é que o desempenho de um ambiente de grid computacional homogêneo composto por oito computadores que utilizam processadores que não sejam o Intel Xeon pode ser diferente do ganho de desempenho observado neste estudo, o qual utilizou oito computadores que utilizam os processadores Intel Xeon.

Antes do início de cada cálculo de *docking*, observou-se um atraso (*delay*) típico de aproximadamente 10 segundos entre o comando de envio de cada cálculo (*task*) na interface do programa GriDoMol, e o início efetivo da realização do mesmo nos *Workers*. Este *delay* é típico da arquitetura de *grid* computacional e é normalmente proveniente das comunicações entre os componentes do *grid* computacional. Estas comunicações incluem verificações de disponibilidade de máquinas, ou seja, se elas ainda estão ligadas e

conectadas ao ambiente de *grid* computacional, e verificação da presença dos requisitos mínimos necessários para a realização da tarefa, na máquina escolhida pelo *grid* computacional. Por conta deste *delay* devido às comunicações e transferência de arquivos no ambiente de *grid* computacional, os cálculos realizados em um cenário com apenas um computador no ambiente de *grid* computacional seriam mais lentos do que se fossem realizados por este mesmo computador fora do ambiente de *grid* computacional, utilizando-se, por exemplo, rotinas computacionais como *shell scripts* ou arquivos de lote (*batch*). Contudo, como o propósito central do ambiente de *grid* computacional é a utilização de processamento distribuído em varias máquinas, esta situação de execução em apenas um computador integrando o ambiente de *grid* computacional é bastante improvável. Vale ressaltar que mesmo que o tempo de execução do *job* (conjunto de *tasks*) no *grid* computacional operando com apenas um computador fosse igual à utilização de *shell scripts* ou arquivos de lote para executar todas estas tarefas (cálculos individuais de *docking*), ainda assim a interface gráfica do GriDoMol e o gerenciamento sequencial das tarefas individuais facilitaria bastante para o usuário que teria que executar toda a tarefa, sem a necessidade de nenhum recurso adicional (*shellscripts* ou arquivos de lote).

5.3.1 RESULTADOS DOS TESTES DE DESEMPENHO COMPUTACIONAL (GRIDOMOL 2.0)

A execução deste primeiro conjunto de cálculos de docking molecular foi realizado com a versão 2.0 do programa GriDoMol e o total de cálculos realizados nesta etapa pode ser visto na Tabela 7.

Tabela 7 – Total de cálculos realizados com a versão 2.0 do programa GriDoMol.

Teste	*# de jobs*	Cálculos individuais de *docking*	Tempo total (wallclock) em segundos
Grid homogêneo (exhaustiveness 8 e 32)	**128**	**27.264**	**3.160.905**
Grid homogêneo (exhaustiveness 1, 2, 4, 16 e 64)	**20**	**4.260**	**1.059.637**
Grid homogêneo ordenado (exhaustiveness 8)	**1**	**213**	**11.026**
Grid heterogêneo (exhaustiveness 8)	**2**	**426**	**25.409**
Total	**151**	**32.163**	**4.256.977**

Os testes de desempenho computacional realizados no ambiente de grid computacional homogêneo foram feitos utilizando-se duas configurações diferentes para os cálculos com o programa AutoDock Vina: o cálculo padrão (*exhaustiveness* = 8) e um cálculo mais preciso (*exhaustiveness* = 32). O tempo necessário para a realização de *docking* molecular no conjunto de 213 complexos (receptor + ligante), no ambiente de grid computacional homogêneo, pode ser observado na Tabela 8 (EX8) e na Tabela 9 (EX32), considerando as diferentes combinações de computadores e núcleos de processamento utilizados durante o conjunto de cálculos. Observa-se que, em ambos os casos, o tempo necessário para a realização do conjunto de cálculos (*job*) diminui conforme adiciona-se computadores e núcleos de processamento ativos em cada *job*.

Tabela 8 – Tempo necessário, em segundos, para realizar o *docking* molecular no conjunto de 213 complexos (receptor + ligante) no ambiente de *grid* computacional com o programa AutoDock Vina (***exhaustiveness* = 8**).

	# CPUS							
# Maq	1	2	3	4	5	6	7	8
1	71656	38243	29472	21483	21557	21245	20980	13489
2	37618	20236	16454	11431	11230	11185	11008	6901
3	29741	14025	10811	7767	7735	7607	7798	4724
4	21600	12747	8544	6140	6089	6162	6010	3236
5	18136	9438	7569	5165	5696	5186	5360	3079
6	16918	9077	6324	4614	4511	4490	4420	2642
7	14746	8375	5728	4634	4058	4065	4029	2407
8	13092	6882	5287	3822	3974	3603	3558	2057

Tabela 9 – Tempo necessário, em segundos, para realizar o *docking* molecular com o conjunto de 213 complexos (receptor + ligante) no ambiente de *grid* computacional com o programa AutoDock Vina (***exhaustiveness* = 32**).

	# CPUS							
# Maq	1	2	3	4	5	6	7	8
1	259440	139617	96400	72507	63349	55102	47587	41033
2	144373	72915	51631	38172	34489	30844	25534	22050
3	101918	51877	35967	26924	23025	21674	17862	15114
4	82679	41454	29656	21417	18875	16384	13951	12090
5	69408	35659	28211	19140	18880	13876	13864	9959
6	61138	35305	25264	18296	15424	12859	10453	8732
7	55350	29580	19712	14797	14254	12502	9355	7922
8	50797	26078	19556	13290	11579	10052	8609	7228

Os valores obtidos na métrica Fator de Desempenho por Núcleos (*FDN*), definido no capítulo de metodologia, podem ser observados na Tabela 10, para os cálculos com o parâmetro *exhaustiveness* = 8, e na Tabela 11, para os cálculos com o parâmetro exhaustiveness = 32.

Tabela 10 – Valores obtidos na métrica Fator de Desempenho por Núcleos no ambiente de *grid* computacional com o programa AutoDock Vina (***exhaustiveness* = 8**).

	# CPUS							
# Maq	1	2	3	4	5	6	7	8
1	1	1,87	2,43	3,34	3,32	3,37	3,42	5,31
2	1	1,86	2,29	3,29	3,35	3,36	3,42	5,45
3	1	2,12	2,75	3,83	3,84	3,91	3,81	6,30
4	1	1,69	2,53	3,52	3,55	3,51	3,59	6,67
5	1	1,92	2,40	3,51	3,18	3,50	3,38	5,89
6	1	1,86	2,68	3,67	3,75	3,77	3,83	6,40
7	1	1,76	2,57	3,18	3,63	3,63	3,66	6,13
8	1	1,90	2,48	3,43	3,29	3,63	3,68	6,36
Previsto	1	2	3	4	5	6	7	8

Tabela 11 – Valores obtidos na métrica Fator de Desempenho por Núcleos no ambiente de *grid* computacional com o programa AutoDock Vina (***exhaustiveness* = 32**).

	# CPUS							
# Maq	1	2	3	4	5	6	7	8
1	1	1,86	2,69	3,58	4,10	4,71	5,45	6,32
2	1	1,98	2,80	3,78	4,19	4,68	5,65	6,55
3	1	1,96	2,83	3,79	4,43	4,70	5,71	6,74
4	1	1,99	2,79	3,86	4,38	5,05	5,93	6,84
5	1	1,95	2,46	3,63	3,68	5,00	5,00	6,97
6	1	1,73	2,42	3,34	3,96	4,75	5,85	7,00
7	1	1,87	2,81	3,74	3,88	4,43	5,92	6,99
8	1	1,95	2,60	3,82	4,39	5,05	5,90	7,03
Previsto	1	2	3	4	5	6	7	8

Na Figura 18 podem ser observados os gráficos de *FDN*, tanto para os cálculos com o *exhaustiveness* = 8 (Figura 18a), quanto com o *exhaustiveness* = 32 (Figura 18b), e seus respectivos valores previstos. Observou-se que, principalmente nos casos com o *exhaustiveness* = 8, os valores de *FDN* apresentaram-se aquém do valor esperado ou previsto para *FDN*, ficando mais evidente nos casos em que mais de quatro núcleos de processamento (CPUs) são utilizados. Enquanto isso, nos cálculos com *exhaustiveness* = 32, observou-se um escalonamento quase linear dos valores. Esta diferença entre os cálculos com *exhaustiveness* 8 e 32 ocorre devido ao fato de que, ao se aumentar o parâmetro *exhaustiveness* para 32 (quatro vezes maior que 8, o valor padrão do AutoDock Vina), também está sendo aumentada a quantidade de buscas pela melhor solução de *docking* e, consequentemente, também está sendo aumentado o tempo gasto durante cada cálculo realizado em cada computador, mantendo o mesmo tempo gasto com as outras operações fundamentais que precisam ser realizadas no *grid* computacional.

Estas operações fundamentais de um ambiente de grid computacional consistem, essencialmente, na etapa de transferência de arquivos e na etapa de comunicação entre os componentes do ambiente de *grid* computacional. A etapa de transferência de arquivos ocorre em dois momentos: no início de uma *task* (cálculo de *docking*), onde os arquivos necessários para a execução da *task* são transferidos do computador do usuário para o computador remoto (no caso do programa AutoDock Vina, a estrutura das moléculas em formato de arquivo PDBQT), e no final de uma *task*, onde os resultados da execução da *task* são enviados do computador remoto para o computador do usuário que requisitou a *task*. A etapa de comunicação ocorre durante todo o momento em que o ambiente de *grid* computacional está operando e ocorre com o intuito de: i) Verificar, constantemente, quais componentes do grid computacional estão em funcionamento; ii) Distribuir, de forma otimizada, as *tasks* de um *job* ao longo dos computadores disponíveis, levando em

consideração a disponibilidade de cada *Worker* (se está em uso ou ocioso), e os pré-requisitos para execução de cada *tasks*, como, por exemplo, o sistema operacional e a quantidade de memória RAM disponível nestes computadores; iii) Gerenciar e informar o estado atual de cada *task* de um *job*, como quais delas iniciaram, quais encontraram um erro e quais foram finalizadas.

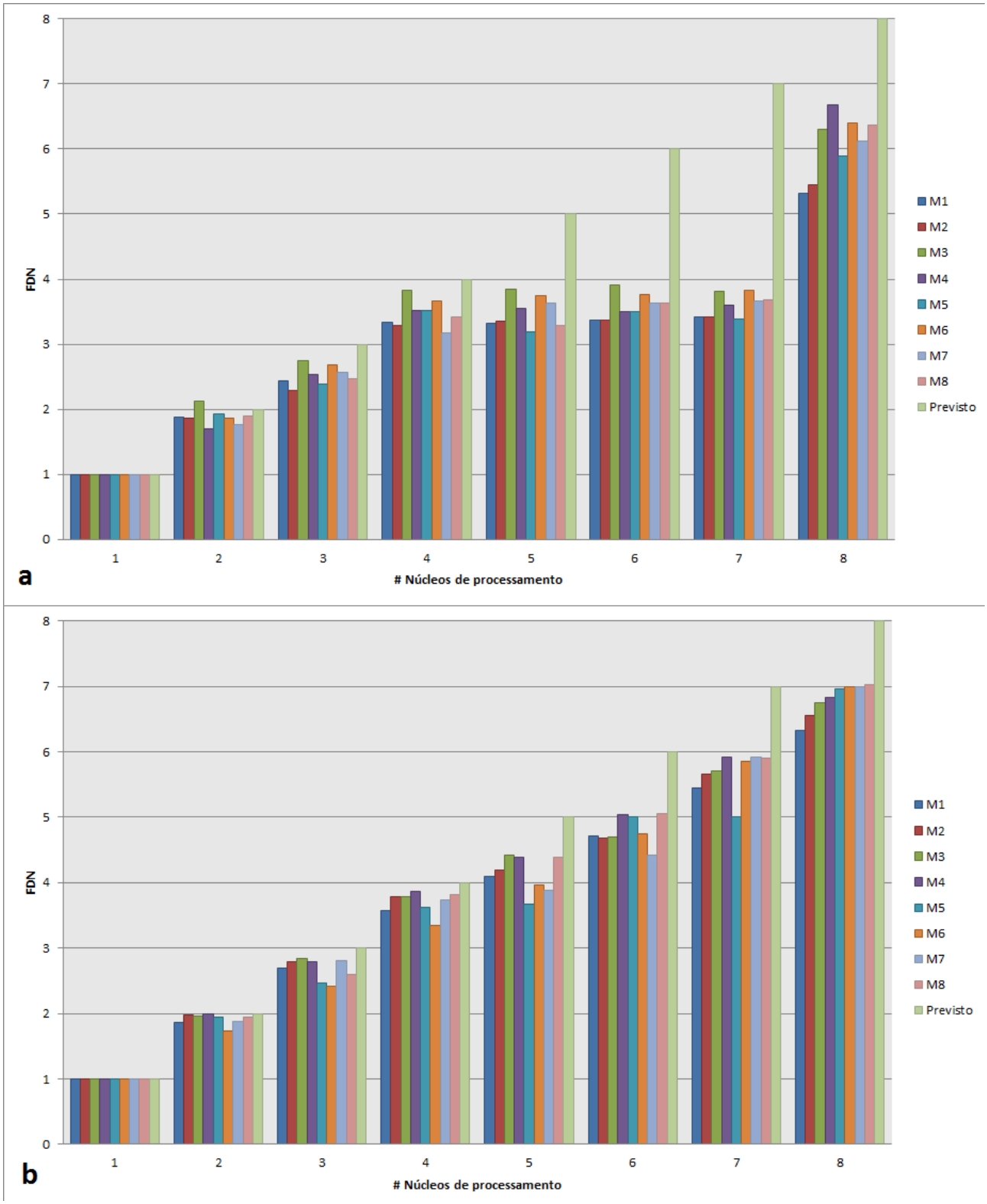


Figura 18 – Gráfico dos valores de *FDN* para o programa AutoDock Vina. A legenda no formato Mx (onde x se refere ao número de computadores utilizados). a) Cálculos com o valor 8 para o parâmetro *exhaustiveness* do programa AutoDock Vina. b) Cálculos com o valor 32 para o parâmetro *exhaustiveness* do programa AutoDock Vina.

Aumentar o número de núcleos de processamento utilizado para cada um dos cálculos de *docking* molecular não reduz o tempo gasto na etapa de transferência de arquivos, uma vez que, independente do número de núcleos de processamento utilizado, são sempre os mesmos arquivos sendo transferidos, e também não reduz o tempo gasto nas comunicações, já que, nesta métrica *FDN*, alterna-se apenas o número de núcleos de processamento utilizados por cada cálculo de *docking* mantendo sempre os mesmos computadores envolvidos na realização do conjunto de cálculos.

Sendo assim, como o aumento do número de núcleos de processamento reduz o tempo sequencial com processamento, mas tipicamente não reduz o tempo gasto nas operações básicas do ambiente de *grid* computacional é natural esperar que ao se aumentar a proporção do tempo gasto durante o processamento nos cálculos de *docking* propriamente ditos, e mantendo o mesmo tempo gasto em comunicações e transferências de arquivos, os benefícios de aplicar estratégias de paralelismo ao longo de vários núcleos de processamento (*multithreading*) se tornam mais evidentes nos cálculos mais robustos (*exhaustiveness* = 32), quando comparados com os cálculos realizados com a configuração padrão (*exhaustiveness* = 8).

Além disto, observa-se que, nos casos onde os cálculos são realizados com o valor 8 para o parâmetro *exhaustiveness* (Figura 18a), quando são utilizados 4, 5, 6 e 7 núcleos de processamento (CPUs), os valores de *FDN* foram praticamente os mesmos entre si, não havendo um ganho significativo de desempenho ao se aumentar os núcleos de processamento utilizados por cada computador de 4 para 5, 6 ou 7. O programa AutoDock Vina divide a área de busca pela melhor solução de *docking e* distribui o cálculo entre os núcleos de processamento do computador. O que pode ocorrer é que alguns destes núcleos de processamento podem estar ociosos a espera que o último núcleo de processamento ativo no computador termine suas buscas pela melhor solução de *docking*, através deste algoritmo interno de paralelização do programa AutoDock Vina.

Desta forma, esta diferença entre o desempenho previsto e observado pode estar relacionada com a limitação de ganho de desempenho na paralelização interna do programa AutoDock Vina, como observado em estudos anteriores (FERREIRA, 2013).

No gráfico que apresenta os valores de *FDN* para os cálculos que utilizam o valor 8 (padrão) para o parâmetro exhaustiveness do programa AutoDock Vina (Figura 18a), observa-se que o valor de *FDN* obtido ao realizar o conjunto de cálculos de docking molecular utilizando 2 núcleos de processamento e 3 computadores (*FDN* = 2,12) ultrapassou um pouco o valor previsto (previsto = 2,00). Como mencionado anteriormente, o ganho de desempenho ao aumentar o número de núcleos de processamento é uma característica intrínseca do programa AutoDock Vina e, pode ocorrer que o ganho de desempenho não seja completamente linear. Além disto, vale a pena ressaltar que o valor previsto é apenas o valor que se espera observar e não um limite teórico intransponível.

Os valores obtidos na métrica Fator de Desempenho por Máquinas (*FDM*), definido no capítulo de metodologia, podem ser observados na Tabela 12, para os cálculos com o parâmetro *exhaustiveness* 8, e Tabela 13 para os cálculos com o parâmetro *exhaustiveness* 32.

Tabela 12 - Valores obtidos na métrica Fator de Desempenho por Máquina no ambiente de *grid* computacional com o programa AutoDock Vina (***exhaustiveness* = 8**).

	# CPUS								
# Maq	1	2	3	4	5	6	7	8	Previsto
1	1	1	1	1	1	1	1	1	1
2	1,90	1,89	1,79	1,88	1,92	1,90	1,91	1,95	2
3	2,41	2,73	2,73	2,77	2,79	2,79	2,69	2,86	3
4	3,32	3,00	3,45	3,50	3,54	3,45	3,49	4,17	4
5	3,95	4,05	3,89	4,16	3,78	4,10	3,91	4,38	5
6	4,24	4,21	4,66	4,66	4,78	4,73	4,75	5,11	6
7	4,86	4,57	5,15	4,64	5,31	5,23	5,21	5,60	7
8	5,47	5,56	5,57	5,62	5,42	5,90	5,90	6,56	8

Tabela 13 - Valores obtidos na métrica Fator de Desempenho por Máquina no ambiente de *grid* computacional com o programa AutoDock Vina (***exhaustiveness* = 32**).

	# CPUS								
# Maq	1	2	3	4	5	6	7	8	Previsto
1	1	1	1	1	1	1	1	1	1
2	1,80	1,91	1,87	1,90	1,84	1,79	1,86	1,86	2
3	2,55	2,69	2,68	2,69	2,75	2,54	2,66	2,71	3
4	3,14	3,37	3,25	3,39	3,36	3,36	3,41	3,39	4
5	3,74	3,92	3,42	3,79	3,36	3,97	3,43	4,12	5
6	4,24	3,95	3,82	3,96	4,11	4,29	4,55	4,70	6
7	4,69	4,72	4,89	4,90	4,44	4,41	5,09	5,18	7
8	5,11	5,35	4,93	5,46	5,47	5,48	5,53	5,68	8

Os gráficos com os valores de *FDM* obtidos podem ser observados na Figura 19a, para os cálculos com a configuração padrão (*exhaustiveness* = 8), e Figura 19b para os cálculos com o parâmetro *exhaustiveness* = 32 do programa AutoDock Vina, e seus respectivos valores previstos. Observa-se um aumento no desempenho computacional conforme são adicionados mais máquinas disponíveis no grid computacional. Este ganho de desempenho se deve a simultaneidade do processamento, uma vez que, ao utilizar dois ou mais computadores, alguns cálculos de *docking* molecular ocorreram em paralelo, em máquinas diferentes, reduzindo-se assim o tempo total necessário para a realização do conjunto de cálculos, quando comparado com o tempo necessário para realizar o mesmo conjunto de cálculos utilizando-se apenas um computador.

Ainda no gráfico que apresenta os valores de *FDM* para os cálculos que utilizam o valor 8 (padrão) para o parâmetro exhaustiveness do programa AutoDock Vina (Figura 19a), observa-se que o valor de *FDM* obtido ao realizar o conjunto de cálculos de docking molecular utilizando 4 computadores e 8 núcleos de processamento (*FDM* = 4,17) ultrapassou um pouco o valor previsto (previsto = 4,00). É importante ressaltar que cada computador que integra o ambiente de grid computacional possui seu próprio sistema operacional e, portanto, parte dos núcleos de processamento destes computadores são utilizados, periodicamente, pelos processos do sistema com o intuito de manter o

computador operando normalmente. Ocorre que os cálculos de docking molecular, em especial os que utilizam todos os núcleos de processamento do computador (8 núcleos, neste estudo) deixam poucos recursos para o sistema operacional, fazendo com que os processos do sistema tenham que disputar recursos computacionais com estes cálculos de docking molecular, atrasando a execução destes cálculos. Acontece que este atraso tem um impacto maior nos casos onde se utiliza apenas um computador uma vez que estes cálculos são executados sequencialmente e, por isso, tendem a ser mais demorados quando comparados com a execução simultânea em paralelo em múltiplos computadores, podendo ocorrer mais processos do sistema operacional disputando recursos computacionais com os cálculos de docking. Sendo assim, levando em consideração que nesta métrica (*FDM*) utiliza-se como base para medir o ganho de desempenho os cálculos realizados com apenas um computador, pode ocorrer que alguns valores de *FDM* excedam o valor previsto, como foi o caso do cálculo com 4 computadores utilizando 8 núcleos de processamento, uma vez que o valor previsto é apenas o valor que se espera observar e não um limite teórico intransponível.

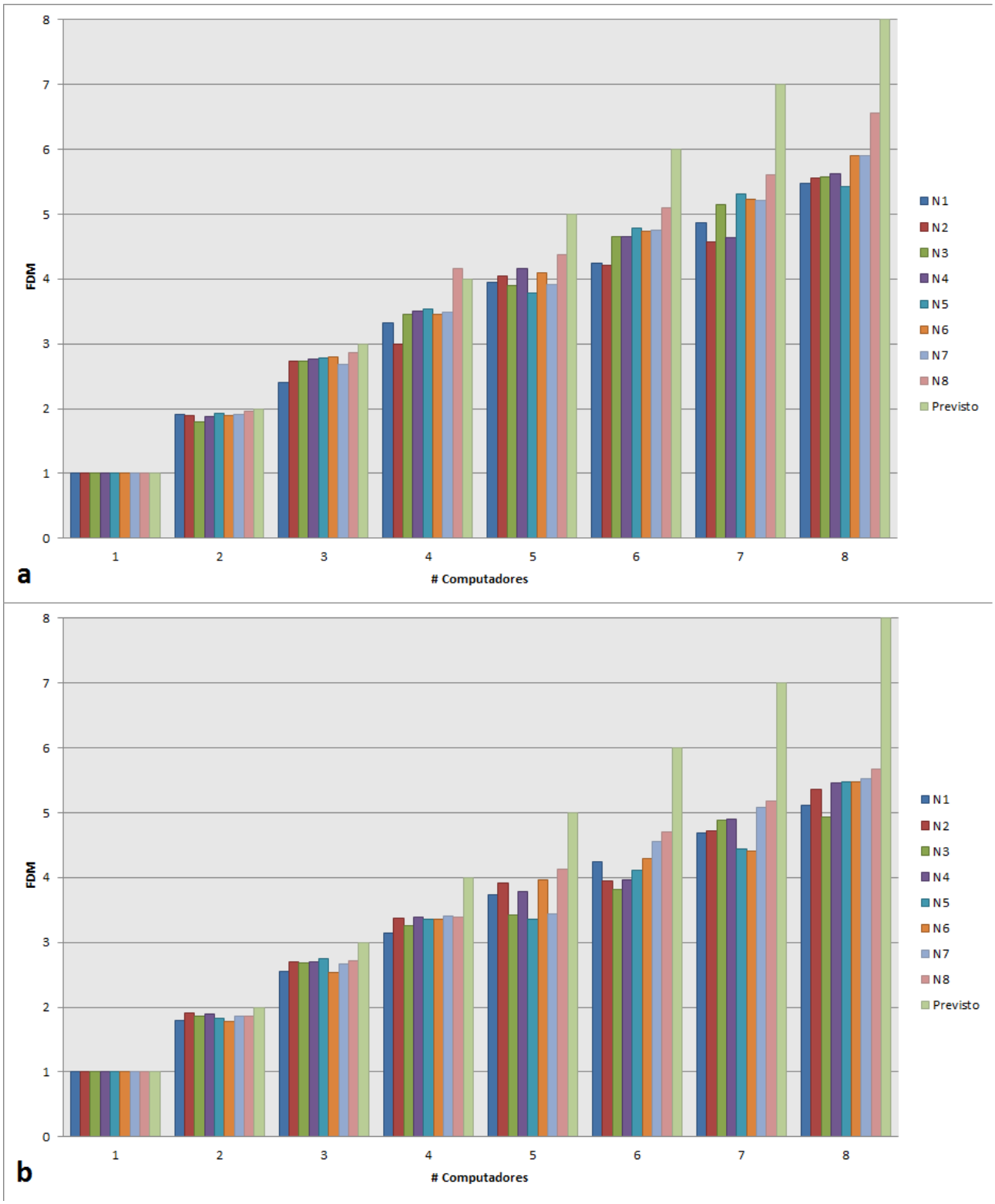


Figura 19 – Gráfico dos valores de *FDM* ao aumentar o número de computadores envolvidos em cada conjunto de cálculos. A legenda no formato Nx (onde x se refere ao número de núcleos de processamento utilizados por computador) se refere aos valores observados. a) Cálculos com o valor 8 para o parâmetro *exhaustiveness* do programa AutoDock Vina. b) Cálculos com o valor 32 para o parâmetro *exhaustiveness* do programa AutoDock Vina.

Também pode-se observar que ao se aproximar de 8 computadores, a diferença entre o *FDM* previsto e o observado aumenta. Durante a análise do tempo gasto por cada cálculo individual de *docking*, observou-se que o antepenúltimo cálculo (complexo receptor + ligante denominado 5ER2) representa o cálculo mais demorado dentre todos os complexos utilizados no conjunto de testes. O tempo necessário para executar o cálculo de *docking* molecular apenas para este complexo, considerando-se o *setup* padrão do AutoDock Vina (*exhaustiveness* = 8) é de 5.273 segundos, ao se utilizar apenas um dos oito núcleo de processamento, e 711 segundos, ao se utilizar todos os oito núcleos de processamento de uma máquina. Em contrapartida, o tempo para concluir este mesmo cálculo utilizando-se o valor 32 para o parâmetro *exhaustiveness* no programa AutoDock Vina é de 21.093 segundos, ao se utilizar apenas um dos oito núcleo de processamento, e 2.755 segundos, ao se utilizar todos os oito núcleos de processamento de uma máquina. A demanda computacional apresentada por cada um dos 213 complexos utilizados neste trabalho, na ordem em que foram realizados, para os *jobs* que utilizam apena um núcleo de processamento pode ser observada no gráfico da Figura 20. Observa-se, neste gráfico, que a demanda computacional apresentada pelo antepenúltimo cálculo de *docking* é muito mais elevada, quando comparada com a demanda computacional dos cálculos próximos a ele na fila de execução do *job.* Isto faz com que muitos dos computadores fiquem ociosos, aguardando apenas a finalização deste cálculo (*task*) no *job* que está em funcionamento no grid computacional, impactando, por vezes significativamente, no tempo total necessário para a realização do conjunto de cálculos (*job*), como um todo. É importante enfatizar que, em um caso real de *virtual screening*, não se sabe, a priori, quais os cálculos que são os mais demorados (maior demanda computacional) no conjunto de tarefas (*tasks*) e, portanto, estes resultados apresentados nesta seção refletem um cenário real de utilização cotidiana para um sistema desta natureza.

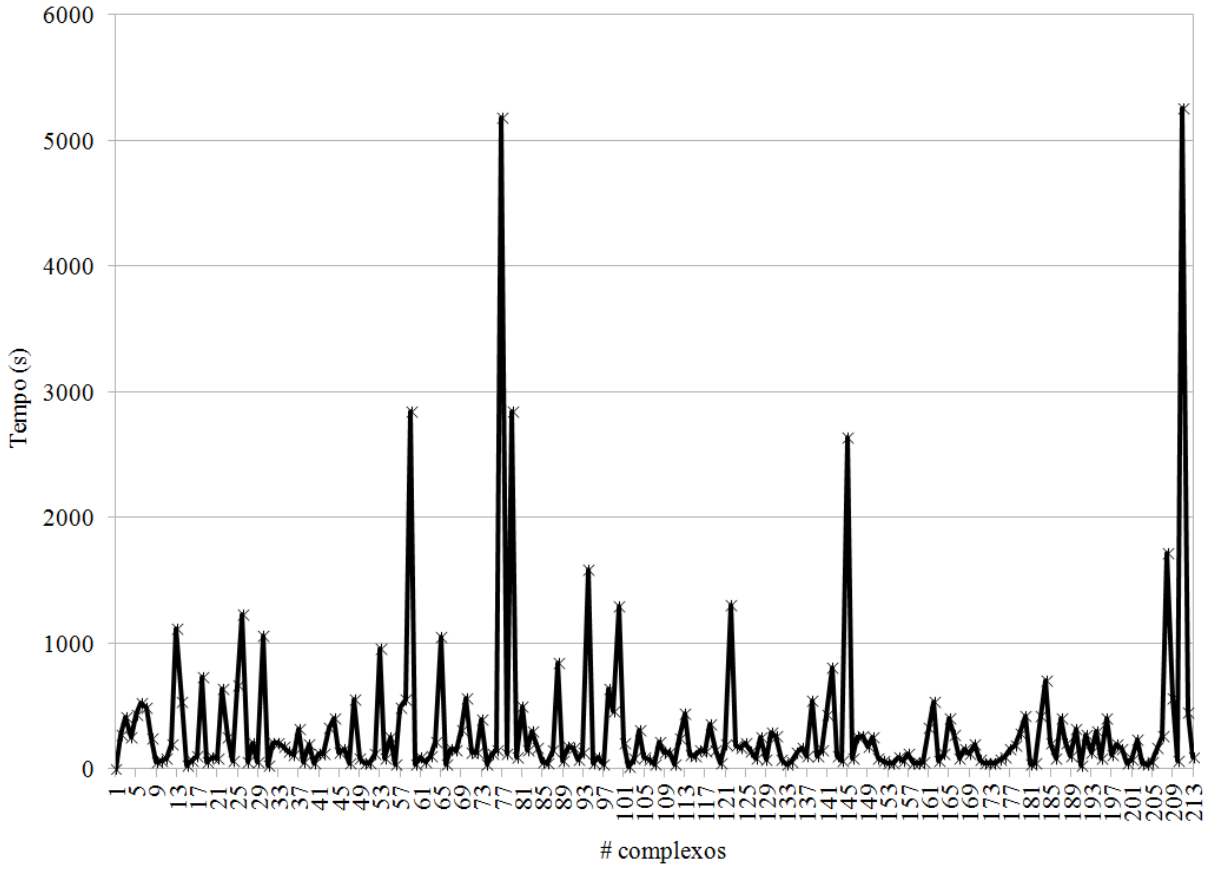


Figura 20 – Gráfico da demanda computacional apresentada por cada um dos 213 complexos utilizados neste estudo, para os cálculos com apenas um núcleo de processamento em uso.

Com o intuito de verificar o impacto da ociosidade durante os últimos cálculos de *docking* nos valores de *FDM*, realizou-se um teste adicional utilizando um *job* contendo os cálculos de docking molecular ordenados de forma decrescente, de acordo com sua demanda computacional observada, de forma que os cálculos mais demorados são realizados no início do *job* enquanto os cálculos mais rápidos são realizados no final do *job*. Como os cálculos com oito computadores e um núcleo de processamento ativo, em cada, foi um dos que apresentou uma diferença maior entre o valor de *FDM* observado e o previsto, escolheu-se realizar novamente este cálculo utilizando-se a sequência de tasks reordenada de forma decrescente (em demanda), ao invés do ordenamento original

em ordem alfabética.

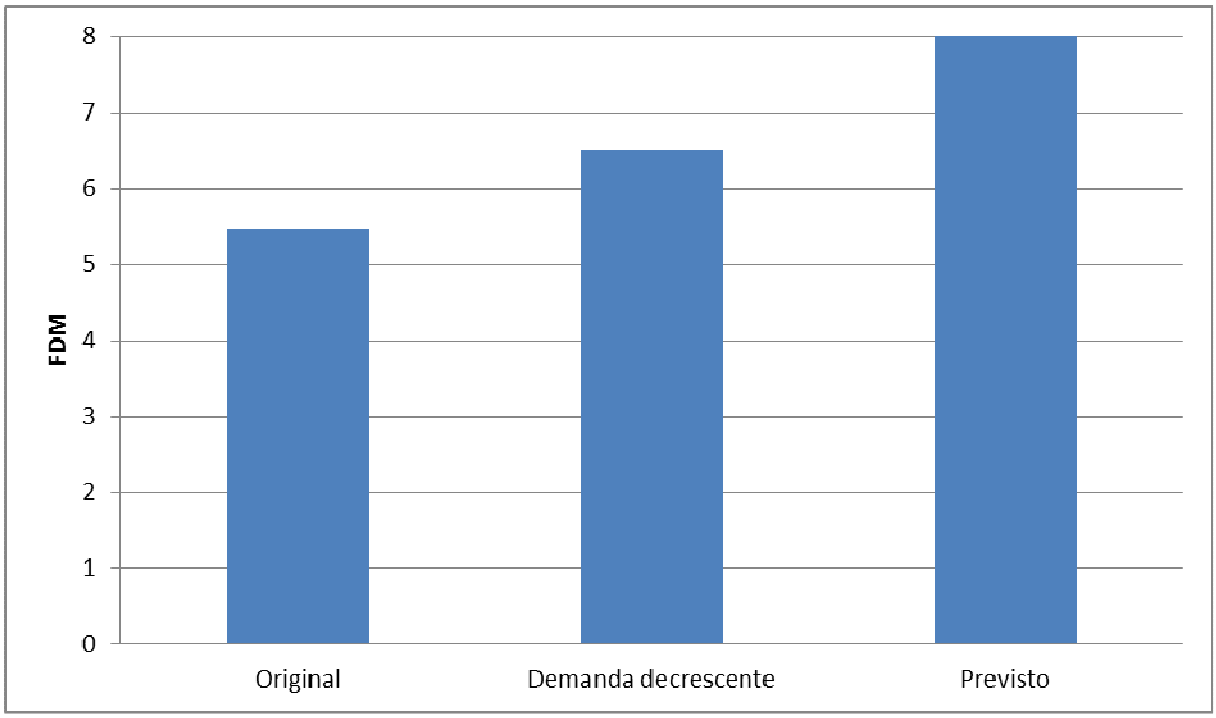


Figura 21 – Gráfico dos valores de FDM para os cálculos que utilizam oito computadores e apenas um núcleo de processamento em cada, considerando a sequência original das tasks (ordem alfabética) e a nova sequência reordenada de forma decrescente (em termos de demanda computacional).

Observa-se no gráfico de *FDM* para os cálculos com oito computadores e um núcleo de processamento (ver Figura 21) que ao se ordenar as tasks de forma decrescente para a demanda computacional, o valor de *FDM* subiu de 5,47 (ordenamento original) para 6,50, ficando mais próximo do valor previsto (8,0). Desta forma, pode-se afirmar que ao se ordenar o conjunto de cálculos, levando-se em consideração a demanda computacional individual de cada cálculo de *docking,* pode-se reduzir a ociosidade apresentada durante os últimos cálculos de *docking* e, consequentemente, obter um ganho de desempenho maior, quando comparado com o ordenamento original (que, para todos os efeitos, constitui um ordenamento aleatório do ponto de vista de

demanda computacional). Vale a pena ressaltar novamente que, como dito anteriormente, não há como saber de forma exata a demanda computacional dos cálculos de *docking* antes de realizá-los uma primeira vez.

Ao se utilizar ambas as formas de distribuição (ou paralelização) para a realização dos cálculos de *docking* molecular com alta performance, o ambiente de *grid* computacional homogêneo e a paralelização *multithreading* implementada internamente pelos autores do programa AutoDock Vina, pode ser obtido um ganho total de desempenho de 34,84 (para *exhaustiveness* = 8) e 35,89 (para *exhaustiveness* = 32) vezes mais rápidos no ambiente de *grid* computacional acessado pelo GriDoMol, quando comparado com a execução do mesmo conjunto de cálculos em apenas um computador e utilizando-se apenas um núcleo de processamento (CPU) ativo.

Na tabela 16, é possível observar a quantidade de cálculos e os tempos gastos por cada um dos dispositivos utilizados para formar o ambiente de grid computacional heterogêneo, sendo eles dois *smartphones* (*Nexus* 4 e *Galaxy* S4), um *tablet* (*Galaxy Tab*), duas máquinas *Pentium* IV e uma máquina *Xeon*, dos 213 cálculos de *docking* molecular. Vale ressaltar que o total apresentado na tabela (123.354 segundos) refere-se ao tempo total que seria gasto se estes cálculos fossem realizados sequencialmente (de maneira não simultânea ou paralela), e que o tempo real (*wallclock*) foi de 11.240 segundos, cerca de 10,97% do tempo que seria gasto de maneira sequencial. Também é importante enfatizar que os núcleos que realizaram mais de 10.000 segundos de processamento (núcleo #2 do *Nexus* 4, núcleo #7 do *Xeon* e o *Pentium* IV #2) estiveram envolvidos nos últimos três cálculos mais demorados e que, na ausência destes cálculos, o tempo total de execução destes núcleos cairia provavelmente para valores abaixo de 7.000 segundos, como a maioria dos outros processadores, como pode ser observado na Tabela 14.

Tabela 14 – Quantidade de cálculos (*tasks*) executados por cada núcleo de cada dispositivo utilizado nos testes do ambiente de *grid* computacional heterogêneo.

Worker	Núcleo #	Tempo	Cálculos	Worker	Núcleo #	Tempo	Cálculos
Nexus4	1	6749	5	**Xeon**	3	6047	25
Nexus4	2	11175	7	**Xeon**	4	6189	16
Nexus4	3	6810	5	**Xeon**	5	5978	18
Galaxy S3	1	7862	1	**Xeon**	6	6047	14
Galaxy S3	2	6052	2	**Xeon**	7	10885	29
Galaxy S3	3	6361	5	**Xeon**	8	6517	31
GalaxyTab	1	6447	1	**Pentium IV (1)**	1	6280	12
Xeon	1	6482	23	**Pentium IV (2)**	1	10648	10
Xeon	2	6825	9	**Total**	**51**	**123354**	**213**

Pode-se observar na Tabela 14 que os 2 computadores que possuem processadores *Pentium* IV executaram quase a mesma quantidade de cálculos (12 e 10 cálculos, respectivamente) do que alguns dos núcleos de processamento do computador *Xeon* (a máquina mais potente), como, por exemplo, os núcleos 2 e 6 (9 e 14 cálculos, respectivamente). Isto se deve ao fato de que estes processadores *Pentium* IV, embora antigos, possuem uma capacidade computacional elevada (*clock* de 3GHz), tornando-os quase tão rápidos quanto um único núcleo de processamento encontrado nos processadores *multicore* modernos. Estes dois computadores *Pentium* IV executaram, juntos, 10,3% do conjunto dos 213 cálculos de *docking*. Vale ressaltar que, por causa da idade destes computadores *Pentium* IV, eles estavam desligados e sem uso (ociosos) no laboratório, antes de serem integrados ao *grid* computacional heterogêneo.

Ainda na Tabela 14, observa-se que os *smartphones* e o *tablet* executaram, juntos, 26 dos 213 cálculos de *docking* molecular. Os processadores inclusos em *smartphones* e *tablets* modernos possuem vários núcleos de processamento, muitos deles com capacidade de processamento (*clock*) superior a 1.2 GHz, sendo que na maior parte do tempo estes núcleos não estão sendo utilizados (estão ociosos) pelo próprio sistema operacional Android. Sendo assim, ao deixar um núcleo de processamento livre para realizar as operações básicas do sistema operacional Android, estes *smartphones* e

tablets conseguiram contribuir com seus núcleos de processamento ociosos para a realização de 12,2% do total de cálculos de *docking*, sem comprometer o funcionamento dos aparelhos, inclusive recebendo ligações telefônicas durante os cálculos, como já foi dito antes.

Outra informação que pode ser observada na tabela 14 é que o computador que possui os processadores *Xeon* realizou a maior parte dos cálculos, totalizando 165 cálculos concluídos, que representa 77,5% do total de cálculos. Este resultado não surpreendeu, uma vez que este computador possui oito núcleos de processamento, sendo cada núcleo deste processador mais rápido do que os demais núcleos de processamento presente nos dispositivos utilizados no ambiente de *grid* computacional heterogêneo. Para medir o ganho de desempenho ao se utilizar um ambiente de *grid* computacional heterogêneo, em comparação com um único dispositivo, foi realizado um teste com o mesmo conjunto de cálculos (213 complexos), sendo executados apenas no computador Xeon. Observou-se que, ao se aplicar a estratégia de paralelismo com um ambiente de *grid* computacional heterogêneo (tempo total de 11.240 segundos), foi possível reduzir em aproximadamente 21% (2.929 segundos a menos) o tempo necessário para a realização do mesmo conjunto de cálculos em um único computador (tempo total de 14.169 segundos). Vale ressaltar, novamente, que os computadores Pentium IV, *smartphones* (*Nexus* 4 e *Galaxy* S3) e o *tablet* (*Galaxy Tab* 2) adicionados ao ambiente de *grid* computacional heterogêneo estavam ociosos e que suas respectivas capacidades computacionais estavam sendo desperdiçadas, por assim dizer.

5.3.2 RESULTADOS DOS TESTES DE DESEMPENHO COMPUTACIONAL (GRIDOMOL 3.0)

Os testes de desempenho computacional com a versão 3.0 do programa GriDoMol foram realizados utilizando os programas de docking molecular AutoDock e o AutoDock

Vina, utilizando diferentes configurações de docking. Para os cálculos de docking com o programa AutoDock Vina, foram utilizadas as configurações de cálculo padrão (*exhaustiveness* = 8) e um cálculo mais preciso (*exhaustiveness* = 32). Para os cálculos de docking com o programa AutoDock, utilizou-se as configurações de cálculo com os valores padrão (P) para os parâmetros de seu algoritmo genético e um cálculo avançado (A) utilizando valores modificados para os parâmetros do algoritmo genético correspondendo a configuração que obteve o melhor custo beneficio (ga_num_evals = 25.000.000, ga_num_generations = 10.000 e ga_run = 25) identificado durante testes preliminares e descrito detalhadamente na seção 5.5 do capítulo de metodologia.

O número de computadores utilizados variou entre 1, 2, 4, 6 e 8. Para os cálculos com o programa AutoDock Vina, foram utilizados todos os oito núcleos de processamento de cada computador. Para os cálculos com o programa AutoDock, como este programa não possui paralelismo interno em nível de CPUs e com o intuito de utilizar o potencial de todos núcleos de processamento destes computadores, optou-se por enviar uma instância de execução por cada Worker disponível no computador (sendo 8, para cada computador), uma vez que cada um dos computadores do ambiente de grid computacional possuem a mesma quantidade de componentes Workers e núcleos de processamento.

Tabela 15 – Tempos de execução do conjunto de cálculos de docking molecular, realizados com a nova versão do programa GriDoMol.

	Tempo (s)			
# Máq	AutoDock (P)	AutoDock (A)	Vina (EX8)	Vina (EX32)
1	17211	285464	9611	26932
2	10373	182955	4897	14093
4	6244	119561	2616	7803
6	4932	103161	1887	5989
8	4493	91826	1476	4754

Na Tabela 15, observa-se o tempo necessário para a realização do conjunto de 213 cálculos de docking molecular no ambiente de grid computacional homogêneo, ao alternar os programas de docking molecular e suas respectivas configurações. Ao adicionar mais computadores ao ambiente de grid computacional, reduz-se o tempo necessário para a realização do conjunto de cálculos.

Tabela 16 – Valores de *FDM* obtidos ao executar o conjunto de cálculos de docking molecular com a versão 3.0 com programa GriDoMol.

# Máq	AutoDock (P)	AutoDock (A)	Vina (EX8)	Vina (EX32)	Previsto
1	1	1	1	1	1
2	1,66	1,56	1,96	1,91	2
4	2,76	2,39	3,67	3,45	4
6	3,49	2,77	5,09	4,50	6
8	3,83	3,11	6,51	5,67	8

Os valores obtidos na métrica Fator de Desempenho por Máquinas (*FDM*) pode ser observado na Tabela 16. Assim como o ocorrido durante os testes realizados com a versão 2.0 do programa GriDoMol, os valores de *FDM* ficaram bastante abaixo do previsto (ver Figura 22). Os valores de *FDM* obtidos pelo programa AutoDock foram mais baixos, não conseguindo atingir, por exemplo, nem a metade do *FDM* previsto para oito computadores. Isto ocorre porque os cálculos com este programa são mais demorados quando comparados com os cálculos com o programa AutoDock Vina e a execução do antepenúltimo cálculo (estrutura PDB: 5ER2) com o programa AutoDock demorou mais de 20 horas para ser finalizado e, possivelmente, houve um período de ociosidade em que os computadores restantes ficaram parados esperando a conclusão deste último e demorado cálculo de docking molecular.

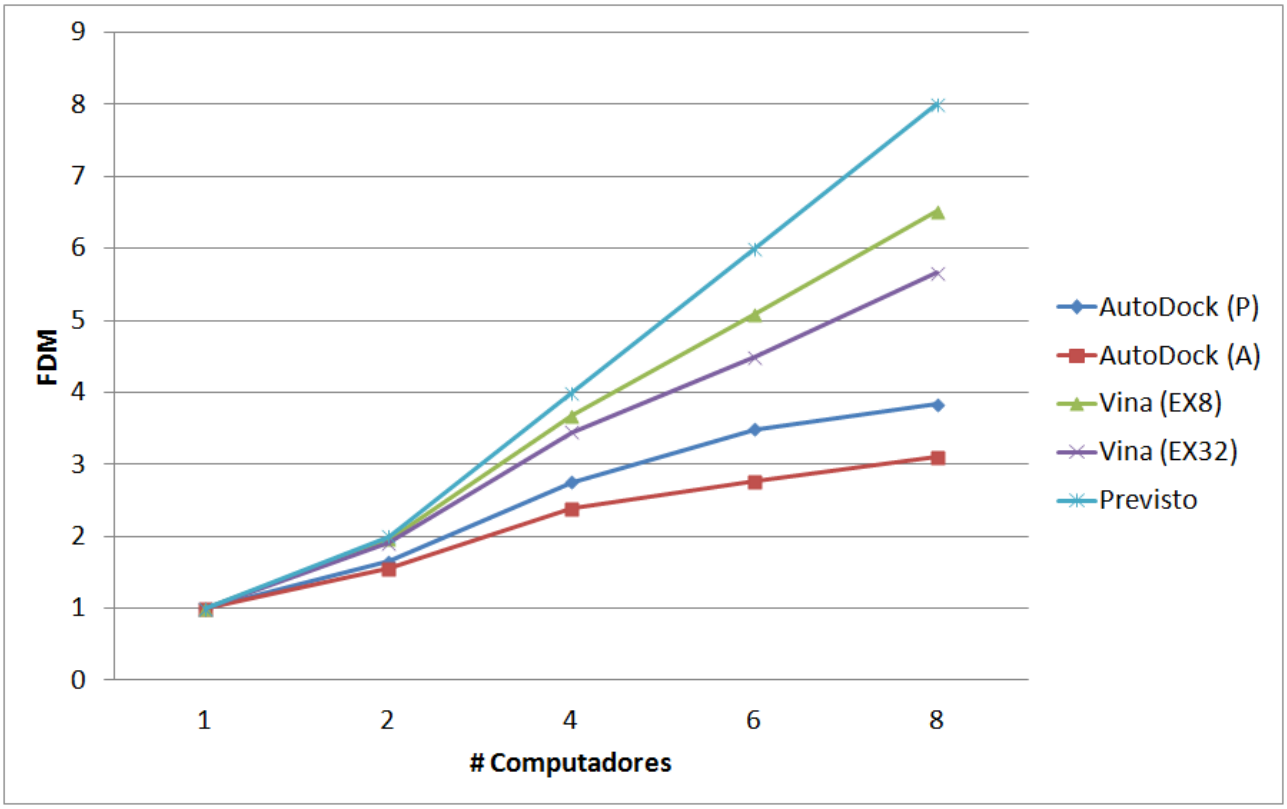


Figura 22 – Gráfico com os valores de *FDM* obtidos ao realizar o conjunto de cálculos de docking molecular com os programas AutoDock e AutoDock Vina, utilizando as configurações padrão (P e EX8) e avançadas (A e EX32).

Com o objetivo de verificar se, assim como constatado durante os testes com a versão 2.0 do programa GriDoMol, a ordem em que os cálculos de docking molecular são realizados possui um impacto significativo no desempenho do ambiente de grid computacional, novos cálculos utilizando o ordenamento por demanda computacional decrescente foram realizados. Observou-se que o desempenho do ambiente de grid computacional aumentou ao realizar os cálculos de docking molecular utilizando este ordenamento decrescente em termos de demanda computacional, fazendo com que os valores de *FDM* aumentassem, se aproximando do valor previsto de *FDM*.

Sendo assim, foi identificada a necessidade de elaboração de uma função interna no programa GriDoMol, em sua versão 3.0, com o intuito de tentar prever, mesmo que de forma aproximada, a demanda computacional de um cálculo de *docking* molecular e,

assim, ordenar automaticamente por demanda computacional decrescente, otimizando o desempenho do ambiente de grid computacional. Embora não haja como prever com precisão a demanda computacional de cada cálculo de *docking* molecular antes de realizá-lo de fato, existem algumas características em um cálculo de *docking* molecular que podem influenciar diretamente a demanda computacional. A função interna de ordenamento por demanda computacional implementada no programa GriDoMol leva em consideração o tamanho da região de busca em torno do sítio ativo do alvo biológico (multiplicando os valores de size x, y e z) e, como critério de desempate, o número de graus de liberdade (torções) que o ligante possui.

Novos testes foram realizados com o objetivo de comparar o desempenho do ambiente de grid computacional ao realizar o conjunto de cálculos de docking molecular utilizando diferentes ordenamentos (ver Figura 23): i) O ordenamento decrescente baseado na demanda computacional observada, após a conclusão de todos os cálculos de docking molecular (Perfeito); ii) O ordenamento decrescente baseado na demanda computacional predita, estimando a demanda computacional baseado no tamanho da região de busca e no número de torsões do ligante (GriDoMol); iii) O ordenamento utilizado inicialmente, aleatório em termos de demanda computacional (ordem alfabética). Estes testes foram realizados utilizando 6 computadores no grid e, como no teste anterior, utilizando todos os 8 núcleos de processamento destes 6 computadores. Ao se executar o conjunto de cálculos de docking molecular utilizando o ordenamento realizado pelo programa GriDoMol, observou-se uma aproximação com os valores de *FDM* obtidos ao utilizar o ordenamento decrescente “exato”. Vale ressaltar que, como o ordenamento decrescente exato baseado na demanda computacional observada ao termino de cada cálculo não é viável, uma vez que não faria sentido realizar o cálculo de docking molecular novamente após ter sido finalizado na primeira vez, o ordenamento predito realizado pelo programa GriDoMol mostrou-se bastante satisfatório e viável, aumentando

consideravelmente o desempenho do ambiente de grid computacional, diminuindo a ociosidade durante os cálculos finais de docking molecular ao posicionar os cálculos com a menor demanda computacional aparente no final da lista de cálculos.

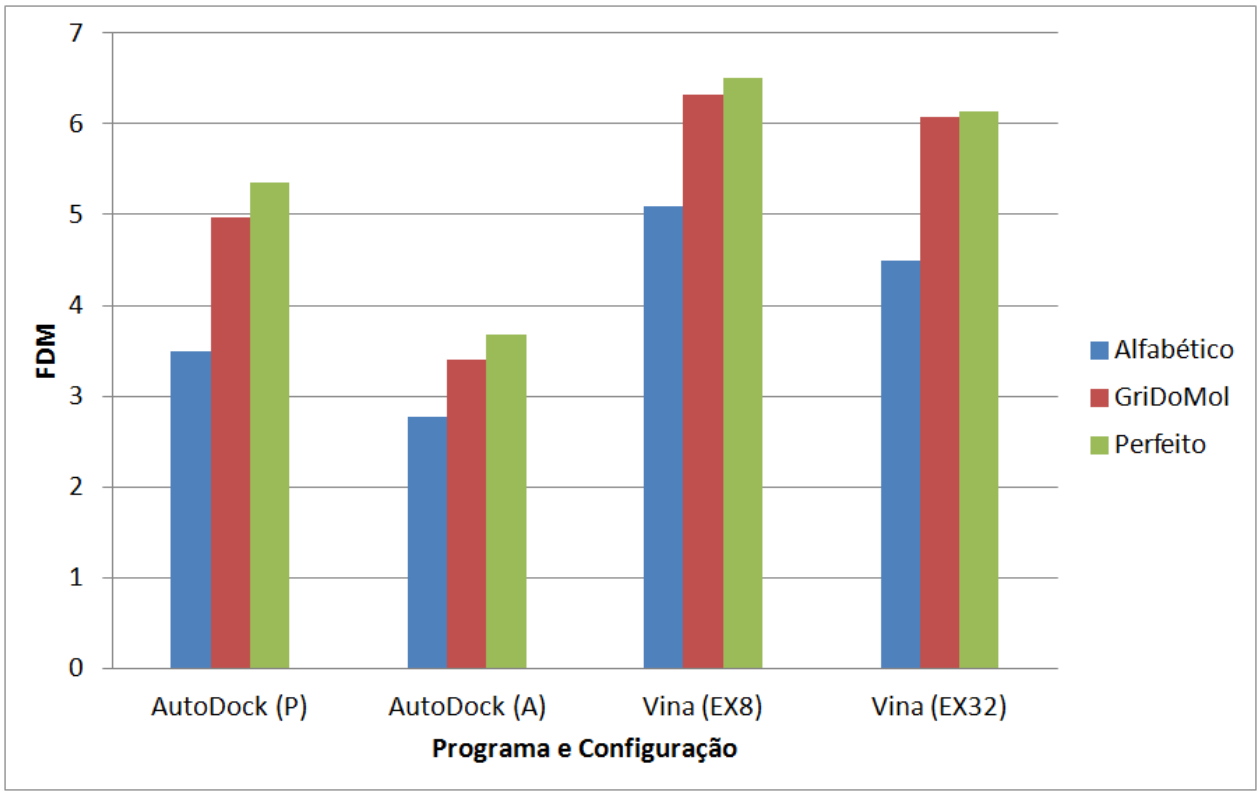


Figura 23 - Gráfico com os valores de *FDM* obtidos ao mudar a ordem de execução do conjunto de cálculos de docking molecular para cada programa de docking molecular em diferentes configurações de docking utilizando seis computadores.

Observa-se também, ainda na Figura 23, que os valores de *FDM* para o ordenamento realizado pelo programa GriDoMol e o ordenamento perfeito, ambos utilizando o programa AutoDock Vina, ultrapassaram o valor de *FDM* previsto para este caso onde seis computadores foram utilizados e, portanto, era previsto que o valor de *FDM* fosse igual a seis. Porém, como mencionado anteriormente, este valor previsto não representa um limite teórico e sim um valor esperado e pode ultrapassar o valor previsto nestas condições onde o conjunto de cálculos de docking molecular realizados utilizando

apenas um computador e todos os núcleos de processamento possui maiores chances de haver processos do sistema operacional disputando recursos com os cálculos de docking, retardando a execução deste conjunto de cálculos que é usado como base para calcular os valores de *FDM*.

Um novo teste utilizando dispositivos heterogêneos foi realizado nesta versão 3.0 do programa GriDoMol em um ambiente de grid computacional formado por um smartphone (*Nexus* 4), um mini computador (*Raspberry PI* 2) e um dos computadores utilizados também no teste homogêneo (Xeon). Como o cálculo de docking molecular para cada um dos 213 complexos utilizados neste estudo possuem uma demanda computacional distinta, dificultando a capacidade de avaliar o número de cálculos que cada um destes dispositivos heterogêneos conseguem executar, optou-se por selecionar o cálculo de docking molecular com a menor demanda computacional (estrutura PDB: 2J77) e criar um job contendo 100 cópias deste cálculo para ser submetido ao ambiente de grid computacional heterogêneo e executado pelo programa de docking molecular AutoDock Vina. Além disto, com o intuito de avaliar a capacidade por núcleo, e com a finalidade de liberar um dos núcleos para o sistema operacional de cada aparelho (*Raspi* e *Nexus*) poder funcionar livremente, permitiu-se a utilização de apenas 3 núcleos de processamento para cada um destes dispositivos.

Tabela 17 – Número de cálculos de docking molecular realizados por cada um dos dispositivos heterogêneos.

Núcleo	Tasks	Núcleo	Tasks	Núcleo	Tasks
Xeon_1	22	Raspi_1	4	Nexus_1	7
Xeon_2	22	Raspi_2	5	Nexus_2	7
Xeon_3	22	Raspi_3	4	Nexus_3	7

O número de cálculos de *docking* molecular realizados por cada um dos núcleos de processamento dos dispositivos heterogêneos pode ser observado na Tabela 17. Observa-se que o desempenho de cada um dos núcleos de processamento de um mesmo dispositivo em realizar cálculos de *docking* molecular foi praticamente o mesmo, com a exceção apenas do núcleo #2 do dispositivo *Raspberry PI*, que realizou 1 cálculo a mais que o núcleo #1 e #3 deste mesmo dispositivo. Ainda nesta tabela, observa-se que o desempenho obtido por um núcleo do *smartphone Nexus* 4 (7 cálculos) equivale a quase dois núcleos do computador *Raspberry PI* (4 cálculos). Os núcleos de processamento do computador *Xeon* foi o que obteve melhor desempenho (22 cálculos, cada núcleo), totalizando 66 dos 100 cálculos de *docking* molecular submetidos. Todos os 3 núcleos de processamento utilizados pelo *smartphone Nexus* 4 realizaram quase o mesmo número de cálculos (21 cálculos) do que um único núcleo do computador *Xeon*. Vale ressaltar que este desempenho superior apresentado pelos núcleos do computador *Xeon*, quando comparado com os núcleos do *Raspberry PI* e do *Nexus* 4, se deve a fatores como a diferença de arquiteturas entre estes dispositivos, enquanto o computador *Xeon* possui a arquitetura x86 de 64 bits os outros dispositivos são da arquitetura ARM de 32 bits, além de diferenças na velocidade de processamento destes núcleos, onde cada núcleo de processamento do computador *Xeon* possui 2,33 GHz contra 0,9 GHz do núcleo do *Raspberry* PI 2 e 1,5 GHz do núcleo do *Nexus* 4.

Considerando que os *smartphones* atuais possuem vários núcleos de processamento e que não é sempre que estes *smartphones* estão utilizando todo seu poder computacional, este ganho de desempenho ao adicionar os *smartphones* ao grid computacional com o objetivo de acelerar a execução de um conjunto de cálculos de *docking* molecular é significativo, uma vez que este poder computacional ocioso seria desperdiçado de outra forma.

Uma das principais vantagens da utilização de computadores *Raspberry PI* em um grid computacional está na sua relação custo/benefício. Embora o desempenho de um núcleo do computador *Raspberry PI* em realizar cálculos de *docking* molecular seja cerca de 5,5 vezes menor do que o desempenho de um núcleo do computador *Xeon*, um computador *Raspberry PI* 2 custa muito menos do que um computador *Xeon*. Baseando-se nos resultados obtidos neste teste com o ambiente de grid computacional heterogêneo acredita-se ser viável a utilização de vários computadores *Raspberry PI* interligados com o intuito de formar um ambiente de grid computacional de baixo custo com um bom desempenho computacional integrado.

5.4 Comparações com os Resultados Experimentais

Pode-se observar na Tabela 20 os valores de energia livre de ligação (ou *binding), em* kcal/mol, obtidos experimentalmente (EXP), e os valores de *binding* obtidos através das melhores soluções de *docking* molecular encontradas pelo programa AutoDock Vina, ao alternar o valor do parâmetro *exhaustiveness* utilizado (EX1, EX2, EX4, EX8, EX16, EX32, EX64), para cada complexo (receptor + ligante) calculado. É possível observar que, em termos gerais, os valores médios de *binding* diminuem gradativamente conforme é aumentado o valor do parâmetro *exhaustiveness*, variando de -7,73 (*exhautiveness* = 1) até -8,15 (*exhaustiveness* = 64). É importante ressaltar que esta redução média de 0,42 kcal/mol nos valores de energia livre de ligação ao longo dos 213 complexos é significativo, uma vez que, ao se aumentar o número de buscar por uma conformação mais estável (através do parâmetro exhaustiveness), as chances de que um ligante seja descartado prematuramente como um falso negativo por ter encontrado uma solução de docking inicial menos estável é reduzida. Desta forma, quanto mais alto for o valor do parâmetro *exhaustiveness* utilizado no cálculo de docking molecular, maiores são as

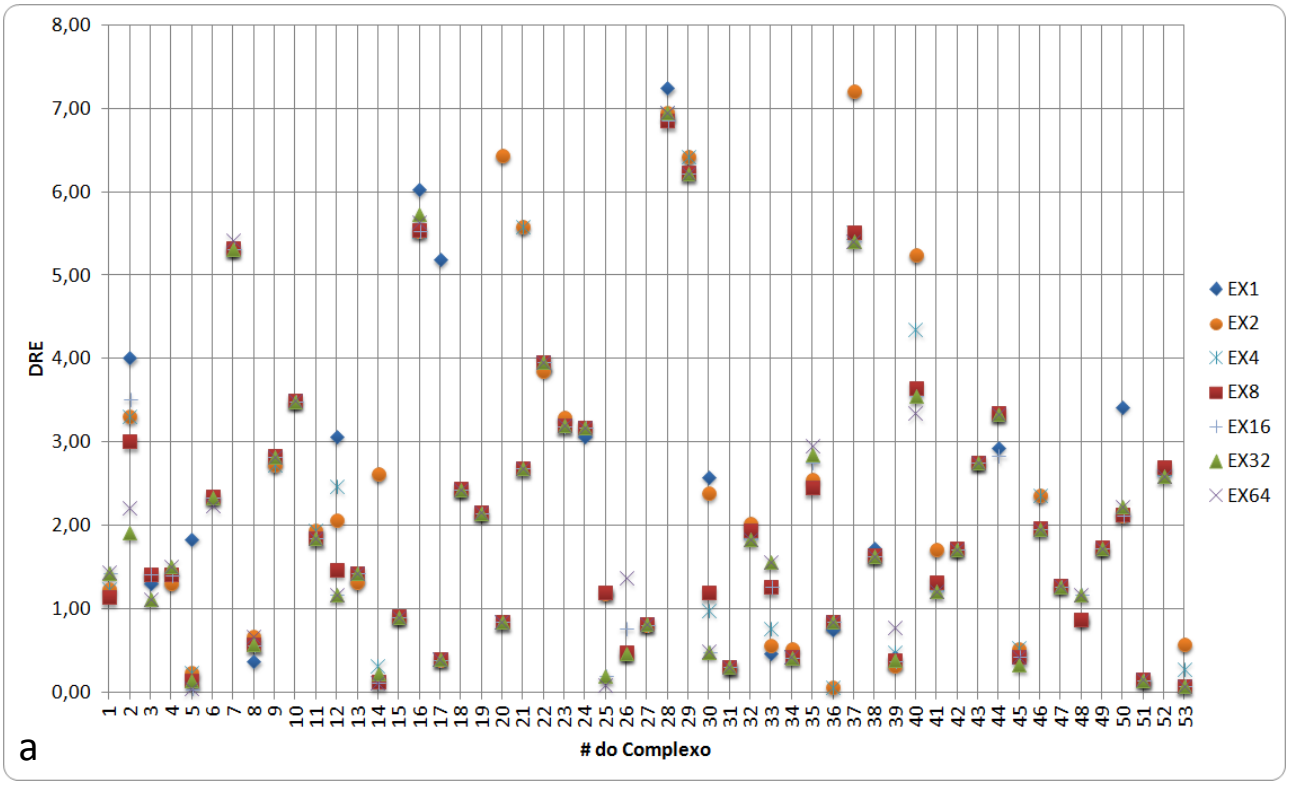
chances de encontrar soluções mais estáveis (valores mais negativos de energia livre de ligação ou *binding*).

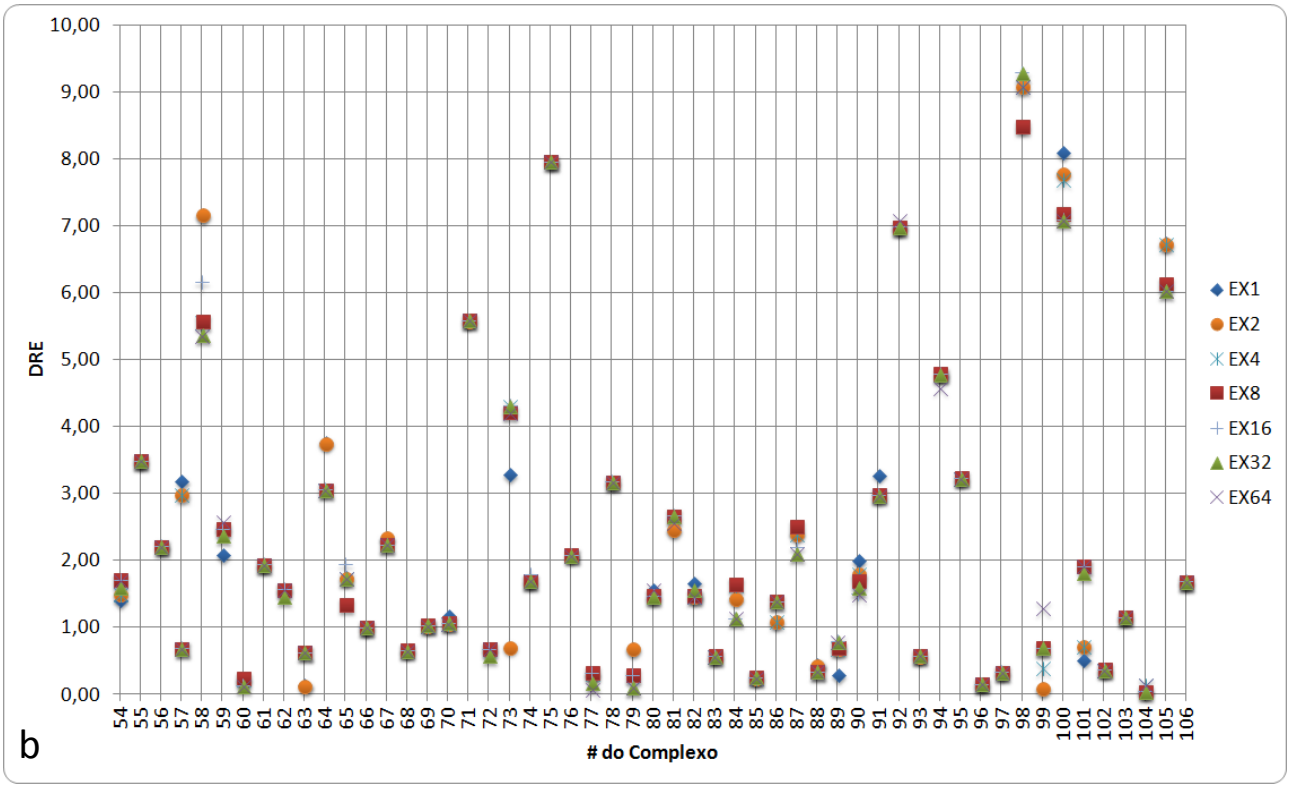
Ainda na Tabela 18, é possível observar que, em alguns casos, a diferença entre os valores de *binding* encontrados nos cálculos de *docking* efetuados pelo programa AutoDock Vina utilizando o valor padrão para o parâmetro *exhautiveness* (EX8) e os valores de binding encontrados nos cálculos de *docking* utilizando parâmetros acima do padrão foram até -1,3 Kcal/mol (PDB: 5ER2, EX32 e EX64) mais estáveis. Em geral, a utilização de valores de *exhaustiveness* acima do valor padrão (8) melhorou, em termos de estabilidade no *binding*, as soluções encontradas em 34 (EX16), 44 (EX32) e 57 (EX64) cálculos do conjunto de 213 complexos para *docking* molecular.

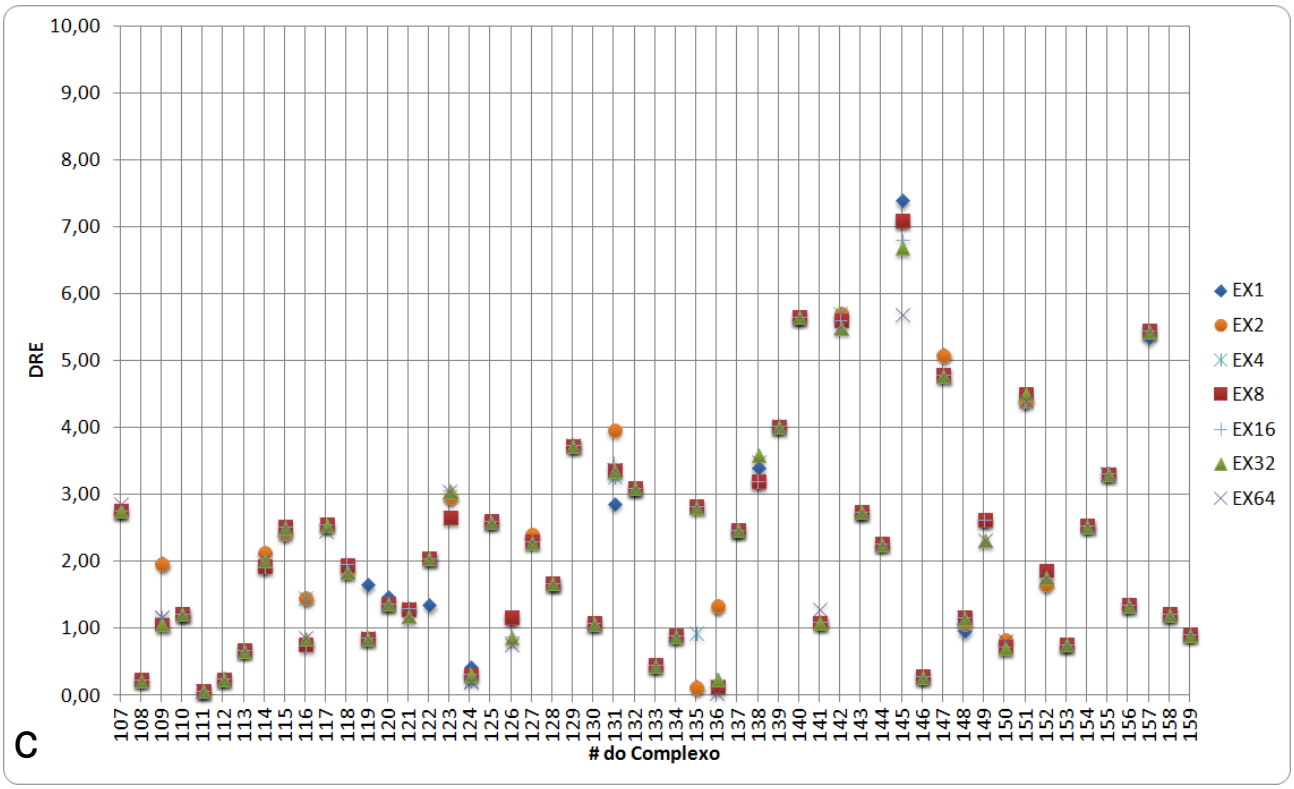
Tabela 18 – Valores de *binding* das melhores soluções de *docking* (exhaustiveness 1 à 64) e os valores obtidos experimentalmente (EXP).

PDB	EXP	EX1	EX2	EX4	EX8	EX16	EX32	EX64
1A30	-5,87	-7,1	-7,1	-7,1	-7	-7,3	-7,3	-7,3
1A94	-10,71	-6,7	-7,4	-7,4	-7,7	-7,2	-8,8	-8,5
1ADL	-7,31	-6	-5,9	-5,9	-5,9	-5,9	-6,2	-6,2
1AMW	-6,19	-7,5	-7,5	-7,6	-7,6	-7,6	-7,7	-7,7
1B39	-9,44	-7,6	-9,2	-9,2	-9,3	-9,4	-9,3	-9,4
1B7H	-10,94	-8,6	-8,6	-8,6	-8,6	-8,6	-8,6	-8,7
1B8O	-14,52	-9,2	-9,2	-9,2	-9,2	-9,2	-9,2	-9,1
1B9J	-8,13	-8,5	-8,8	-8,7	-8,7	-8,7	-8,7	-8,8
1BCU	-4,47	-7,2	-7,2	-7,2	-7,3	-7,3	-7,3	-7,3
1BGQ	-11,69	-8,2	-8,2	-8,2	-8,2	-8,2	-8,2	-8,2
1BXO	-13,64	-11,7	-11,7	-11,7	-11,8	-11,8	-11,8	-11,8
1BXQ	-10,07	-7	-8	-7,6	-8,6	-8,9	-8,9	-8,9
1C1V	-10,42	-9,1	-9,1	-9	-9	-9	-9	-9
1C88	-7,22	-4,6	-4,6	-6,9	-7,1	-7,1	-7	-7,1
1D7J	-4,50	-3,6	-3,6	-3,6	-3,6	-3,6	-3,6	-3,6
1DF8	-13,23	-7,2	-7,7	-7,7	-7,7	-7,7	-7,5	-7,6
1E66	-13,49	-8,3	-13,1	-13,1	-13,1	-13,1	-13,1	-13,1
1FKB	-13,23	-10,8	-10,8	-10,8	-10,8	-10,8	-10,8	-10,8
1FKI	-9,55	-11,7	-11,7	-11,7	-11,7	-11,7	-11,7	-11,7
1FLR_	-13,64	-7,2	-7,2	-12,8	-12,8	-12,8	-12,8	-12,8
1FTM	-10,38	-4,8	-4,8	-4,8	-7,7	-7,7	-7,7	-7,7
1G2K	-10,86	-7	-7	-6,9	-6,9	-6,9	-6,9	-6,9
1G74	-8,89	-5,6	-5,6	-5,7	-5,7	-5,7	-5,7	-5,7
1GPK	-7,33	-10,4	-10,5	-10,5	-10,5	-10,5	-10,5	-10,5
1H23	-11,39	-10,2	-10,2	-10,2	-10,2	-11,2	-11,2	-11,3
1HFS	-11,87	-11,4	-11,4	-11,4	-11,4	-11,1	-11,4	-10,5
1HNN	-8,51	-7,7	-7,7	-7,7	-7,7	-7,7	-7,7	-7,7
1IF7	-14,35	-7,1	-7,4	-7,4	-7,5	-7,5	-7,4	-7,4
1J37	-12,22	-5,8	-5,8	-5,8	-6	-6	-6	-6
1JQ8	-8,19	-5,6	-5,8	-7,2	-7	-7,7	-7,7	-7,7
1JYS	-4,80	-4,5	-4,5	-4,5	-4,5	-4,5	-4,5	-4,5
1KEL	-9,93	-7,9	-7,9	-8	-8	-8,1	-8,1	-8,1
1LBK	-4,34	-4,8	-4,9	-5,1	-5,6	-5,6	-5,9	-5,9
1LOL	-8,72	-8,2	-8,2	-8,3	-8,3	-8,3	-8,3	-8,3
1LOQ	-5,05	-7,6	-7,6	-7,5	-7,5	-7,8	-7,9	-8
1LVU	-7,26	-6,5	-7,2	-7,2	-8,1	-8,1	-8,1	-8,1
1MQ6	-15,21	-8	-8	-9,7	-9,7	-9,8	-9,8	-9,8
1N2V	-5,57	-7,3	-7,2	-7,2	-7,2	-7,2	-7,2	-7,2
1N5R	-7,72	-7,3	-7,4	-8,2	-8,1	-8,1	-8,1	-8,5
1NC1	-8,35	-3,1	-3,1	-4	-4,7	-4,8	-4,8	-5
1NJA	-8,61	-6,9	-6,9	-7,3	-7,3	-7,4	-7,4	-7,4
1NJE	-5,18	-6,9	-6,9	-6,9	-6,9	-6,9	-6,9	-6,9
1NVQ	-11,26	-8,5	-8,5	-8,5	-8,5	-8,5	-8,5	-8,5
1NWL	-3,26	-6,2	-6,6	-6,6	-6,6	-6,1	-6,6	-6,6
1O0M	-7,03	-6,6	-6,5	-6,5	-6,6	-6,6	-6,7	-6,6
1O3F	-10,86	-8,5	-8,5	-8,5	-8,9	-8,9	-8,9	-8,9
1O5B	-7,87	-6,6	-6,6	-6,6	-6,6	-6,6	-6,6	-6,6
1OS0	-8,23	-9,1	-9,1	-9,1	-9,1	-9,4	-9,4	-9,4
1P1Q	-6,67	-8,4	-8,4	-8,4	-8,4	-8,4	-8,4	-8,4
1PB8	-7,03	-3,6	-4,9	-4,9	-4,9	-4,9	-4,8	-4,8
1PBQ	-8,55	-8,7	-8,7	-8,7	-8,7	-8,7	-8,7	-8,7
1PS3	-3,11	-5,8	-5,8	-5,8	-5,8	-5,7	-5,7	-5,7
1Q84	-15,07	-14,5	-14,5	-14,8	-15	-15	-15	-15
1Q8T	-6,49	-7,9	-8	-8	-8,2	-8,2	-8,1	-8,2
1QI0	-3,21	-6,7	-6,7	-6,7	-6,7	-6,7	-6,7	-6,7
1R5Y	-8,81	-6,6	-6,6	-6,6	-6,6	-6,6	-6,6	-6,6
1RE8	-12,99	-9,8	-10	-10	-12,3	-12,3	-12,3	-12,3
1SL3	-16,17	-9	-9	-10,6	-10,6	-10	-10,8	-10,8
1SLG	-5,32	-7,4	-7,8	-7,8	-7,8	-7,8	-7,7	-7,9
1SV3	-5,93	-5,7	-5,7	-5,7	-5,7	-5,8	-5,8	-5,8
1SWR	-9,44	-7,5	-7,5	-7,5	-7,5	-7,5	-7,5	-7,5
1TRD	-7,37	-5,8	-5,8	-5,8	-5,8	-5,8	-5,9	-5,8
1TSY	-6,77	-6,9	-6,9	-7,4	-7,4	-7,4	-7,4	-7,4
1TXR	-10,56	-6,8	-6,8	-7,5	-7,5	-7,5	-7,5	-7,5
1U1B	-10,64	-8,9	-8,9	-8,9	-9,3	-8,7	-8,9	-8,9
1UTO	-3,10	-4,1	-4,1	-4,1	-4,1	-4,1	-4,1	-4,1
1V48	-10,64	-8,3	-8,3	-8,4	-8,4	-8,4	-8,4	-8,4
1VSO	-6,44	-7,1	-7,1	-7,1	-7,1	-7,1	-7,1	-7,1
1W3K	-5,87	-6,9	-6,9	-6,9	-6,9	-6,9	-6,9	-6,9
1W3L	-8,57	-7,4	-7,5	-7,5	-7,5	-7,5	-7,5	-7,5
1X1Z	-15,09	-9,5	-9,5	-9,5	-9,5	-9,5	-9,5	-9,5
1XGJ	-8,19	-7,5	-7,5	-7,5	-7,5	-7,5	-7,6	-7,5
1XH6	-9,99	-6,7	-10,7	-14,3	-14,2	-14,3	-14,3	-14,2
1Y1Z	-4,20	-5,9	-5,9	-5,9	-5,9	-6	-5,9	-5,9
1Y6Q	-15,96	-8	-8	-8	-8	-8	-8	-8
1YC1	-8,42	-10,5	-10,5	-10,5	-10,5	-10,5	-10,5	-10,5
1ZEA	-7,12	-6,8	-6,8	-6,8	-6,8	-6,8	-7,3	-7,2
2ARM	-10,48	-7,3	-7,3	-7,3	-7,3	-7,3	-7,3	-7,3
2AVQ	-5,99	-5,3	-5,3	-6,1	-5,7	-5,7	-6,1	-6,1
2B1V	-7,83	-9,4	-9,3	-9,3	-9,3	-9,3	-9,3	-9,4
2B7D	-11,87	-9,4	-9,4	-9,2	-9,2	-9,2	-9,2	-9,3
2BRB	-6,63	-8,3	-8,1	-8,1	-8,1	-8,1	-8,2	-8,1
2BZ6	-9,67	-9,1	-9,1	-9,1	-9,1	-9,1	-9,1	-9,1
2CET	-10,94	-9,5	-9,5	-9,3	-9,3	-9,8	-9,8	-9,8
2CGF	-8,74	-9	-9	-9	-9	-9	-9	-9
2CTC	-5,31	-6,4	-6,4	-6,4	-6,7	-6,7	-6,7	-6,7
2D1O	-10,50	-8,1	-8,1	-8,1	-8	-8,3	-8,4	-8,4
2D3U	-9,44	-9	-9	-9,1	-9,1	-9,1	-9,1	-9,1
2EXM	-5,61	-5,9	-6,3	-6,3	-6,3	-6,4	-6,4	-6,4
2F34	-9,40	-7,4	-7,6	-7,6	-7,7	-7,9	-7,8	-7,9
2FLR	-8,32	-11,6	-11,3	-11,3	-11,3	-11,3	-11,3	-11,3
2G5U	-11,58	-4,6	-4,6	-4,6	-4,6	-4,6	-4,6	-4,5
2G71	-10,18	-9,6	-9,6	-9,6	-9,6	-9,6	-9,6	-9,6
2G94	-12,99	-8,2	-8,2	-8,2	-8,2	-8,2	-8,2	-8,4
2G9Q	-8,73	-5,5	-5,5	-5,5	-5,5	-5,5	-5,5	-5,5
2GSS	-6,74	-6,9	-6,9	-6,9	-6,9	-6,9	-6,9	-6,9
2HA3	-4,13	-3,8	-3,8	-3,8	-3,8	-3,8	-3,8	-3,8
2HB3	-15,48	-7	-6,4	-6,4	-7	-6,2	-6,2	-6,4
2I4J	-8,40	-8,5	-8,5	-8,8	-9,1	-9,1	-9,1	-9,7
2I4X	-15,99	-7,9	-8,2	-8,3	-8,8	-8,9	-8,9	-8,9
2J47	-7,38	-7,9	-8,1	-8,1	-9,3	-9,3	-9,2	-9,3
2J77	-6,67	-6,3	-6,3	-6,3	-6,3	-6,3	-6,3	-6,3
2J78	-8,76	-7,6	-7,6	-7,6	-7,6	-7,6	-7,6	-7,6
2J94	-8,55	-8,5	-8,5	-8,7	-8,6	-8,6	-8,6	-8,7
2JBJ	-13,23	-6,5	-6,5	-6,5	-7,1	-7,2	-7,2	-7,2
2JDN	-7,59	-5,9	-5,9	-5,9	-5,9	-5,9	-5,9	-5,9
2JDU	-9,17	-6,4	-6,4	-6,4	-6,4	-6,4	-6,4	-6,3
2JDY	-5,96	-6,2	-6,2	-6,2	-6,2	-6,2	-6,2	-6,2
2OBF	-12,07	-10,1	-10,1	-10,9	-11	-10,9	-11	-10,9
2ON6	-9,22	-8	-8	-8	-8	-8	-8	-8
2OSF	-5,23	-5,3	-5,3	-5,3	-5,3	-5,3	-5,3	-5,3
2P15	-14,05	-14,3	-14,3	-14,3	-14,3	-14,3	-14,3	-14,3
2P4Y	-12,28	-11,6	-11,6	-11,6	-11,6	-11,6	-11,6	-11,6
2PGZ	-7,83	-5,9	-5,7	-5,9	-5,9	-5,9	-5,8	-5,8
2POW	-9,82	-7,4	-7,4	-7,4	-7,3	-7,3	-7,3	-7,3
2PQ9	-11,06	-9,6	-9,6	-9,6	-10,3	-10,3	-10,2	-10,2
2PU2	-6,04	-8,6	-8,6	-8,6	-8,6	-8,6	-8,6	-8,5
2QBP	-11,46	-9,6	-9,6	-9,6	-9,5	-9,5	-9,6	-9,6
2QE4	-10,86	-9,2	-10	-10	-10	-10	-10	-10
2QFT	-7,18	-5,7	-5,8	-5,8	-5,8	-5,8	-5,8	-5,8
2QM9	-10,60	-9,3	-9,3	-9,3	-9,3	-9,3	-9,4	-9,4
2QMJ	-5,74	-7,1	-7,8	-7,8	-7,8	-7,8	-7,8	-7,8
2QWB	-3,74	-6,7	-6,7	-6,4	-6,4	-6,8	-6,8	-6,8
2QWD	-6,62	-6,2	-6,3	-6,4	-6,3	-6,3	-6,3	-6,4
2QWE	-10,20	-7,6	-7,6	-7,6	-7,6	-7,6	-7,6	-7,6
2R23	-5,08	-3,9	-3,9	-3,9	-3,9	-4,2	-4,2	-4,3
2RFH	-9,30	-6,9	-6,9	-7	-7	-7	-7	-7
2RKM	-5,32	-7	-7	-7	-7	-7	-7	-7
2STD	-13,44	-9,7	-9,7	-9,7	-9,7	-9,7	-9,7	-9,7
2USN	-8,88	-7,8	-7,8	-7,8	-7,8	-7,8	-7,8	-7,8
2UWO	-11,87	-9	-7,9	-8,6	-8,5	-8,4	-8,5	-8,5
2V00	-4,99	-8,1	-8,1	-8,1	-8,1	-8,1	-8,1	-8,1
2V2H	-5,14	-5,6	-5,6	-5,6	-5,6	-5,6	-5,6	-5,6
2VL4	-8,20	-7,3	-7,3	-7,3	-7,3	-7,3	-7,3	-7,3
2VO5	-6,67	-6,8	-6,8	-7,6	-9,5	-9,5	-9,5	-9,5
2VOT	-9,74	-8,4	-8,4	-9,6	-9,6	-9,5	-9,5	-9,7
2W66	-5,53	-8	-8	-8	-8	-8	-8	-8
2WEC	-5,40	-8,8	-8,6	-8,6	-8,6	-8,6	-9	-8,9
2WN9	-11,62	-7,6	-7,6	-7,6	-7,6	-7,6	-7,6	-7,6
2X00	-15,46	-9,8	-9,8	-9,8	-9,8	-9,8	-9,8	-9,8
2X91	-9,20	-8,1	-8,1	-8,1	-8,1	-8,1	-8,1	-7,9
2X96	-5,50	-11,2	-11,2	-11,2	-11,1	-11,1	-11	-11
2XAB	-12,65	-9,9	-9,9	-9,9	-9,9	-9,9	-9,9	-9,9
2XDL	-4,23	-6,5	-6,5	-6,5	-6,5	-6,5	-6,5	-6,5
2XEG	-14,60	-7,2	-7,5	-7,5	-7,5	-7,5	-7,8	-7,9
2XJ7	-9,09	-8,8	-8,8	-8,8	-8,8	-8,8	-8,8	-8,8
2XMY	-13,59	-8,5	-8,5	-8,8	-8,8	-8,8	-8,8	-8,8
2ZB0	-9,03	-10	-10,1	-10,1	-10,2	-10,2	-10,2	-10,2
2ZCQ	-12,03	-9,4	-9,4	-9,4	-9,4	-9,4	-9,7	-9,7
2ZCR	-9,37	-10,2	-10,2	-10,1	-10,1	-10,1	-10,1	-10,2
3ACW	-6,49	-10,9	-10,9	-10,9	-10,9	-11	-10,9	-11
3ADV	-4,13	-5,8	-5,8	-6	-6	-5,9	-5,9	-5,9
3B3C	-5,43	-6,2	-6,2	-6,2	-6,2	-6,2	-6,2	-6,2
3B7I	-7,94	-5,4	-5,4	-5,4	-5,4	-5,4	-5,4	-5,4
3B92	-10,91	-7,6	-7,6	-7,6	-7,6	-7,6	-7,6	-7,6
3BFU	-8,55	-7,2	-7,2	-7,2	-7,2	-7,2	-7,2	-7,2
3BGS	-12,85	-7,5	-7,4	-7,4	-7,4	-7,4	-7,4	-7,4
3BRA	-3,68	-4,9	-4,9	-4,9	-4,9	-4,9	-4,9	-4,9
3CFT	-5,72	-4,8	-4,8	-4,8	-4,8	-4,8	-4,8	-4,8
3CJ2	-6,62	-7,9	-7,9	-7,9	-7,9	-7,9	-7,9	-7,9
3CKP	-8,66	-9,3	-9,3	-9	-9	-9,7	-9,1	-9,1
3COW	-9,41	-10,5	-10,5	-10,6	-10,6	-10,6	-10,6	-10,6
3D4Z	-6,67	-7,6	-7,6	-7,6	-7,6	-7,6	-7,6	-7,6
3DXG	-3,27	-6,4	-6,4	-6,4	-6,4	-6,4	-6,4	-6,7
3E8R	-13,29	-8,6	-9,8	-9,8	-11,4	-11,6	-11,6	-11,5
3E93	-12,07	-14	-14	-14	-14	-14	-14	-14
3EHY	-7,98	-7,4	-7,4	-7,4	-7,4	-7,4	-7,4	-7,4
3EJR	-11,69	-7	-7,1	-7,1	-7,1	-7,2	-8,5	-8,5
3F17	-11,77	-9,8	-9,8	-9,8	-9,8	-9,8	-9,8	-9,9
3F3T	-6,26	-11,3	-11,3	-11,3	-11,3	-11,4	-11,4	-11,4
3F80	-5,76	-6,1	-6,1	-6,1	-6,1	-6,1	-6,1	-6,2
3FCQ	-3,78	-5,4	-5,9	-6	-6	-6	-6	-6
3FK1	-3,57	-6,1	-6,1	-6	-6	-6,1	-6,1	-6,1
3G2Z	-3,22	-6,4	-6,4	-6,5	-6,6	-6,5	-6,5	-6,5
3G32	-5,06	-8	-8	-8	-8	-8	-8	-8
3G35	-6,38	-8,6	-8,6	-8,4	-8,4	-8,4	-8,1	-8,1
3G5D	-10,86	-9,9	-9,9	-9,9	-9,9	-9,9	-9,9	-9,9
3GBA	-12,37	-10,2	-10,2	-10,2	-10,2	-10,2	-10,1	-10,2
3GE7	-11,87	-10,7	-10,7	-10,7	-10,8	-10,8	-10,8	-10,7
3GNW	-12,41	-12,6	-12,6	-12,6	-12,6	-12,6	-12,6	-12,6
3GV9	-2,89	-5,1	-5,1	-5,1	-5,1	-5,1	-5,1	-5,1
3GY4	-6,96	-6,3	-6,3	-6,3	-6,3	-6,3	-6,3	-6,3
3HEC	-6,10	-8	-8,5	-9,9	-12,2	-12,2	-12,2	-12,2
3HZK	-9,14	-4,8	-4,8	-4,8	-4,8	-4,7	-4,7	-4,7
3HZM	-7,53	-3,9	-3,9	-3,9	-3,9	-3,9	-3,9	-3,9
3IE3	-9,11	-5,4	-5,4	-5,6	-5,9	-5,9	-5,8	-5,8
3IMC	-4,04	-6	-6	-6,1	-6,1	-6,1	-6,1	-6,1
3IUB	-6,19	-9,2	-9,8	-9,8	-9,8	-9,8	-9,8	-9,8
3IWW	-10,86	-10,2	-10,2	-10,2	-10,2	-10,2	-10,2	-10,2
3JVS	-8,92	-8,6	-8,6	-8,6	-8,6	-8,6	-8,6	-8,6
3K8Q	-15,39	-7,2	-7,2	-7,4	-7,4	-7,4	-7,4	-7,4
3KEJ	-9,71	-8,1	-10	-13,6	-13,6	-13,6	-13,6	-13,6
3KEK	-11,41	-12,5	-12,5	-12,5	-12,5	-12,6	-12,6	-12,6
3KME	-8,38	-8,2	-8,2	-8,2	-8,2	-8,1	-8,1	-8,1
3KV2	-8,58	-7,2	-7,2	-7,2	-7,3	-7,3	-7,3	-7,3
3L4U	-10,26	-5,5	-5,5	-5,5	-5,5	-5,7	-5,4	-5,7
3L4W	-8,19	-5,2	-5,5	-5,5	-5,5	-5,5	-5,5	-5,5
3L7A	-5,91	-10,6	-10,6	-10,6	-10,6	-10,7	-10,7	-10,6
3L7D	-2,97	-9,9	-9,9	-9,9	-9,9	-9,9	-9,9	-9,9
3LKA	-3,85	-6,9	-6,9	-6,9	-6,9	-6,9	-6,9	-6,9
3MFV	-3,44	-6,9	-6,8	-6,8	-6,8	-6,8	-6,8	-6,8
3NEX	-8,19	-5,4	-5,4	-5,4	-5,4	-5,3	-5,4	-5,4
3PCE	-2,73	-5,1	-5,1	-5,2	-5,3	-5,2	-5,2	-5,2
3PCJ	-9,85	-5,3	-5,3	-5,3	-5,3	-5,3	-5,3	-5,3
3PCN	-4,99	-5,7	-5,7	-5,7	-5,7	-5,7	-5,7	-5,7
3STD	-15,16	-12,6	-12,7	-12,7	-12,6	-12,6	-12,6	-12,6
456C	-13,33	-7,6	-8,5	-8,7	-8,9	-9,4	-9,6	-9,6
4ER2	-12,69	-7,7	-7,7	-7,6	-7,6	-8,2	-8,2	-8
4TIM	-2,95	-6	-6	-6	-6	-6	-6	-6
4TMN	-13,87	-7,5	-9,3	-9,6	-9,6	-9,6	-9,6	-9,6
5ER2	-8,96	-8,1	-8,1	-8,4	-8,4	-9,7	-9,7	-9,7
6CPA	-15,72	-9,3	-9,3	-9,3	-9,3	-9,3	-9,3	-9,3
6STD	-11,79	-9,7	-9,7	-9,7	-9,7	-9,7	-9,7	-9,7
MÉDIA	-8,78	-7,73	-7,84	-7,99	-8,08	-8,11	-8,14	-8,15

Na Figura 24, pode-se observar os valores obtidos para a métrica denominada "Diferença em Relação ao Experimental" (*DRE*), em relação aos valores de energia livre de ligação para as soluções de *docking* molecular encontradas pelo programa AutoDock Vina, ao serem variados os valores do parâmetro *exhaustiveness* entre 1, 2, 4, 8, 16, 32 e 64. Observa-se, na maioria dos casos encontrados nestes gráficos, os cálculos com os valores 8, 16, 32 e 64 para o parâmetro *exhaustiveness* são os que possuem o valor de *DRE* mais próximos de 0 (zero) e, consequentemente, com valores de energia livre de ligação mais próximos do que é observado experimentalmente. Isto ocorre porque, ao aumentar o valor do parâmetro *exhaustiveness*, também aumenta o número de buscas pela melhor solução de *docking,* aumentando as chances de encontrar uma solução de *docking* mais estável em comparação com os casos onde são usados os menores do parâmetro *exhaustiveness*.


8,00
7,00
6,00
5,00
4,00
3,00
2,00
1,00
0,00
DRE
EX1
EX2
EX4
EX8
EX16
EX32
EX64
do Complexo
a


10,00
9,00
8,00
7,00
6,00
5,00
4,00
3,00
2,00
1,00
0,00
DRE
EX1
EX2
EX4
EX8
EX16
EX32
EX64
do Complexo
b



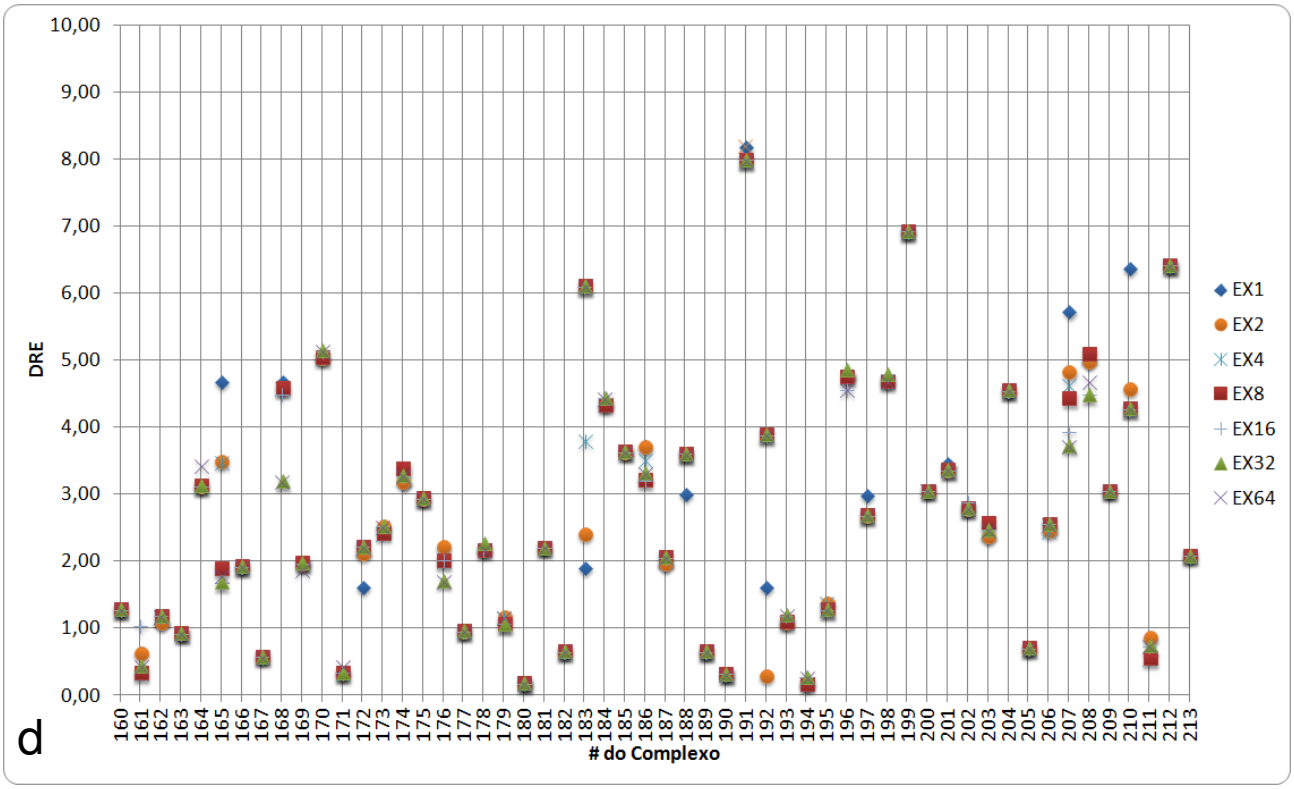


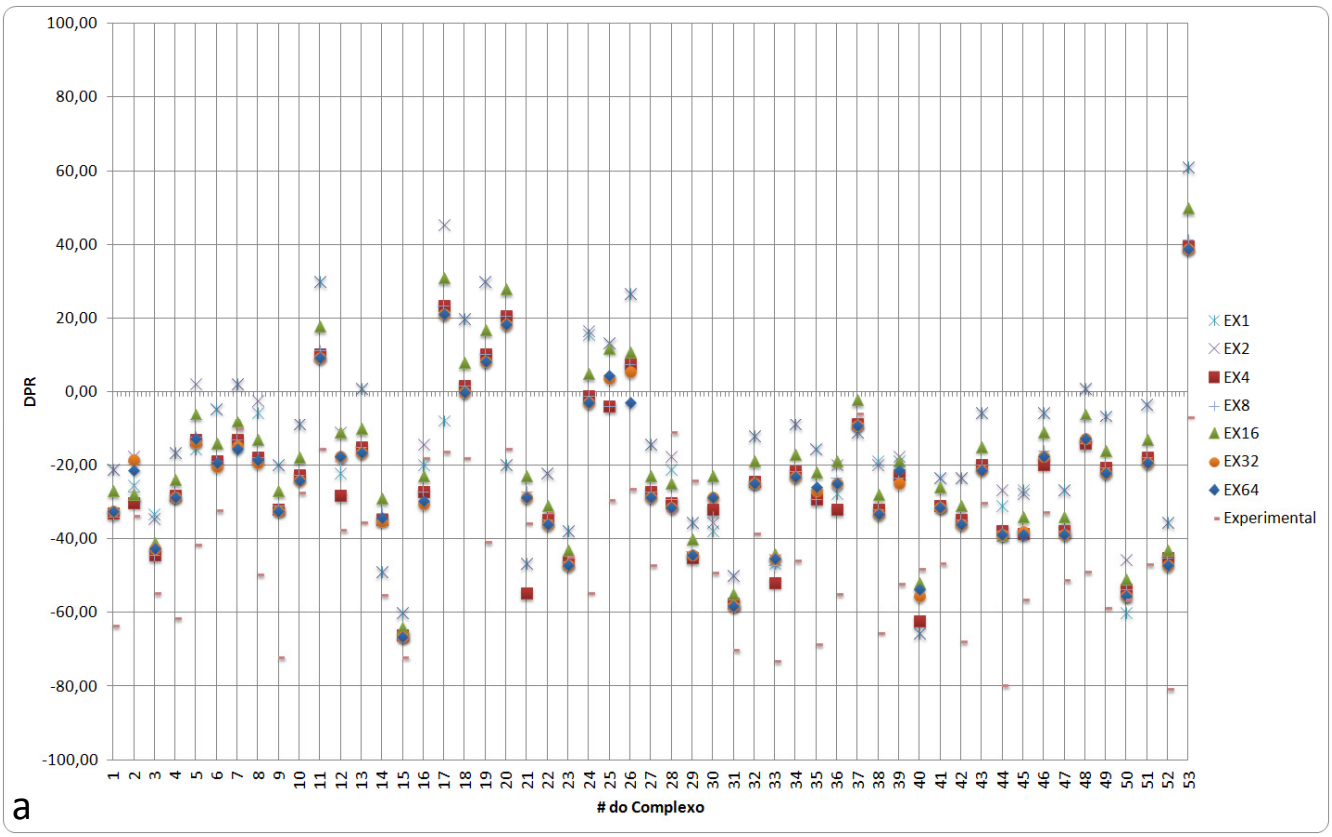
Figura 24 - Gráfico dos valores de *DRE* para o programa AutoDock Vina. a) Complexos de 1 à 53. b) Complexos de 54 a 106. c) Complexos de 107 à 159. d) Complexos de 160 à 213. A legenda no formato EXy (onde y se refere ao valor usado para o parâmetro exhaustiveness).

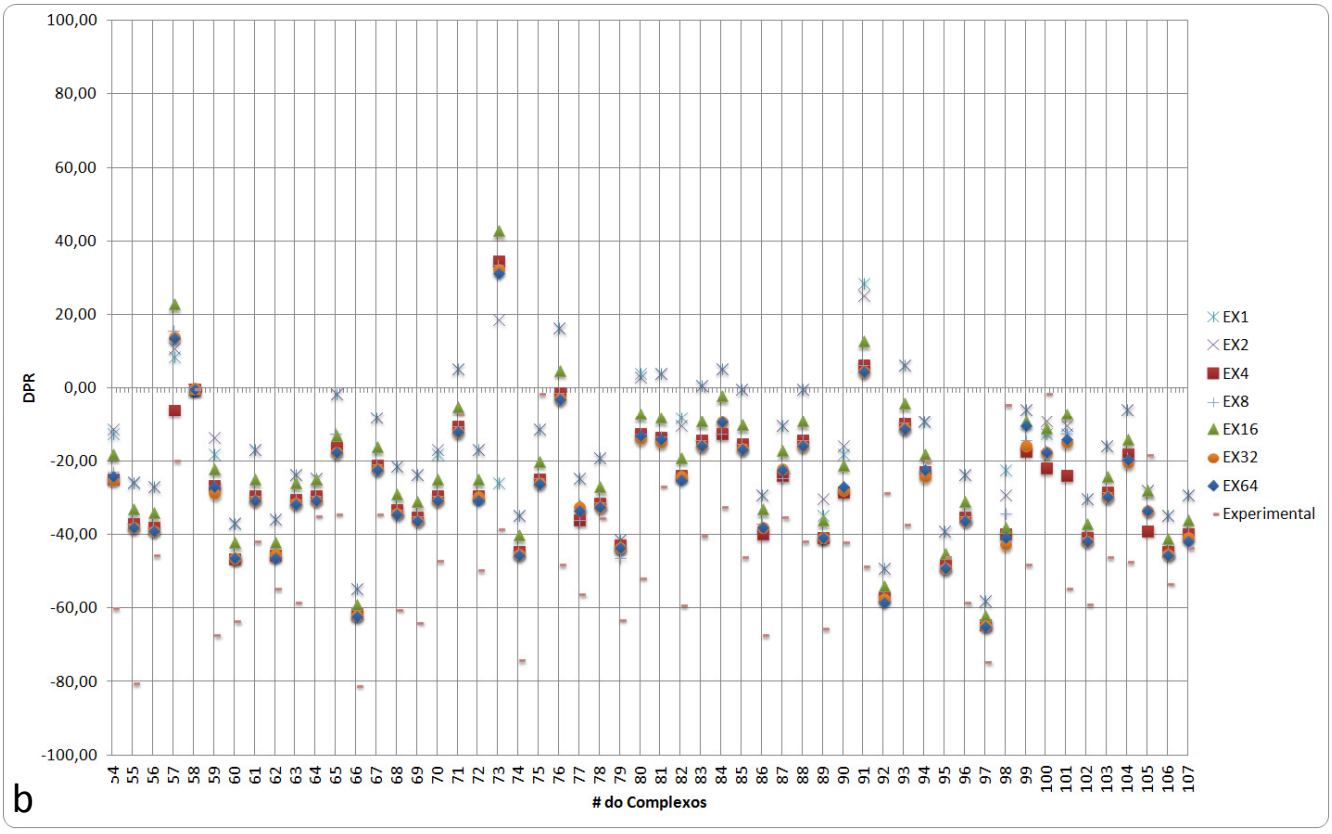
Pode-se observar na Figura 25 os valores obtidos para a métrica denominada "Diferença Percentual Relativa" (DPR) de energia livre de ligação para cada complexo, em relação ao complexo de referência (PDB: 1SL3), considerando-se os resultados experimentais. Além disso, também são apresentados nestas figuras os dados de DPR para as soluções de *docking* molecular encontradas com o programa AutoDock Vina, ao serem alternados os valores do parâmetro *exhaustiveness* entre 1, 2, 4, 8, 16, 32 e 64.

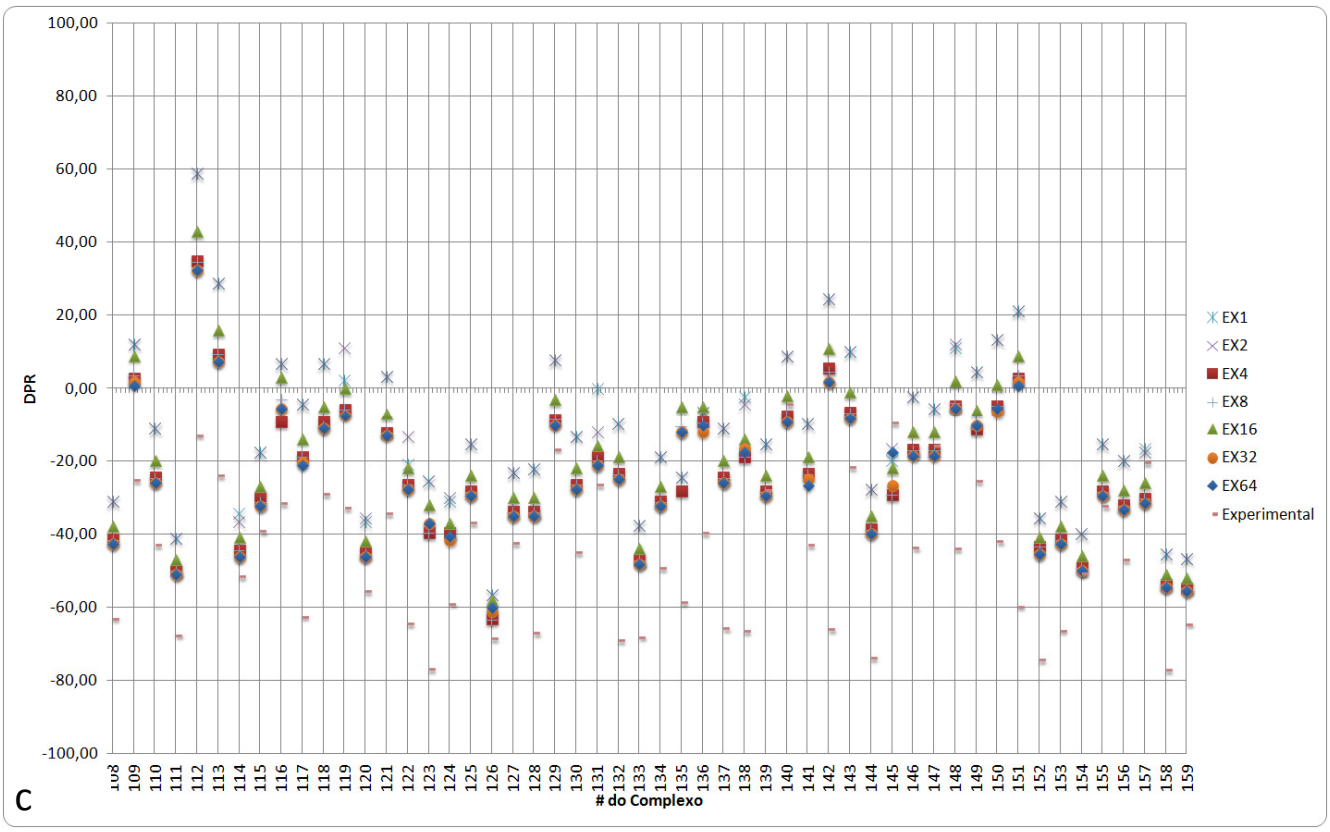
Observa-se que, na maioria dos casos, o aumento do valor do parâmetro *exhaustiveness* causa tipicamente uma aproximação dos valores de *DPR* obtidos nas soluções de *docking* em relação ao valor de DPR experimental. Por exemplo, enquanto o valor de DPR para o complexo #210 (PDB: 4TMN) obtido pela solução de *docking* utilizando-se os valores 32 e 64 do parâmetro exhaustiveness foi de -11, ou seja, 11% menos estável que o complexo de referência (PDB: 1SL3), este valor está mais próximo do valor de DPR experimental, que é -14, ou seja, 14% menos estável que o 1SL3, comparado com o valor de DPR obtido pela solução de *docking* utilizando o valor padrão (8) do parâmetro exhaustiveness que é de -9, ou seja, 9% menos estável que o complexo 1SL3. Como, geralmente, estudos de *virtual screening* são realizados antes dos testes experimentais, tipicamente com o intuito de reduzir o tempo e os custos do desenvolvimento de um novo fármaco, descartando-se as moléculas menos promissoras como candidatas, é muito importante que os métodos de *docking* molecular utilizados para este fim obtenham os valores mais confiáveis de *binding* observados experimentalmente e, desta forma, reduzam ao máximo o número de falsos positivos e falsos negativos nesta etapa inicial do planejamento de um novo fármaco. Desta forma, pode-se dizer que ao aumentar o valor do parâmetro *exhaustiveness* do programa AutoDock Vina, é possível prever com maior precisão o comportamento de um conjunto de moléculas de interesse em um caso de *virtual screening,* principalmente se for levada em conta a estabilidade relativa dos complexos (DPR), ou seja, quais complexos são mais

estáveis do que outros, e quanto mais estáveis.

O complexo #58 (PDB: 1SL3) foi escolhido como referência devido ao fato de que este complexo possui o menor valor de energia livre de ligação medido experimentalmente (-16,17 Kcal/mol), quando comparado com os outros complexos do conjunto de teste (213 complexos, no total) utilizado neste trabalho. Observou-se, em alguns casos, soluções de *docking* molecular encontradas pelo programa AutoDock Vina com valores positivos de *DPR*, indicando que os valores de energia livre preditos pela solução de *docking* para estes complexos seriam mais negativos (mais estáveis) do que os valores preditos pela solução de *docking* para o complexo 1SL3 (o mais estável experimentalmente). Por exemplo, o complexo #53 (PDB: 1Q84), que nos cálculos com o valor 8 para o parâmetro *exhaustiveness* possui um DPR de 42, indicando que ele é 42% mais estável que o 1SL3. Isto ocorre porque, embora o complexo 1SL3 seja o mais estável (menor energia) observado experimentalmente, ele não foi o complexo mais estável encontrado pelas soluções de *docking* com o programa AutoDock Vina, considerando-se todos os valores de *exhaustiveness* testados (1 à 64). Ainda no exemplo acima, a melhor solução de *docking* molecular para o complexo 1Q84, ao utilizar o valor 8 para o parâmetro *exhaustiveness*, foi de -15 Kcal/mol, enquanto que o valor de energia livre de ligação encontrado pela solução de *docking*, com o mesmo valor para o parâmetro exhaustiveness (8), foi de -10,6 Kcal/mol para o complexo 1SL3. Experimentalmente, observa-se uma energia livre de *binding* de -15,07 Kcal/mol para o complexo 1Q84 e -16,17 Kcal/mol para o complexo 1SL3.


100,00
80,00
60,00
40,00
20,00
0,00
-20,00
-40,00
-60,00
-80,00
-100,00
DPR
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
do Complexo
EX1
EX2
EX4
EX8
EX16
EX32
EX64
Experimental
a


100,00
80,00
60,00
40,00
20,00
0,00
-20,00
-40,00
-60,00
-80,00
-100,00
DPR
54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107
do Complexos
EX1
EX2
EX4
EX8
EX16
EX32
EX64
Experimental
b



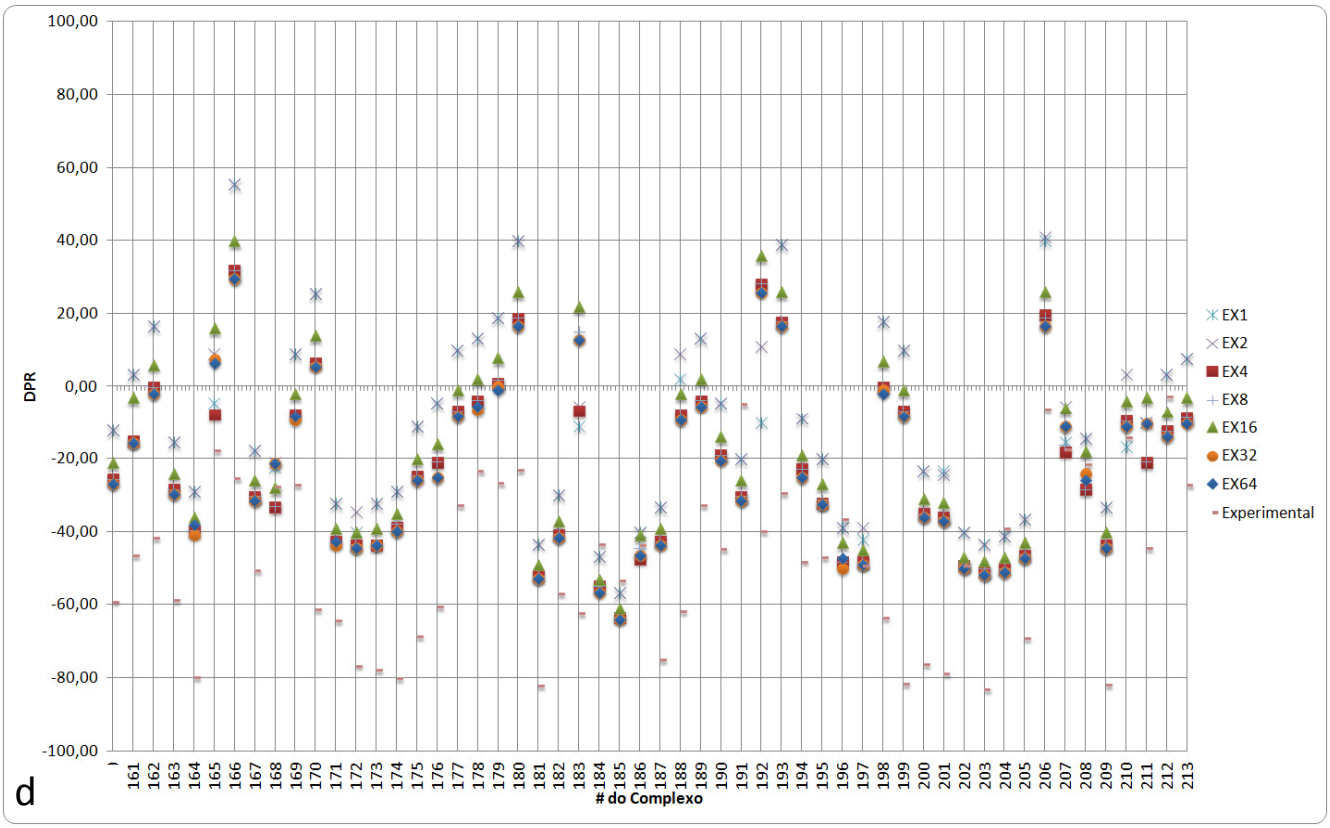


Figura 25 - Gráfico dos valores de *DPR* para o programa AutoDock Vina. a) Complexos de 1 à 53. b) Complexos de 54 à 107. c) Complexos de 108 à 159. d) Complexos de 160 à 213. A legenda no formato EXy (onde y se refere ao valor usado para o parâmetro exhaustiveness).

Tabela 19 - Tabela com os valores de RMSD das melhores soluções de *docking* (exhaustiveness 1 até 64), quando comparadas com as respectivas estruturas experimentais co-cristalizadas.

PDB	EX1	EX2	EX4	EX8	EX16	EX32	EX64	PDB	EX1	EX2	EX4	EX8	EX16	EX32	EX64	PDB	EX1	EX2	EX4	EX8	EX16	EX32	EX64
1A30	4,34	2,87	2,87	2,87	2,92	2,92	2,92	**1XGJ**	3,01	2,41	2,41	2,41	2,50	2,50	2,50	**2XAB**	0,20	0,20	0,19	0,19	0,20	0,19	0,19
1A94	12,56	11,51	11,51	7,35	8,17	1,61	1,45	**1XH6**	10,70	12,68	1,10	1,10	1,09	1,09	0,38	**2XDL**	0,34	0,32	0,32	0,32	0,32	0,33	0,33
1ADL	4,39	4,32	4,32	2,55	2,49	2,49	2,51	**1Y1Z**	0,18	0,18	0,18	0,17	0,18	0,17	0,17	**2XEG**	7,68	7,68	7,68	7,27	7,27	6,95	6,62
1AMW	1,64	1,64	1,22	1,96	1,96	1,96	1,98	**1Y6Q**	0,45	0,45	0,45	0,49	0,47	0,47	0,47	**2XJ7**	0,30	0,30	0,29	0,29	0,30	0,30	0,30
1B39	5,38	2,64	2,64	2,64	2,14	2,35	1,24	**1YC1**	0,81	0,81	0,81	0,81	0,81	0,82	0,80	**2XMY**	3,78	3,01	2,09	2,09	2,78	2,68	2,56
1B7H	1,45	1,45	1,17	1,17	1,17	1,17	1,16	**1ZEA**	5,22	5,34	7,06	9,12	7,34	7,23	7,23	**2ZB0**	2,40	2,43	2,43	1,29	1,29	1,29	1,29
1B8O	0,22	0,22	0,22	0,22	0,22	0,22	0,24	**2ARM**	4,34	4,34	3,27	2,83	2,83	4,25	4,25	**2ZCQ**	5,05	2,79	2,98	3,66	4,62	4,97	5,37
1B9J	1,21	1,27	1,31	1,28	1,09	1,09	1,26	**2AVQ**	8,17	6,18	6,18	6,30	6,08	5,04	5,65	**2ZCR**	1,36	1,36	1,36	1,47	1,47	1,47	1,47
1BCU	0,69	0,69	0,69	0,69	0,69	0,67	0,67	**2B1V**	0,86	0,82	0,82	0,82	0,82	0,82	0,86	**3ACW**	1,66	1,66	1,53	1,33	1,64	1,32	1,29
1BGQ	0,50	0,50	0,50	0,50	0,50	0,50	0,50	**2B7D**	1,12	1,12	2,09	2,09	2,09	2,09	1,19	**3ADV**	1,09	1,09	1,09	4,43	4,43	5,70	5,75
1BXO	0,51	0,51	0,36	0,49	0,49	0,49	0,52	**2BRB**	1,42	1,42	1,40	1,42	1,42	1,41	1,41	**3B3C**	3,07	2,79	2,79	2,80	2,80	2,80	2,79
1BXQ	5,26	7,36	7,38	1,61	1,61	1,61	1,61	**2BZ6**	1,37	1,37	1,37	1,37	1,37	1,52	0,93	**3B7I**	1,65	1,65	1,65	1,65	1,65	1,65	1,65
1C1V	1,77	1,77	1,77	1,77	1,34	1,34	1,34	**2CET**	4,77	4,77	4,25	1,19	1,05	1,05	0,94	**3B92**	1,35	1,35	1,31	1,31	1,35	1,16	1,15
1C88	6,73	6,73	0,93	0,89	0,89	0,91	0,90	**2CGF**	0,53	0,50	0,50	0,50	0,50	0,50	0,50	**3BFU**	0,99	0,99	0,99	0,37	0,37	0,37	0,98
1D7J	1,78	1,78	1,80	1,72	1,32	1,32	1,32	**2CTC**	1,49	1,49	1,49	1,49	1,11	1,11	1,11	**3BGS**	0,74	0,76	0,76	0,69	0,69	0,77	0,79
1DF8	0,93	0,24	0,24	0,24	0,24	0,61	0,31	**2D1O**	4,03	1,70	1,70	1,70	1,70	1,66	1,66	**3BRA**	1,96	1,96	1,93	1,93	1,91	1,87	1,88
1E66	5,54	0,27	0,27	0,27	0,27	0,27	0,27	**2D3U**	0,71	1,14	0,37	1,01	0,99	1,00	1,00	**3CFT**	4,43	2,89	2,87	1,74	1,78	1,78	4,26
1FKB	0,49	0,48	0,50	0,50	0,49	0,49	0,49	**2EXM**	5,38	3,00	2,96	2,61	5,72	5,70	2,85	**3CJ2**	0,49	0,49	0,49	0,49	0,49	0,49	0,49
1FKI	0,22	0,22	0,22	0,22	0,22	0,22	0,22	**2F34**	1,36	1,25	1,25	1,41	1,44	1,33	1,18	**3CKP**	0,95	0,95	1,13	1,13	0,60	0,60	0,60
1FLR_	7,43	4,71	0,66	0,66	0,66	0,66	0,66	**2FLR**	2,42	2,42	2,42	2,42	0,84	0,84	0,87	**3COW**	0,71	0,71	0,70	0,92	0,67	0,68	0,68
1FTM	11,67	11,67	11,67	0,86	0,86	0,86	0,86	**2G5U**	2,19	2,19	2,11	2,11	2,11	0,86	0,86	**3D4Z**	0,91	0,88	0,88	0,88	0,88	0,87	0,87
1G2K	2,25	2,25	2,35	2,35	2,39	2,39	2,02	**2G71**	0,75	0,75	0,75	0,75	0,75	0,75	0,75	**3DXG**	4,13	4,13	2,63	2,59	2,68	3,85	3,85
1G74	3,10	3,10	2,00	2,00	2,00	2,00	2,00	**2G94**	11,28	3,16	3,16	3,16	3,31	3,31	3,27	**3E8R**	7,84	7,84	7,84	2,08	2,08	1,02	1,48
1GPK	3,61	0,23	0,23	0,24	0,23	0,24	0,24	**2G9Q**	0,46	0,47	0,47	0,47	0,47	0,46	0,46	**3E93**	0,37	0,37	0,37	0,39	0,37	0,39	0,39
1H23	3,50	3,33	3,22	3,22	3,22	3,22	3,13	**2GSS**	3,24	3,24	2,47	2,53	1,88	1,88	2,94	**3EHY**	3,48	1,82	1,09	0,75	0,75	0,75	0,75
1HFS	1,06	1,06	1,06	1,06	1,26	1,27	1,28	**2HA3**	1,76	1,76	0,79	1,61	1,61	1,11	1,11	**3EJR**	4,67	4,62	4,67	4,67	2,76	1,05	1,05
1HNN	1,17	1,17	1,17	1,17	1,17	1,17	1,17	**2HB3**	5,60	4,35	4,35	4,35	4,42	4,40	4,41	**3F17**	3,64	3,64	2,14	2,14	2,14	2,14	2,05
1IF7	2,08	2,08	2,08	2,08	2,81	3,05	3,05	**2I4J**	9,30	9,02	9,02	1,32	1,32	8,74	8,20	**3F3T**	0,60	0,60	0,59	0,60	0,60	0,60	0,60
1J37	2,70	2,70	2,70	0,76	1,11	1,08	1,08	**2I4X**	6,24	1,58	1,58	1,58	1,18	2,24	1,93	**3F80**	1,36	1,36	1,36	1,36	1,36	1,36	0,98
1JQ8	6,89	6,89	3,16	3,86	4,59	4,47	4,47	**2J47**	5,87	4,19	4,19	0,61	0,58	0,58	0,58	**3FCQ**	3,39	1,57	1,68	1,50	1,53	1,12	0,62
1JYS	1,79	0,97	0,97	0,97	0,97	0,95	0,96	**2J77**	0,31	0,31	0,31	0,30	0,30	0,30	0,30	**3FK1**	4,72	3,38	3,58	4,33	5,62	5,62	5,62
1KEL	1,36	1,36	1,14	1,14	1,23	0,87	0,87	**2J78**	0,31	0,31	0,31	0,31	0,31	0,31	0,31	**3G2Z**	2,37	1,72	1,72	2,39	2,43	2,37	2,44
1LBK	5,84	5,43	5,43	5,43	3,06	5,88	5,88	**2J94**	0,65	0,65	0,39	0,52	0,52	0,52	0,46	**3G32**	3,29	3,29	3,33	2,86	2,26	1,05	1,04
1LOL	2,46	2,37	2,37	2,44	2,61	1,48	0,43	**2JBJ**	1,65	1,65	1,65	1,45	1,27	1,40	1,36	**3G35**	0,36	0,36	0,86	0,86	0,86	0,84	0,84
1LOQ	1,21	1,21	1,21	1,05	1,06	1,04	1,05	**2JDN**	3,47	1,82	1,82	0,92	2,38	0,91	2,32	**3G5D**	1,50	1,88	1,88	1,74	1,74	0,81	0,80
1LVU	5,92	5,61	4,02	1,09	1,09	1,09	1,09	**2JDU**	0,23	0,23	0,23	0,23	0,22	0,22	0,41	**3GBA**	0,16	0,17	0,17	0,17	0,16	0,16	0,16
1MQ6	6,32	6,32	1,42	1,42	1,41	1,41	1,38	**2JDY**	0,52	0,52	0,52	0,52	0,52	0,52	0,52	**3GE7**	0,93	0,93	0,92	0,92	0,92	0,92	0,94
1N2V	2,76	2,62	0,64	0,82	0,66	0,50	0,50	**2OBF**	1,68	1,68	0,32	0,31	0,35	0,29	0,41	**3GNW**	0,30	0,30	0,30	0,30	0,30	0,30	0,29
1N5R	5,11	1,41	1,41	1,09	1,09	1,09	1,10	**2ON6**	2,67	2,67	2,68	1,60	1,56	1,56	1,57	**3GV9**	3,01	1,22	1,22	1,24	1,42	0,78	1,81
1NC1	9,12	9,12	2,44	2,44	2,44	2,44	2,44	**2OSF**	2,25	1,96	1,15	1,15	1,15	1,16	1,16	**3GY4**	1,90	1,47	0,56	0,48	1,15	1,15	0,72
1NJA	3,37	3,37	3,37	3,37	3,37	3,34	3,34	**2P15**	0,28	0,28	0,28	0,28	0,28	0,28	0,28	**3HEC**	9,05	8,87	8,42	0,52	0,53	0,54	0,54
1NJE	4,61	4,60	4,60	4,47	4,45	1,29	4,51	**2P4Y**	0,88	0,58	0,58	0,58	0,58	0,58	0,58	**3HZK**	4,24	4,24	5,16	4,27	5,43	4,16	4,16
1NVQ	1,77	1,77	3,24	3,24	3,24	3,21	3,20	**2PGZ**	4,27	4,27	4,27	4,27	4,27	4,27	4,31	**3HZM**	3,18	2,97	2,97	2,97	3,19	3,19	3,19
1NWL	7,07	5,65	5,65	5,65	5,65	5,65	5,46	**2POW**	1,43	1,43	1,01	1,04	1,10	1,09	1,09	**3IE3**	3,04	3,04	3,42	3,11	3,11	3,42	3,40
1O0M	1,07	1,14	1,14	1,04	1,03	1,12	1,01	**2PQ9**	2,69	2,69	2,68	0,97	0,97	0,97	1,15	**3IMC**	0,28	0,28	0,28	0,26	0,26	0,26	0,25
1O3F	6,85	5,74	2,14	1,65	1,44	1,44	1,41	**2PU2**	6,35	6,35	2,21	2,21	0,74	0,74	0,74	**3IUB**	7,55	2,60	1,65	1,65	1,00	1,00	1,01
1O5B	1,02	1,02	1,02	1,00	1,00	1,00	1,00	**2QBP**	1,18	1,18	1,18	1,20	1,20	1,19	1,07	**3IWW**	0,77	0,77	0,69	0,69	0,69	0,69	0,69
1OS0	6,72	6,44	1,46	1,46	1,46	1,46	1,21	**2QE4**	3,82	0,58	0,58	0,58	0,58	0,57	0,57	**3JVS**	1,49	1,49	1,18	1,20	1,20	0,81	0,84
1P1Q	2,30	1,15	1,15	1,15	1,15	1,19	1,19	**2QFT**	4,55	0,89	0,89	0,89	0,89	0,89	0,89	**3K8Q**	1,47	1,60	1,60	1,60	1,02	2,24	1,83
1PB8	7,35	0,74	0,74	0,74	0,74	0,76	0,76	**2QM9**	4,37	0,77	0,77	0,77	0,77	0,82	0,82	**3KEJ**	12,05	7,33	0,61	0,34	0,34	0,34	0,34
1PBQ	0,27	0,81	0,81	0,81	0,81	0,81	0,81	**2QMJ**	5,97	2,01	2,01	2,01	2,01	2,01	2,01	**3KEK**	0,64	0,64	0,64	0,73	0,65	0,73	0,73
1PS3	3,65	3,65	3,65	3,65	3,58	3,53	3,53	**2QWB**	0,55	0,55	0,82	0,82	0,86	0,86	0,85	**3KME**	0,52	0,52	0,52	0,52	0,56	0,56	0,56
1Q84	0,55	0,55	0,51	0,40	0,40	0,40	0,40	**2QWD**	0,87	0,88	0,91	0,63	0,63	0,63	0,69	**3KV2**	1,26	1,26	1,38	1,25	1,25	1,25	1,25
1Q8T	1,46	1,46	1,46	1,46	1,47	1,21	2,19	**2QWE**	0,76	0,76	0,76	0,75	0,50	0,46	0,46	**3L4U**	1,87	1,87	1,87	1,87	1,56	1,56	1,56
1QI0	5,04	6,92	6,92	6,92	6,41	6,92	6,92	**2R23**	4,48	4,78	4,25	4,84	4,86	4,86	4,95	**3L4W**	0,69	0,68	0,53	0,53	0,74	0,67	0,67
1R5Y	4,50	1,53	1,53	1,53	1,53	1,53	1,76	**2RFH**	1,21	1,21	1,23	1,40	1,23	1,16	1,16	**3L7A**	1,42	1,42	1,42	1,44	1,44	1,40	0,62
1RE8	9,04	6,25	6,25	1,15	1,15	1,15	1,15	**2RKM**	0,74	0,74	0,74	0,73	1,03	1,03	1,03	**3L7D**	1,58	1,55	1,29	1,29	1,27	1,27	1,27
1SL3	7,88	5,60	0,69	0,69	0,65	0,65	0,65	**2STD**	1,13	1,13	1,13	1,13	1,13	1,13	0,37	**3LKA**	5,26	1,79	1,69	1,84	1,82	2,01	2,37
1SLG	7,89	7,82	6,46	6,46	6,46	6,46	6,46	**2USN**	2,28	2,28	2,28	2,28	2,28	1,87	1,87	**3MFV**	1,71	1,71	1,71	1,71	1,48	1,02	1,44
1SV3	2,26	1,23	1,23	2,26	2,11	2,01	2,01	**2UWO**	4,72	1,29	0,40	0,40	0,39	0,40	0,40	**3NEX**	7,25	5,25	3,77	5,07	5,07	4,38	7,32
1SWR	0,51	0,97	0,97	0,97	0,96	0,64	0,64	**2V00**	1,36	1,40	1,39	1,39	1,39	1,39	0,71	**3PCE**	3,02	3,54	5,20	5,23	5,23	4,51	4,51
1TRD	1,30	1,30	1,81	0,69	1,40	1,44	1,42	**2V2H**	1,92	1,92	1,92	0,89	1,48	1,48	1,48	**3PCJ**	4,20	4,20	4,36	4,19	0,97	4,19	4,19
1TSY	1,01	1,01	0,74	0,74	0,74	0,74	0,74	**2VL4**	0,42	0,42	0,42	0,42	0,42	0,42	0,42	**3PCN**	1,96	1,96	1,96	1,96	3,76	1,53	1,37
1TXR	3,03	1,70	1,64	1,64	1,60	1,88	1,60	**2VO5**	2,97	4,52	3,62	1,18	0,95	0,95	0,95	**3STD**	1,50	1,49	1,49	1,18	1,18	1,17	1,17
1U1B	3,08	3,08	2,55	1,77	1,72	1,72	1,72	**2VOT**	3,62	3,84	1,52	1,52	0,70	0,70	1,47	**456C**	9,30	3,01	2,76	0,76	1,01	0,80	0,81
1UTO	1,10	1,10	1,10	1,10	1,10	1,06	1,06	**2W66**	0,35	0,35	0,33	0,33	0,33	0,33	0,33	**4ER2**	2,61	2,61	2,61	3,01	3,01	3,01	3,63
1V48	1,03	1,03	1,29	1,02	1,02	0,92	0,88	**2WEC**	3,94	3,94	3,94	3,94	3,94	5,95	4,10	**4TIM**	1,89	1,74	1,74	1,74	1,74	1,89	1,89
1VSO	2,13	2,13	2,13	1,68	1,68	1,87	1,18	**2WN9**	4,59	5,76	5,76	5,76	5,22	5,76	5,76	**4TMN**	7,10	1,36	1,64	1,64	1,40	1,40	1,30
1W3K	1,82	1,67	1,67	1,67	1,20	1,67	1,67	**2X00**	0,78	0,78	0,78	0,78	0,78	0,78	0,78	**5ER2**	8,63	8,63	9,05	6,57	2,29	2,29	1,15
1W3L	4,30	1,94	1,94	1,94	0,60	0,60	0,60	**2X91**	7,11	7,31	7,31	7,31	7,24	7,02	3,23	**6CPA**	3,70	3,70	3,25	3,25	2,29	2,29	2,52
1X1Z	0,49	0,50	0,50	0,49	0,50	0,50	0,48	**2X96**	5,35	5,35	5,35	7,68	6,31	7,68	6,32	**6STD**	0,20	0,19	0,24	0,22	0,24	0,24	0,24
																Média	3,05	2,49	2,12	1,80	1,74	1,72	1,70

Observa-se na Tabela 19 os melhores valores de RMSD entre as soluções de *docking* encontradas pelo programa AutoDock Vina, e a solução cristalográfica (referência experimental no arquivo PDB correspondente), ao alternar os valores do parâmetro *exhaustiveness* em 1, 2, 4, 8, 16, 32 e 64, para cada complexo (receptor + ligante). É possível observar que, em média, os valores de RMS melhoraram (diminuem, se aproximando de zero) gradativamente conforme é aumentado o valor de *exhaustiveness*, variando de 3,05 (EX1) à 1,70 (EX64). Desta forma, quanto mais alto for o valor do parâmetro *exhaustiveness*, maior é a precisão com que o programa AutoDock Vina encontra soluções de *docking* mais próximas da posição encontrada na estrutura cristalográfica, obtida experimentalmente, do respectivo complexo (Receptor + Ligante), quando comparado com os valores mais baixos deste parâmetro (*exhaustiveness*). Em outras palavras, quanto mais alto for o valor do parâmetro *exhaustiveness*, melhor é o resultado previsto de *re-docking*.

No gráfico da Figura 26, é possível observar os menores valores de RMSD obtidos pelas soluções de *docking* encontradas pelo programa AutoDock Vina ao alternar os diferentes valores do parâmetro *exhaustiveness*. Observa-se que, ao aumentar os valores do parâmetro *exhaustiveness,* é possível, na maioria dos casos, reduzir o valor de RMSD e, consequentemente, obter resultados (soluções de *docking* molecular) mais próximos dos observados experimentalmente. Observa-se também que os piores valores de RMSD encontrados (mais elevados) vieram dos cálculos com *exhaustiveness* 1, 2 e 4. Isto era esperado, uma vez que nestes casos estão sendo utilizados valores mais baixos do que o valor padrão (8) e recomendado pelo desenvolvedor original do programa. Se ocorresse que estes cálculos realizados com valores menores de *exhaustivess* obtivessem valores melhores (mais baixos) de RMSD que o valor padrão (*exhautiveness* 8), não faria sentido o valor padrão deste parâmetro ser mais elevado, uma vez que isto causa um impacto significativo na demanda computacional do cálculo de *docking* molecular.

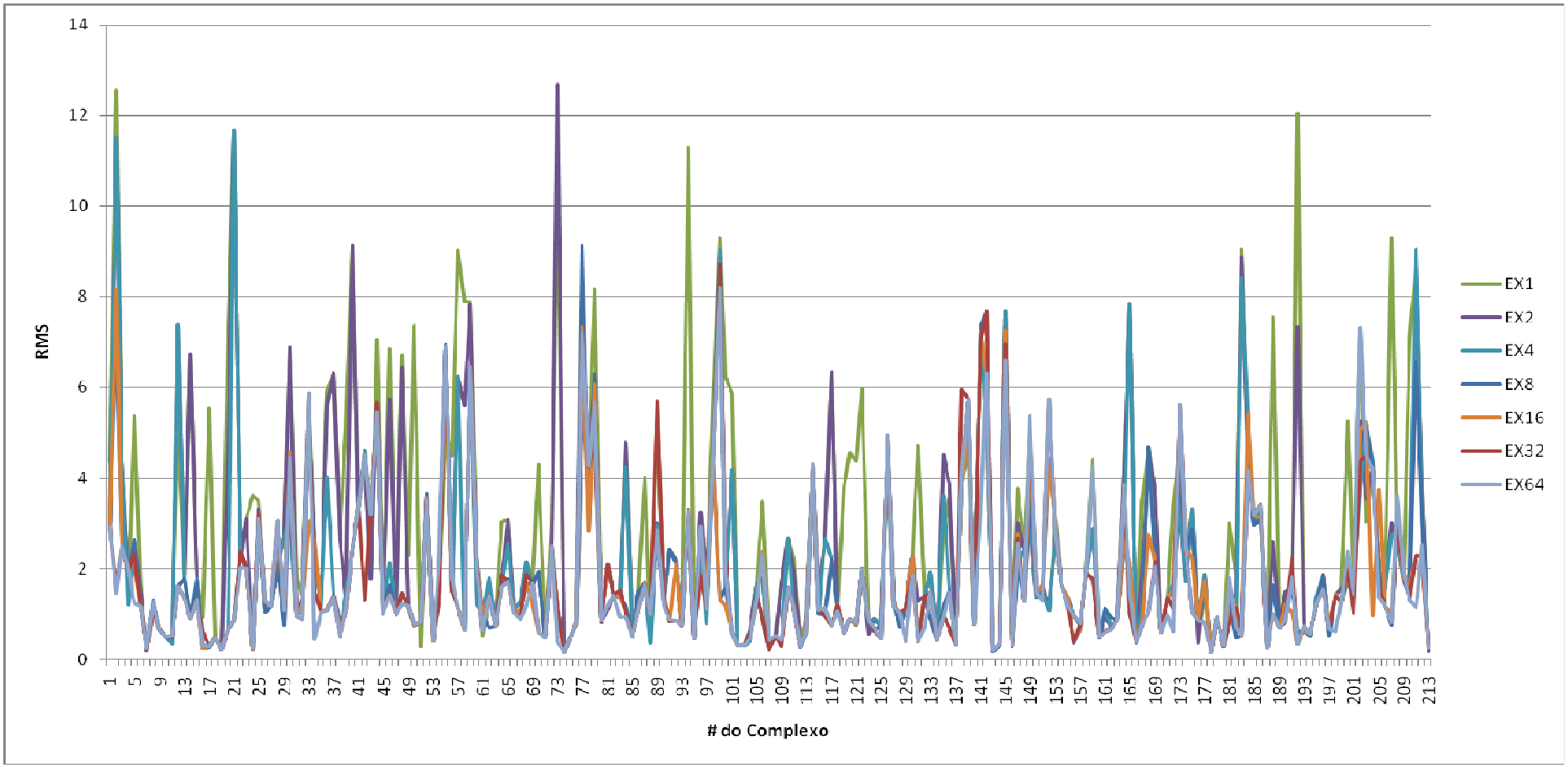


Figura 26 – Gráfico dos valores de RMSD das soluções de *docking* molecular encontradas pelo programa AutoDock Vina e a posição cristalográfica. A legenda no formato EXy (onde y se refere ao valor usado para o parâmetro exhaustiveness.

Observa-se também, na Figura 26, que os valores 16, 32 e 64 do parâmetro *exhaustiveness* conseguiram os melhores valores de RMSD. Isto ocorre porque, embora a função de pontuação do programa AutoDock Vina tenha sido calibrada por seu autor original com foco na reprodução dos valores experimentais de energia livre de ligação (*binding*), a busca mais exaustiva por melhores soluções de *docking* molecular, relacionada a altos valores do parâmetro *exhaustiveness*, aumenta as chances de um bom resultado de *re-docking*, ou seja, um RMSD mais próximo de zero.

Como resultado geral, foi possível observar que ao se utilizar valores mais altos do parâmetro *exhaustiveness* no programa AutoDock Vina, é possível obter soluções de *docking* mais estáveis (de menor energia), e mais precisas (mais próximas do observado nas estruturas cristalográficas, obtidas experimentalmente).

5.5 Vacinologia Reversa

A vacinologia reversa é uma abordagem *in silico* que busca, a partir do genoma do patógeno, identificar antígenos capazes de desencadear uma resposta no sistema imunológico com o intuito de desenvolver uma vacina contra uma determinada doença. Esta abordagem foi inicialmente utilizada na descoberta de uma vacina contra Meningococcus B, onde os esforços convencionais estavam estagnados por décadas (SETTE; RAPPUOLI, 2000).

A utilização de métodos de vacinologia reversa permite que o

desenvolvimento de vacinas ocorra de forma mais rápida, reduzindo o tempo médio de desenvolvimento para apenas 1 ou 2 anos, quando comparado com o tempo médio de desenvolvimento de uma nova vacina utilizando apenas os métodos convencionais, que leva entre 5 e 15 anos (SETTE; RAPPUOLI, 2000). Isto ocorre porque, ao se utilizar esta abordagem de vacinologia reversa *in silico*, é possível identificar e testar praticamente todos os antígenos proteicos que podem ser expressos pelo patógeno, criando um verdadeiro "catálogo" de candidatos a antígenos, aumentando as chances de encontrar antígenos com potencial de se tornar uma vacina (SETTE; RAPPUOLI, 2000). Desta forma, apenas os melhores candidatos são testados experimentalmente, reduzindo os custos (financeiro e tempo) associados a descoberta de uma nova vacina. Acontece que, ao longo deste "catálogo" de antígenos, a combinação de epítopos e alvos imunológicos a serem testados através de métodos *in silico* pode ser bastante elevada, ocasionando uma maior demanda por recursos computacionais.

Durante a execução de um trabalho de colaboração em vacinologia reversa (E SILVA et. al, 2016), cujo objetivo foi identificar candidatos a epítopos com potencial de desencadear uma resposta celular, em humanos, contra diferentes espécies de *Leishmania braziliensis*, houve a necessidade de elaborar uma metodologia para lidar com um grande número de complexos (Epítopo + alvos imunológicos). Como o objetivo deste trabalho em colaboração era de encontrar epítopos com o potencial de desencadear uma resposta celular, avaliou-se a afinidade com que candidatos à epítopos, fragmentos de peptídeo, interagem com estruturas imunológicas conhecidas

como MHC (*Major Histocompatibility Complex*), onde cada um destes epítopos presente no antígeno pode interagir com MHCs de classes distintas. A partir do genoma da *Leishmania braziliensis*, foram preditos 230 epítopos, constituídos por nove resíduos de aminoácido cada, tendo 21 alelos de MHC Classe I como alvo, e também 2.319 epítopos, constituídos por quinze resíduos de aminoácido cada, tendo 12 alelos de MHC Classe II como alvo, totalizando um conjunto de 32.658 complexos (MHCI = 21 x 230, MHCII = 12 x 2.319) a serem testados *in silico*. Sendo assim, optou-se por acrescentar esta funcionalidade de vacinologia reversa ao programa GriDoMol para, desta forma, utilizar o poder computacional da arquitetura distribuída do ambiente de grid computacional para preparar, submeter e analisar os resultados obtidos em cada etapa dos estudos de vacinologia reversa.

A vacinologia reversa com o programa GriDoMol parte do ponto em que o usuário já possui a lista da sequência de aminoácidos de cada um dos peptídeos candidatos a epítopos devidamente selecionados e armazenados em arquivos do tipo FASTA, devidamente caracterizados de acordo com as suas respectivas classes de MHCs.

A lista contendo os códigos PDB dos alelos de MHCs de classes I e II humanos que foram selecionados pode ser observado na Tabela 20. Estes alelos foram preparados utilizando-se o programa Pymol (DELANO, 2004), removendo-se as moléculas de água, ligantes, repetições de resíduos de aminoácidos e dímeros. Além disto, o peptídeo co-cristalizado de cada uma destas estruturas teve o número de aminoácidos dimensionado para ter o

tamanho adequado para cada classe de MHC, sendo 9 aminoácidos para o MHC de Classe I, e 15 aminoácidos para o MHC de Classe II. Os aminoácidos não canônicos foram transformados em alanina.

Tabela 20 – Código PDB das estruturas dos alelos de MHC Classe I e II disponibilizadas no programa GriDoMol.

MHC Class I	2HJL	3C9N	3HCV	3KPP	3L3D	3RL1	3VCL	3VFS	3X11	4F7M	4G8G	4HWZ
	4JQX	4MJ5	4MJI	4NQV	4O2C	4QRR	4QRU	4WU5	4XXC			
MHC Class II	1A6A	1BX2	1H15	1S9V	1UVQ	1YMM	2NNA	2Q6W	3C5J	3LQZ	3PL6	3WEX

Para a preparação e realização dos cálculos de docking molecular, foi utilizado o Rosetta framework (LEAVER-FAY *et al.*, 2011), mais especificamente os módulos FixBB, Relax e FlexPepDock, no modo refinamento (RAVEH; LONDON; SCHUELER-FURMAN, 2010). A suíte de programas do Rosetta framework é bastante utilizado em todo o mundo, inclusive em projetos que utilizam arquiteturas distribuídas, como alguns dos projetos citados na revisão bibliográfica desta tese como, por exemplo, os projetos Foldit (COOPER et al., 2010), Rosetta@Home (Rosetta@Home, 2016), entre outros projetos da plataforma BOINC (BOINC, 2016). A implementação da metodologia para vacinologia reversa no programa GriDoMol dividiu-se em três etapas.

5.5.1 PRIMEIRA ETAPA: CRIAÇÃO DOS COMPLEXOS

Esta primeira etapa da metodologia de vacinologia reversa implementada no programa GriDoMol consiste na criação, estabilização e repontuação (rescore) de cada um dos complexos (epítopos + MHC) que serão utilizados durante a realização dos cálculos de docking molecular.

A geração dos complexos é realizada através do módulo FixBB do framework Rosetta, o qual se encarrega de substituir a sequência de aminoácidos do peptídeo que está co-cristalizado junto com cada alelo (nas estruturas tridimensionais, no formato PDB) pelos aminoácidos do candidato a epítopo (do arquivo FASTA fornecido pelo usuário). Desta forma, o módulo FixBB é utilizado sistematicamente de forma a gerar complexos para todas as combinações entre candidatos a epítopos e alelos de MHC pertencentes a mesma classe alvo destes epítopos. Entretanto, durante testes preliminares identificou-se que alguns dos complexos gerados estavam em conformações instáveis. Isto ocorre porque, como o módulo FixBB não move os átomos da cadeia principal, a conformação em que estes candidatos a epítopos assumiram é a mesma conformação do peptídeo co-cristalizado e, como houve substituição de aminoácidos, nem sempre a conformação observada no peptídeo co-cristalizado será uma conformação estável para estes novos peptídeos candidatos a epítopos.

Assim, com o intuito de estabilizar estes novos complexos, utilizou-se o módulo Relax do framework Rosetta. Este módulo foi utilizado logo após o módulo FixBB, com o intuito de reposicionar o candidato a epítopo em uma

conformação energeticamente mais estável, próximo em sua vizinhança química. Para evitar que o módulo Relax realizasse alterações na estrutura dos alelos de MHC, o arquivo de entrada para a execução do módulo Relax permitiu apenas movimentações e rotações por parte dos aminoácidos do candidato a epítopo, gerando uma estrutura final mais estável para cada complexo. Porém, como a pontuação (score) atribuída a estes complexos leva em consideração a estabilidade energética total de cada complexo como um todo, incluindo a estabilidade individual especifica de cada alelo de MHC, foi necessário utilizar uma função de pontuação que levasse em consideração apenas as interações entre o candidato a epítopo e os aminoácidos de interface do alelo de MHC.

Desta forma, com o intuito de classificar e quantificar apena as interações entre o candidato a epítopo e o alelo de MHC, é necessário utilizar o módulo FlexPepDock para realizar o procedimento de repontuação (rescore) dos complexos recém criados para obter os valores de Interface Score (isc), cujo valor é a soma das contribuições energéticas das interações entre os aminoácidos de interface entre o alelo e o epítopo. É importante ressaltar que, embora o objetivo do FlexPepDock seja realizar os cálculos de docking molecular, neste momento o FlexPepDock é usado apenas para realizar o procedimento de rescore das estruturas dos complexos gerados, não havendo quaisquer modificações nas estruturas obtidas anteriormente através do módulo Relax.

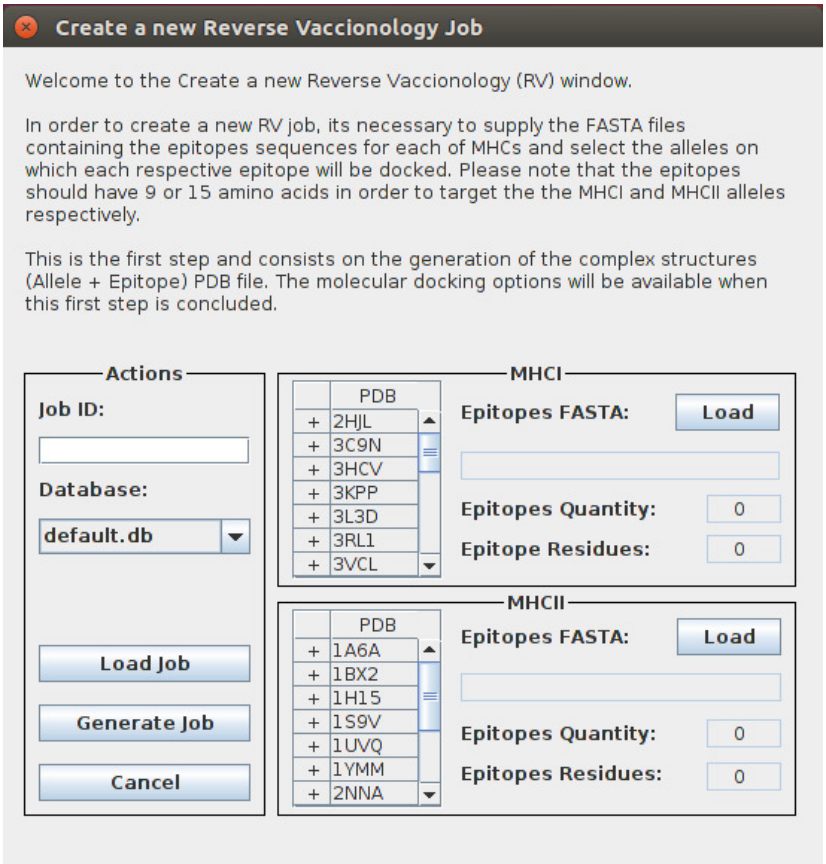


Figura 27 – Tela de criação do job para a primeira etapa do estudo de vacinologia reversa.

A interface de criação do job para a primeira etapa de vacinologia reversa no programa GriDoMol (Figura 27) é subdividida em três seções. Nas seções *MHCI* e *MHCII* o usuário pode carregar quais candidatos a epítopos deseja utilizar nos cálculos de docking para cada classe de MHC através de um arquivo do tipo FASTA, o qual contém a sequência de aminoácidos de cada candidato a epítopo, e quais estruturas de alelos que se deseja utilizar em seus cálculos. Na aba *Actions*, o usuário pode carregar um job que foi criado anteriormente com o intuito de realizar modificações tais como, por exemplo, selecionar outros alelos de MHCs ou utilizar outros epítopos, através do botão

Load Job ou criar um novo job, através do botão *Generate Job*, cujo identificador será o nome inserido no campo *Job ID*. Além disto, ainda na seção *Actions*, observa-se que há um campo denominado *Database* cuja função seria alternar entre os bancos de dados contendo as estruturas, previamente preparadas, de outros alelos de MHCs Classe I e II. Porém, atualmente apenas o banco de dados padrão (default.db) está disponível e outros bancos de dados poderão ser implementadas futuramente no programa.

Após escolher a opção *Generate Job*, o programa GriDoMol cria automaticamente uma estrutura de subpastas, a partir de uma pasta inicial com o nome do job, e gera todos os arquivos de entrada necessários (ver Tabela 21) para submeter ao ambiente de grid computacional este primeiro conjunto de procedimentos que consiste na execução dos módulos FixBB, Relax e FlexPepDock (rescore).

Tabela 21 – Arquivos gerados pelo programa GriDoMol, para cada complexo, com o intuito de submeter ao grid computacional os procedimentos iniciais do estudo de vacinologia reversa.

Arquivo	Função
Flags_fixbb	Arquivo com os parâmetros de execução do módulo FixBB.
Flags_relax	Arquivo com os parâmetros de execução do módulo Relax.
Flags_rescore	Arquivo com os parâmetros de execução do módulo FlexPepDock (rescore).
Resfile	Arquivo utilizado pelo FixBB para determinar cada aminoácido que será substituído e qual tipo de aminoácido entrará no lugar.
Movemap	Utilizado pelo módulo Relax para determinar quais aminoácidos podem ser movidos. Todos os aminoácidos do alelo foram bloqueados (BBCHI NO), permitindo apenas a movimentação e rotação dos aminoácidos do epítopo.

5.5.2 SEGUNDA ETAPA: FILTROS E CÁLCULO DE DOCKING MOLECULAR

Como a demanda computacional para realizar o cálculo de docking molecular para todas as combinações possíveis de epítopos e alelos de MHCs costuma ser bastante elevada, utilizou-se a estratégia de filtrar e classificar os resultados obtidos na etapa anterior de forma a reduzir a demanda computacional ao buscar os melhores candidatos a epítopos. Desta forma, pode-se avaliar todos os complexos baseando-se, essencialmente, em dois critérios.

O primeiro critério de avaliação é a ocorrência de “janelas”, onde a sequência de aminoácidos de um candidato a epítopo para os alelos de MHC Classe I (9 aminoácidos) aparece inserido em algum trecho da sequência de aminoácidos de um candidato a epítopo para os alelos de MHC Classe II (15 aminoácidos). Acredita-se que se um epítopo de 15 aminoácidos tem uma boa afinidade pelos alelos de MHC Classe II e sua “janela” de 9 aminoácidos possui uma boa afinidade pelos alelos de MHC Classe I, este “par” possui um potencial maior de apresentar propriedades imunogênicas significativas. Sendo assim, o programa GriDoMol procura ocorrências onde um epítopo de 9 aminoácidos é uma janela em um epítopo de 15 aminoácidos e forma “pares” de epítopos MHCs Classe I e II.

O segundo critério de avaliação leva em consideração a soma da frequência com que um candidato a epítopo é encontrado na lista contendo X% das melhores soluções de docking em alelos de suas respectivas classes de MHCs, sendo o valor de X determinado pelo usuário na interface do programa

GriDoMol. Se o candidato a epítopo em questão aparece na lista das X% melhores soluções de docking para um alelo, soma-se um a sua frequência. Caso o epítopo não apareça na lista, nenhuma operação é realizada. Desta forma, pode-se avaliar a afinidade média de cada epítopo ao longo de seus respectivos alelos de MHCs. O objetivo de classificar os epítopos baseando-se na soma de suas frequências é que, quando se busca por um candidato a epítopo que possa apresentar uma boa resposta imunológica, é importante que este epítopo possua uma boa afinidade pelo maior número possível de alelos de MHCs e, assim, ter o maior efeito imunogênico. O fluxograma do algoritmo implementado no programa GriDoMol para contabilizar a soma das frequências de cada candidato a epítopo utilizando, como exemplo, o valor de 30% como ponto de corte das melhores soluções de docking, para cada alelo, pode ser observado na Figura 28.

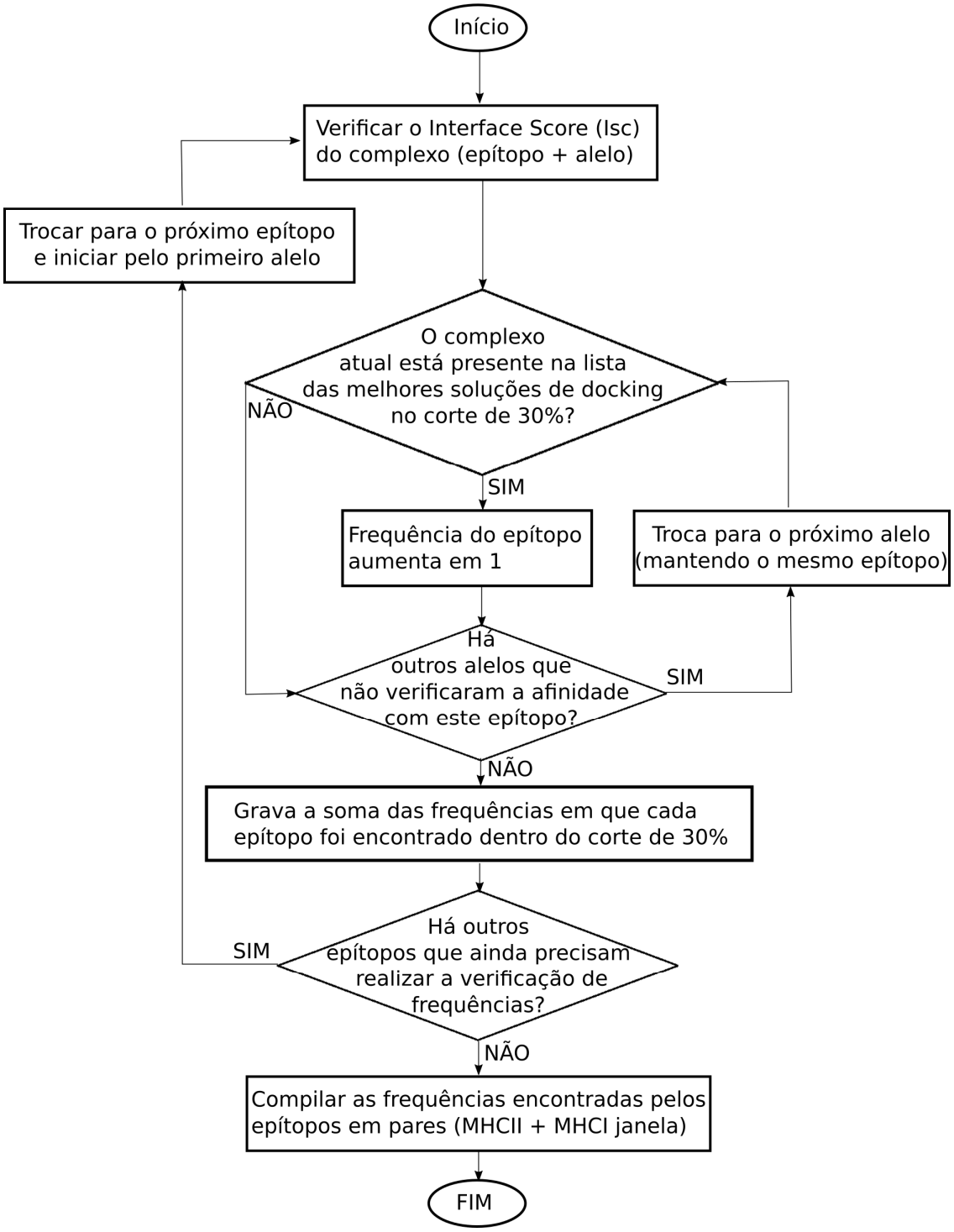


Figura 28 – Fluxograma do algoritmo de filtragem implementado no programa GriDoMol para medir a frequência com que cada epítopo aparece entre os 30% dos epítopos mais estáveis para cada alelo (adaptado de E SILVA *et al.*, 2016).

Observa-se na Figura 29 a interface do programa GriDoMol para a criação do job desta segunda etapa do estudo de vacinologia reversa. Por se tratar de um exemplo real e por questões de confidencialidade, as sequencia de aminoácidos destes epítopos foram protegidas.

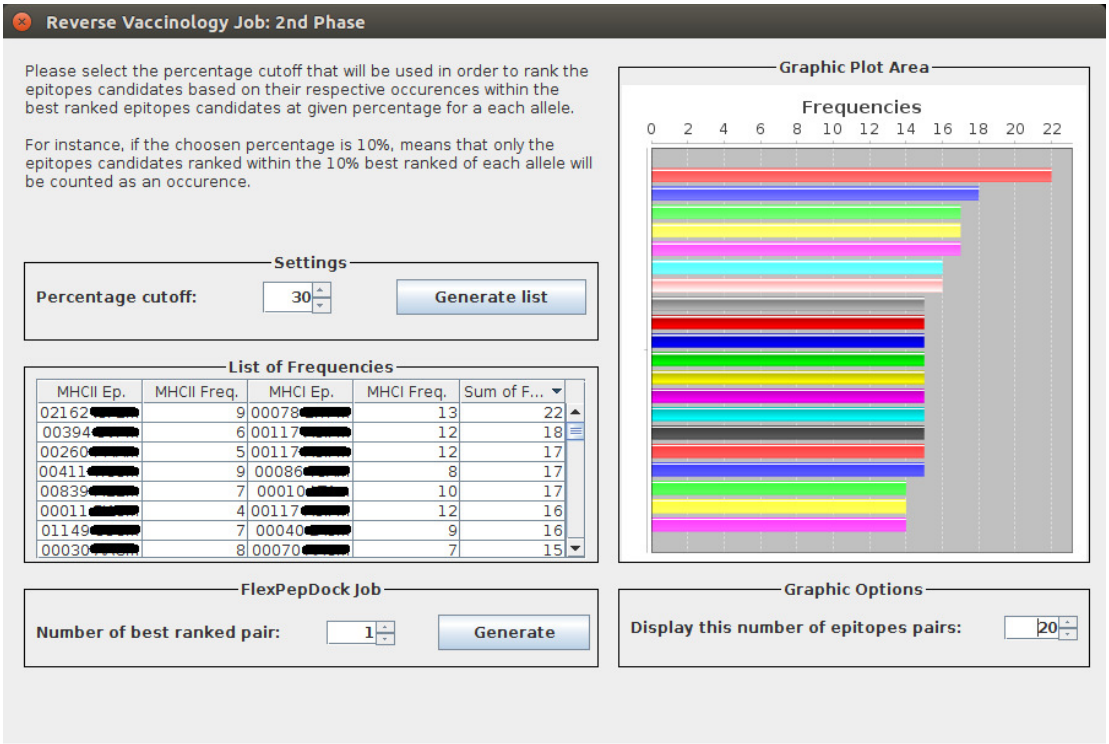


Figura 29 – Interface de criação do job para a segunda etapa do estudo de vacinologia reversa.

No campo *Settings,* o usuário pode escolher qual a porcentagem que será utilizada como ponto de corte para as melhores soluções de *docking* em cada alelo de MHC. Este valor dependerá de um equilíbrio e não pode ser baixo demais, a ponto de não registrar muitas frequências, e nem alto demais, a ponto de permitir que soluções de *docking* não muito boas sejam contadas na

frequência. Após utilizar o botão *Generate List*, a lista das frequências individuais de cada epítopo e a soma da frequência dos pares serão calculadas e exibidas na seção *List of Frequencies*. O usuário pode a qualquer momento gerar uma nova lista ao alterar o valor da porcentagem de corte e clicar novamente no botão *Generate List*. Na seção *Graphic Plot Area*, o usuário poderá analisar, através de um gráfico de barras, a soma das frequências dos pares de epítopos. A quantidade de pares que serão exibidos no gráfico pode ser definido na seção *Graphic Options*. Na seção *FlexPepDock Job* o usuário pode definir quantos pares de epítopos serão utilizados nos cálculos de docking molecular com o módulo FlexPepDock do Rosetta, utilizando as configurações padrão deste módulo, obtendo 100 soluções de docking em cada cálculo individual de docking molecular. Ao clicar no botão *Generate*, o programa GriDoMol cria o segundo job de vacinologia reversa, permitindo a submissão destes cálculos de docking molecular ao ambiente de grid computacional para a execução em paralelo.

5.5.3 ÚLTIMA ETAPA: CÁLCULO DE DOCKING COM MAIOR NÚMERO DE SOLUÇÕES

Esta ultima etapa consiste em realizar cálculos de docking molecular entre os candidatos a epítopos e os alelos de MHCs buscando um número maior de soluções (definido pelo usuário), aumentando as chances de encontrar soluções mais estáveis, com o intuito de refinar os resultados obtidos na etapa anterior.

Os resultados de *docking* molecular para os pares de epítopos selecionados na etapa anterior são tabelados e descritos nesta terceira e última etapa do estudo de vacinologia reversa com o programa GriDoMol (Figura 30). Com o intuito de prover mais dados para o usuário analisar os resultados, além dos valores de Interface Score (Isc) da melhor solução de docking para cada complexo, outras duas pontuações também foram incluídas para a análise na interface gráfica, o *Interface Buried Surface Area* (Ibsa) e o Interface Hydrogen Bonds (Ihb). O *Interface Buried Surface Area* representa o tamanho da interface do alelo de MHC que interage com o candidato a epítopo enquanto o *Interface Hydrogen Bonds* representa o número de ligações de hidrogênio formadas entre o epítopo e a interface do alelo de MHC.

Como o objetivo principal é encontrar pares de epítopos que possuam uma boa afinidade pelo maior número possível de alelos de MHCs, incluiu-se os valores médios obtidos por cada par de epítopos, ao longo de seus respectivos MHCs, nestas três pontuações (Isc, Ibsa e Ihb). Um exemplo mostrando a média (Avg) de Ihb pode ser visto na Figura 8b.

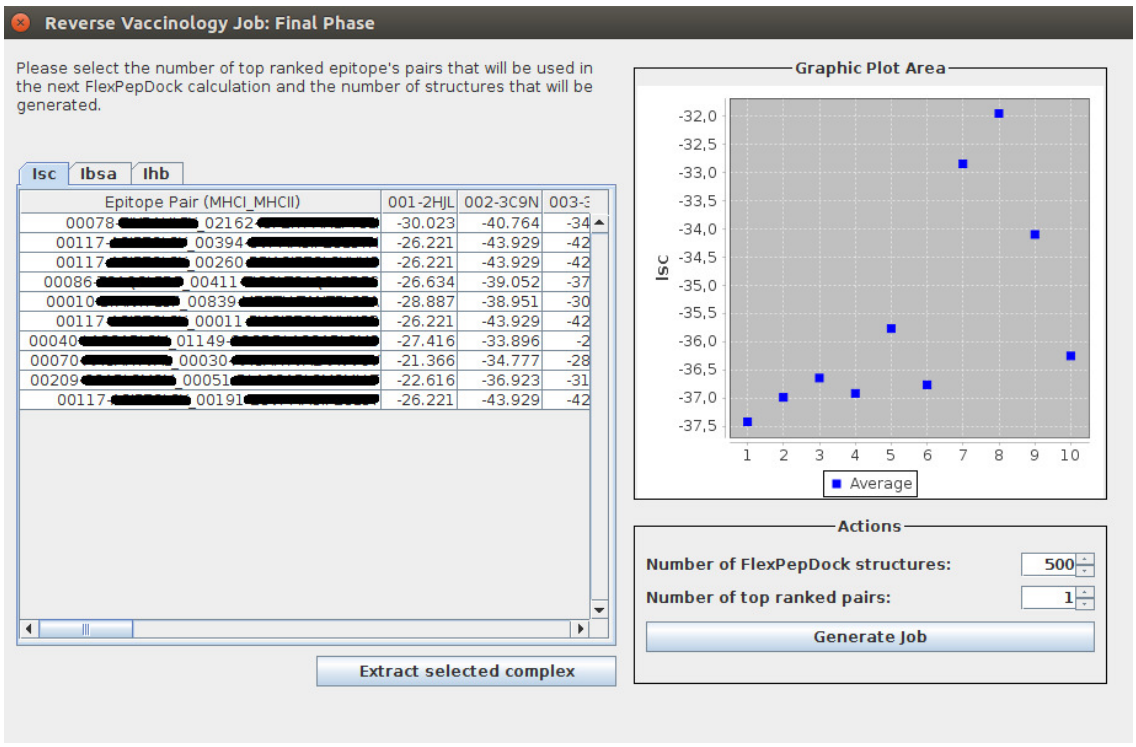


Figura 30 – Interface de criação do job para a terceira e última etapa do estudo de vacinologia reversa.

A interface de criação do programa GriDoMol para a criação do job desta terceira etapa pode ser observado na Figura 30. As abas Isc, Ibsa e Ihb alternam qual tipo de pontuação é exibida na tabela e no gráfico das médias da seção *Graphic Plot Area* (Figuras 30a e 30b). Ao selecionar um dos resultados e apertar o botão *Extract Selected Complex*, pode-se extrair e salvar o arquivo PDB contendo a solução de docking em um local a ser escolhido pelo usuário. Na seção *Actions*, o usuário pode definir a quantidade de soluções de *docking* obtidas durante cada cálculo individual de *docking* molecular e quantos dos melhores pares de epítopos, baseados na média de Isc de cada candidato à epítopo e seus respectivos alvos, farão parte dos cálculos de *docking* molecular

desta etapa. O botão *Generate Job* o programa GriDoMol cria este último job de vacinologia reversa, permitindo a submissão destes cálculos de *docking* molecular, obtendo um número maior de soluções, ao ambiente de grid computacional para a execução em paralelo. É importante ressaltar que esta última etapa pode ser repetida sistematicamente, permitindo que o usuário refine seus resultados obtidos nos cálculos de *docking* molecular, tantas vezes quantas ele ache necessário.

5.5.4 VALIDAÇÃO EXPERIMENTAL DOS EPÍTOPOS PREDITOS

Com o intuito de validar os resultados obtidos através desta metodologia de vacinologia reversa utilizada pelo programa GriDoMol, os 10 melhores candidatos a epítopos de 15 aminoácidos encontrados pelo programa GriDoMol foram sintetizados e testados experimentalmente (E SILVA et al., 2016). Testes foram realizados para mensurar a capacidade de indução da proliferação das células mononucleares do sangue periférico (PBMC) derivadas de pacientes humanos curados após tratamentos. Embora todos os candidatos a epítopos tenham sido capazes de estimular a proliferação de muitos PBMC destes pacientes, os candidatos a epítopos 1, 3, 5, 6 e 7 apresentaram diferenças estatísticas significativas quando comparados entre os pacientes tratados e os voluntários de controle (E SILVA et al., 2016).

Sendo assim, ao se utilizar a metodologia de vacinologia reversa implementada no programa GriDoMol para filtrar e distribuir os cálculos de docking molecular entre candidatos a epítopos e suas respectivas classes de alelos de MHCs no ambiente de grid computacional, pôde-se selecionar bons

candidatos a epítopos. Vários estudos estão sendo conduzidos pelo grupo experimental, em modelos in vitro e in vivo, no intuito de elucidar cada vez mais o comportamento imunológico destes epítopos selecionados pela abordagem in silico de vacinologia reversa usando o programa GriDoMol.

6 CONCLUSÕES

Apresentou-se, neste presente trabalho, o desenvolvimento das versões 2.0 (em C++) e 3.0 (em Java) do programa GriDoMol, uma plataforma unificada para execução de cálculos de *docking* molecular em ambiente distribuído do tipo *grid* computacional, visando alta performance.

Uma das principais novidades na versão 2.0 do programa GriDoMol é o suporte a opções avançadas dos programas AutoDock e AutoDock Vina. Desta forma, é possível alterar os parâmetros de busca destes programas e encontrar melhores soluções de *docking* durante os cálculos, ao custo de um aumento de demanda computacional (tempo). Além disto, nesta versão 2.0 do programa GriDoMol, adicionou-se a possibilidade de escolher, individualmente, qual programa de *docking* molecular realizará cada cálculo, enquanto que antes (versão 1.0) só era possível selecionar um programa de *docking* molecular para executar todo o conjunto de cálculos de *docking* molecular.

A versão 3.0 do programa GriDoMol, desenvolvida na linguagem de programação Java, foi desenvolvida de forma modular e em camadas com o intuito de facilitar a integração de novos componentes como, por exemplo, programas de docking molecular, middlewares de ambiente de grid computacional e novas metodologias, como foi o caso da adição da abordagem de vacinologia reversa, implementada durante a etapa final do desenvolvimento do programa GriDoMol 3.0, sem a necessidade de reescrever as funções principais do programa. Futuramente, novas formas de acesso remoto via Web e Web Mobile poderão ser implementadas ao programa com um esforço de

desenvolvimento menor, uma vez que a parte lógica do programa está completamente separada das interfaces gráficas.

O ganho de desempenho ao se utilizar um ambiente de *grid* computacional para execução de cálculos de *docking* molecular, em comparação ao processamento sequencial realizado em um só computador, ocorre principalmente devido à simultaneidade dos processos sendo executados em várias máquinas, em paralelo. Enquanto em um único computador, todo o processamento é realizado sequencialmente, onde o tempo gasto na execução sequencial destas fases é sempre cumulativo, ao passo que em um ambiente de *grid* computacional estes processamentos ocorrem simultaneamente, o que faz com que o tempo total gasto na execução em paralelo destes cálculos seja significativamente reduzido.

É importante salientar as vantagens alcançadas com o desenvolvimento de um programa específico voltado para esta finalidade (utilização de um grid computacional para a distribuição de cálculos de *docking* molecular). Pode-se imaginar uma situação, na ausência do GriDoMol, onde o usuário dispusesse de diversos computadores (8 máquinas, por exemplo) em seu laboratório para realizar, digamos, o mesmo conjunto de tarefas apresentado neste trabalho, ou seja, 213 cálculos individuais de *docking* molecular. Neste caso, este usuário precisaria fazer uma divisão equitativa (213/8), que daria cerca de 26 a 27 cálculos individuais por máquina. Além de precisar submeter manualmente os cálculos em cada máquina, mesmo utilizando-se um script (ou "arquivo de lote", em *batch*) para isto, como a demanda computacional de cada cálculo

individual de *docking* molecular pode ser muito diferente, inclusive em ordens de grandeza de tempo gasto, o que aconteceria, provavelmente, seria uma distribuição não equitativa de esforço computacional. Em outras palavras, a máquina que recebesse os cálculos de *docking* mais rápidos (de menor demanda computacional), executaria a tarefa de calcular os cerca de 26 a 27 cálculos em uma fração do tempo que outra máquina que recebesse os cálculos de *docking* mais demorados (de maior demanda computacional), isto considerando-se um grid computacional homogêneo como o que foi utilizado neste trabalho de pesquisa. Se, adicionalmente, o grid fosse heterogêneo, ou seja, composto por máquinas de diferentes configurações de hardware e software, seria ainda menos previsível para o usuário a demanda computacional por máquina e uma distribuição minimamente equitativa das tarefas seria literalmente impossível de se fazer, pois a demanda computacional dos cálculos de *docking* só é verdadeiramente conhecida depois que eles são concluídos. O GriDoMol representa uma vantagem estratégica neste sentido, pois além de servir como ferramenta para o usuário submeter os cálculos no grid computacional de maneira automática, pois é o próprio sistema que acessa as outras máquinas disponíveis no grid computacional, a distribuição das *tasks* ocorre sob demanda (a lista de *tasks* individuais que ainda precisam ser executadas pelo sistema) e sob oferta (a lista de máquinas do grid computacional que vão finalizando *tasks* anteriores e tornando-se novamente disponíveis para executar as *tasks* que ainda precisam ser calculadas), maximizando-se assim, a performance computacional e

minimizando o tempo total que o *job* (conjunto de *tasks*) precisa para ser executado nos computadores.

Entretanto, há um limite para o ganho de desempenho obtido através da aplicação de ambientes de *grid* computacional para execução de *docking* molecular em larga escala (*virtual screening*). Durante os últimos cálculos de um *job*, onde há mais computadores disponíveis do que cálculos para realizar, ocorre uma ociosidade por parte dos computadores excedentes. Desta forma, em uma execução de *virtual screening* típica, com moléculas extremamente diferentes entre si, é possível que o cálculo mais demorado seja o último, por exemplo, maximizando o tempo ocioso por parte dos outros computadores integrantes do ambiente de *grid* computacional. Com o intuito de amenizar esta situação, a nova versão (3.0) do programa GriDoMol já permite que sejam enviados vários *jobs* para uma lista de espera e, desta forma, os computadores que estariam ociosos, a espera do *job* anterior terminar, agora podem iniciar a execução de um novo *job* enquanto as últimas *tasks* do *job* anterior ainda estão sendo finalizadas, efetivamente minimizando a ociosidade nesta arquitetura distribuída.

Além de permitir cálculos de docking molecular voltados a estudos de *virtual screening*, uma nova metodologia foi desenvolvida e implementada no programa GriDoMol para contemplar casos de estudos em vacinologia reversa, onde o usuário pode avaliar (in silico) candidatos a epítopos em vários alelos de MHC Classe I e II humanos, com o objetivo de desenvolver uma vacina contra uma determinada doença. Esta metodologia foi concebida com o intuito

de possibilitar o uso do ambiente de grid computacional para solucionar um problema de alta demanda computacional que ocorreu em um estudo de vacinologia reversa onde as combinações de candidatos a epítopos e alelos de MHCs humanos ultrapassaram 32 mil complexos (alelo + epítopo) a serem testados (E SILVA et al., 2016). A validação desta metodologia ocorreu neste mesmo estudo através de testes experimentais onde os melhores candidatos a epítopos encontrados pelo programa GriDoMol foram reconhecidos pelo sistema imunológico humano, criando uma resposta imune protetora (E SILVA et al., 2016).

Os testes de desempenho demonstraram a capacidade do programa GriDoMol em criar, submeter e gerenciar *jobs* (conjunto de cálculos de *docking*) no ambiente de *grid* computacional, formado pela plataforma OurGrid, em conjunto com os programas de *docking* molecular AutoDock e AutoDock Vina para executar cálculos de *docking* molecular em larga escala de forma otimizada, utilizando ordenamento predito, com maior precisão e em menos tempo, quando comparado com a execução sequencial deste mesmo conjunto de cálculos em um único computador, atingindo os principais objetivos desta tese de doutorado.

7 PERSPECTIVAS

Espera-se implementar, futuramente, novas funcionalidades no programa GriDoMol, munindo-o de novas funcionalidades, como por exemplo:

- Permitir a interação entre o usuário e o sistema GriDoMol através de acesso remoto via *Web* e *Web Mobile*;
- Possibilitar a realização de refinamento dos resultados, como, por exemplo, utilizar funções de *score* alternativas para as soluções de *docking* (CASE et al., 2005) e otimização de geometrias (PURANEN; VAINIO; JOHNSON, 2010; HESS et al., 2008);
- Permitir a realização de análises detalhadas dos resultados de *docking*, utilizando ferramentas disponíveis na máquina do usuário, tais como o PyMol (DELANO, 2009), para a visualização molecular, e o Binana (DURRANT; MCCAMMON, 2011), para analisar as características das interações intermoleculares que ocorrem entre o receptor (alvo) e o ligante;
- Possibilitar que o usuário escolha, diretamente de bancos de dados localizados na internet, as moléculas que serão utilizadas na realização do *docking* molecular, como, por exemplo, através de códigos PDB (RCSB Protein Data Bank, 2016) e outros bancos de dados.

REFERÊNCIAS

ABREU, R. M. V.; FROUFE, H. J. C.; QUEIROZ, M.; FERREIRA, I. MOLA: a bootable, self-configuring system for virtual screening using AutoDock4/Vina on computer clusters. **Journal of Cheminformatics**, v. 2, 2010. ISSN 1758-2946.

ADT (2016). Disponível em: <http://autodock.scripps.edu/resources/adt>. Acessado em Novembro de 2016.

ANDRADE, N.; CIRNE, W.; BRASILEIRO, F.; ROISENBERG, P. OurGrid: An approach to easily assemble grids with equitable resource sharing. In: FEITELSON, D.;RUDOLPH, L., et al (Ed.). **Job Scheduling Strategies for Parallel Processing**. Berlin: Springer-Verlag Berlin, v.2862, 2003. p.61-86. (Lecture Notes in Computer Science). ISBN 0302-9743

Android FAQ BOINC (2016). Disponível em: <http://boinc.berkeley.edu/wiki/Android_FAQ>. Acessado em Novembro de 2016.

ANGLANO, C.; CANONICO, M.; GUAZZONE, M. The ShareGrid Peer-to-Peer Desktop Grid: Infrastructure, Applications, and Performance Evaluation. **Journal of Grid Computing**, v. 8, n. 4, p. 543–570, 4 jun. 2010.

BERMAN, H.M.; WESTBROOK, J.; FENG, Z.; GILLILAND, G.; BHAT, T.N.; WEISSIG, H.; SHINDYALOV, I.N.; BOURNE, P.E.; The Protein Data Bank. **Nucleic Acids Research**, 2000. 28: 235-242.

AutoDock Vina (2016). Disponível em: <http://vina.scripps.edu/>. Acessado em Novembro de 2016.

BIENFAIT, B.; ERTL, P. JSME: a free molecule editor in JavaScript. **Journal of cheminformatics**, v. 5, p. 24, jan. 2013.

BOINC (2016). Disponível em: <http://boinc.berkeley.edu/>. Acessado em Novembro de 2016.

BOINC Stats (2016). Disponível em: <http://boincstats.com/>. Acessado em Novembro de 2016.

BUYYA, R.; BRANSON, K.; GIDDY, J.; ABRAMSON, D. **The virtual laboratory: A toolset for utilising the world-wide grid to design drugs**. Los Alamitos: Ieee Computer Soc, 2002. 278-279 ISBN 0-7695-1582-7.

CASE, D. A.; CHEATHAM, T. E.; DARDEN, T.; GOHLKE, H.; LUO, R.; MERZ, K. M.; ONUFRIEV, A.; SIMMERLING, C.; WANG, B.; WOODS, R. J. The Amber biomolecular simulation programs. **Journal of Computational Chemistry,** v. 26, n. 16, p. 1668-1688, Dec 2005. ISSN 0192-8651.

CIRNE, W. (2002). **Grids Computacionais Arquiteturas, Tecnologias e Aplicações**. In Proceedings of the 3rd Workshop on High Performance Computing Systems (WSCAD'2002), Vitória, Brazil.

COOPER, SETH; KHATIB, FIRAS; TREUILLE, ADRIEN; BARBERO, JANOS; LEE, JEEHYUNG; BEENEN, MICHAEL; LEAVER-FAY, ANDREW; BAKER, DAVID; POPOVIĆ, ZORAN; PLAYERS, F. Predicting protein structures with a multiplayer online game. **Nature**, v. 466, n. 7307, p. 756–60, 5 ago. 2010.

CUDA (2016). Disponível em <http://www.nvidia.com/object/cuda_home_new.html>. Acessado em Novembro de 2016.

DA SILVA, M; NESMACHNOW, S; GEIER, M; MOCSKOS, E; ANGIOLINI, J; LEVI, V; CRISTOBAL, A. Efficient Fluorescence Microscopy Analysis over a Volunteer Grid/Cloud Infrastructure. **HIGH PERFORMANCE COMPUTING, CARLA 2014**, v. 485, p. 113–127, 2014.

DELANO, W. L. Use of PYMOL as a communications tool for molecular science. **Abstracts of Papers of the American Chemical Society,** v. 228, p. U313-U314, Aug 2004. ISSN 0065-7727.

DELANO, W. L. PyMOL molecular viewer: Updates and refinements. **Abstracts of Papers of the American Chemical Society,** v. 238, Aug 2009. ISSN 0065-7727.

DESJARLAIS, R. L.; SEIBEL, G. L.; KUNTZ, I. D.; FURTH, P. S.; ALVAREZ, J. C.; DEMONTELLANO, P. R. O.; DECAMP, D. L.; BABE, L. M.; CRAIK, C. S. STRUCTURE-BASED DESIGN OF NONPEPTIDE INHIBITORS SPECIFIC FOR THE HUMAN IMMUNODEFICIENCY VIRUS-1 PROTEASE. **Proceedings of the National Academy of Sciences of the United States of America,** v. 87, n. 17, p. 6644-6648, Sep 1990. ISSN 0027-8424.

Discovering Dengue Drugs (2016). Disponível em: <http://www.worldcommunitygrid.org/ research/dddt2/overview.do>. Acessado em Novembro de 2016.

DockThor (2016). Disponível em: <http://www.dockthor.lncc.br/>. Acessado em Novembro de 2016.

DOMINGUEZ, C.; BOELENS, R.; BONVIN, A. HADDOCK: A protein-protein docking approach based on biochemical or biophysical information. **Journal of the American Chemical Society,** v. 125, n. 7, p. 1731-1737, Feb 2003. ISSN 0002-7863.

Drug Search for Leishmaniasis (2016). Disponível em: <http://www.worldcommunitygrid.org/ research/dsfl/overview.do>. Acessado em Novembro de 2016.

DURRANT, J. D.; MCCAMMON, J. A. BINANA: A novel algorithm for ligand-binding characterization. **Journal of Molecular Graphics & Modelling,** v. 29, n. 6, p. 888-893, Apr 2011. ISSN 1093-3263.

E SILVA, R.F.; FERREIRA, L.F.G.R; HERNANDES, M.Z.; DE BRITO, E.M.F.; DE OLIVEIRA, O.C.; DA SILVA, A.A.; DE-MELO-NETO, O.P.; REZENDE, A.M.; PEREIRA, V.R.A. Combination of In Silico Methods in the Search for Potential CD4+ and CD8+ T Cell Epitopes in the Proteome of Leishmania braziliensis. **Frontiers in Immunology**, v. 7, p. 327 2016. ISSN 1664-3224.

Eclipse (2016). Disponível em: <https://www.eclipse.org/>. Acessado em Novembro de 2016.

FERREIRA, L.F.G.R. *Desenvolvimento e implementação de software para aplicação de grids computacionais em modelagem para inovação terapêutica*. 2013. 122 f. Dissertação (Mestrado em Inovação Terapêutica) – Universidade Federal de Pernambuco, Recife. 2013.

FightAIDS@Home (2016). Disponível em: <http://www.worldcommunitygrid.org/research/fahb/overview.do>. Acessado em Novembro de 2016.

FISCHER, E. (1894). **"Einfluss der Configuration auf die Wirkung der Enzyme"**. Ber. Dt. Chem. Ges. 27: 2985–2993.

Folding@Home (2016). Disponível em: <http://folding.stanford.edu/>. Acessado em Novembro de 2016.

FOLDIT Portal (2016). Disponível em: <http://fold.it/portal/>. Acessado em Novembro de 2016.

GELDENHUYS, W. J.; GAASCH, K. E.; WATSON, M.; ALLEN, D. D.; VAN DER SCHYF, C. J. Optimizing the use of open-source software applications in drug discovery. **Drug Discov Today,** v. 11, n. 3-4, p. 127-32, Feb 2006. ISSN 1359-6446

GO Fight Against Malaria (2016). Disponível em: <http://www.worldcommunitygrid.org/research/gfam/overview.do>. Acessado em Novembro de 2016.

GOODSELL, D. S.; MORRIS, G. M.; OLSON, A. J. Automated docking of flexible ligands: Applications of AutoDock. **Journal of Molecular Recognition,** v. 9, n. 1, p. 1-5, Jan-Feb 1996. ISSN 0952-3499.

HANWELL, MARCUS D; CURTIS, DONALD E; LONIE, DAVID C; VANDERMEERSCH, TIM; ZUREK, EVA; HUTCHISON, G. R. Avogadro: an advanced semantic chemical editor, visualization, and analysis platform. **Journal of cheminformatics**, v. 4, n. 1, p. 17, jan. 2012.

Help Fight Childhood Cancer (2016). Disponível em: <http://www.worldcommunitygrid.org/research/hfcc/overview.do>. Acessado em Novembro de 2016.

HESS, B.; KUTZNER, C.; VAN DER SPOEL, D.; LINDAHL, E. GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation. **Journal of Chemical Theory and Computation,** v. 4, n. 3, p. 435-447, Mar 2008. ISSN 1549-9618.

Influenza Antiviral Drug Search (2016). Disponível em <http://www.worldcommunitygrid.org/research/flu1/overview.do>. Acessado em Novembro de 2016.

IRTAZA, A.; JAFFAR, M. A.; MAHMOOD, M. T. Semantic Image Retrieval in a Grid Computing Environment Using Support Vector Machines. **The Computer Journal**, v. 57, n. 2, p. 205–216, 22 ago. 2013.

Java (2016). Disponível em: <http://www.java.com/>. Acessado em Novembro de 2016.

JIANG, X. H.; KUMAR, K.; HU, X.; WALLQVIST, A.; REIFMAN, J. DOVIS 2.0: an efficient and easy to use parallel virtual screening tool based on AutoDock 4.0. **Chemistry Central Journal,** v. 2, Sep 2008. ISSN 1752-153X.

KHATIB, FIRAS; COOPER, SETH; TYKA, MICHAEL D; XU, KEFAN; MAKEDON, ILYA; POPOVIC, ZORAN; BAKER, DAVID; PLAYERS, F. Algorithm discovery by protein folding game

players. **Proceedings of the National Academy of Sciences of the United States of America**, v. 108, n. 47, p. 18949–53, 22 nov. 2011.

KOSHLAND, D. E. APPLICATION OF A THEORY OF ENZYME SPECIFICITY TO PROTEIN SYNTHESIS. **Proceedings of the National Academy of Sciences of the United States of America,** v. 44, n. 2, p. 98-104, 1958. ISSN 0027-8424.

KOZAKOV, DIMA; BRENKE, RYAN; COMEAU, STEPHEN R; VAJDA, S. PIPER: an FFT-based protein docking program with pairwise potentials. **Proteins**, v. 65, n. 2, p. 392–406, 1 nov. 2006.

LEAVER-FAY A.; TYKA, M.; LEWIS, S.M.; LANGE, O.F.; THOMPSON, J.; JACAK, R.; ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules. **Methods Enzymol,** 487:545–574, 2011.

LENGAUER, T.; RAREY, M. Computational methods for biomolecular docking. **Current Opinion in Structural Biology,** v. 6, n. 3, p. 402-406, Jun 1996. ISSN 0959-440X.

Linpack (2016). Disponível em <https://software.intel.com/en-us/articles/intel-mkl-benchmarks-suite>. Acessado em Novembro de 2016.

Linux Deploy (2016). Disponível em <https://github.com/meefik/linuxdeploy>. Acessado em Novembro de 2016.

LNCC (2016). Disponível em <http://www.lncc.br/frame.html>. Acessado em Novembro de 2016.

LIU, ZHIHAI; LI, YAN; HAN, LI; LI, JIE; LIU, JIE; ZHAO, ZHIXIONG; NIE, WEI; LIU, YUCHEN; WANG, R. PDB-wide collection of binding data: current status of the PDBbind datab**ase. Bioinformatics (Oxford, Eng**land), v. 31, n. 3, p. 405–12, 1 fev. 2015.

Marvin JS (2016). Disponível em <https://www.chemaxon.com/products/marvin/marvin-js/>. Acessado em Novembro de 2016.

MCINTOSH-SMITH, S; PRICE, J; SESSIONS, R; IBARRA, A. High performance in silico virtual drug screening on many-core processors. **International Journal of High Performance Computing Applications**, p. 1094342014528252–, 9 abr. 2014.

MCGEE, P. (2005). **Modeling success with in silico tools.** Drug Discovery & Development. 8, 24.

MolCal (2016). Disponível em: <http://molcalc.org/>. Acessado em Novembro de 2016.

MORRIS, G. M.; HUEY, R.; LINDSTROM, W.; SANNER, M. F.; BELEW, R. K.; GOODSELL, D. S.; OLSON, A. J. AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility. **Journal of Computational Chemistry,** v. 30, n. 16, p. 2785-2791, Dec 2009. ISSN 0192-8651.

MPICH (2016). Disponível em: <https://www.mpich.org/>. Acessado em Novembro de 2016.

Outsmart Ebola Together (2016). Disponível em: <http://www.worldcommunitygrid.org/research/oet1/overview.do>. Acessado em Novembro de 2016.

OpenCL (2016). Disponível em: <https://www.khronos.org/opencl/>. Acessado em Novembro de 2016.

OpenFire (2016). Disponível em: <http://www.igniterealtime.org/projects/openfire/>. Acessado em Novembro de 2016.

OpenZika (2016). Disponível em: <https://www.worldcommunitygrid.org/research/zika/details.do>. Acessado em Novembro de 2016.

PURANEN, J. S.; VAINIO, M. J.; JOHNSON, M. S. Accurate Conformation-Dependent Molecular Electrostatic Potentials for High-Throughput In Silico Drug Discovery. **Journal of Computational Chemistry,** v. 31, n. 8, p. 1722-1732, Jun 2010. ISSN 0192-8651.

QT (2016). Disponível em: <http://www.qt.io/>. Acessado em Novembro de 2016.

RAVEH, B.; LONDON, N.; SCHUELER-FURMAN, O.; Sub-angstrom modeling of complexes between flexible peptides and globular proteins. **Proteins: Structure, Function and Bioinformatics**, v. 78, n. 9, p 2029–2040, 2010.

RCSB Protein Data Bank (2016). Disponível em: <http://www.rcsb.org>. Acessado em Novembro de 2016.

ROHL, C; STRAUSS, C; MISURA, K; BAKER, D. Protein structure prediction using rosetta. **NUMERICAL COMPUTER METHODS, PT D**, v. 383, 2004.

ROSE, PETER W; PRLIĆ, ANDREAS; BI, CHUNXIAO; BLUHM, WOLFGANG F; CHRISTIE, COLE H; DUTTA, SHUCHISMITA; GREEN, RACHEL KRAMER; GOODSELL, DAVID S; WESTBROOK, JOHN D; WOO, JESSE; YOUNG, JASMINE; ZARDECKI, CHRISTINE; BERMAN, HELEN M; BOURNE, PHILIP E; BURLEY, S. K. The RCSB Protein Data Bank: views of structural biology for basic and applied research and education. **Nucleic acids research**, v. 43, n. D1, p. D345–356, 26 nov. 2014.

Rosetta@Home (2016). Disponível em: <http://boinc.bakerlab.org/rosetta/>. Acessado em Novembro de 2016.

SADASHIV, N.; KUMAR, S. M. D. Cluster, grid and cloud computing: A detailed comparison. Computer Science & Education (ICCSE), 2011 6th International Conference on, 2011. 3-5 Aug. 2011. p.477-482.

SÁNCHEZ-LINARES, IRENE; PÉREZ-SÁNCHEZ, HORACIO; CECILIA, JOSÉ M; GARCÍA, J. M. High-Throughput parallel blind Virtual Screening using BINDSURF. **BMC bioinformatics**, v. 13 Suppl 1, p. S13, jan. 2012.

SANDER, THOMAS; FREYSS, JOEL; VON KORFF, MODEST; RUFENER, C. DataWarrior, An Open-Source Program For Chemistry Aware Data Visualization And Analysis. **Journal of chemical information and modeling**, v. 55, n. 2, p. 460–73, 5 jan. 2015.

Say No to Schistosoma (2016). Disponível em <http://www.worldcommunitygrid.org/research/sn2s/overview.do>. Acessado em Novembro de 2016.

SAYLE, R. A.; MILNERWHITE, E. J. RASMOL - BIOMOLECULAR GRAPHICS FOR ALL. **Trends in Biochemical Sciences,** v. 20, n. 9, p. 374-376, Sep 1995. ISSN 0968-0004.

SETTE, A.; RAPPUOLI, R. Review Reverse Vaccinology : Developing Vaccines in the Era of Genomics. **Immunity**, v. 33, n. 4, p. 530–541, 2000.

SIMONSEN, M; PEDERSEN, C; CHRISTENSEN, M; THOMSEN, R. GPU-Accelerated High-Accuracy Molecular Docking using Guided Differential Evolution. **GECCO-2011: PROCEEDINGS OF THE 13TH ANNUAL GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE**, p. 1803–1810, 2011.

SINAPAD (2015). Disponível em <https://www.lncc.br/sinapad/>. Acessado em Janeiro de 2015.

SMITH, R. **Grid Computing: A Brief Technology Analysis**.

SORNA, VENKATASWAMY; THEISEN, EMILY R; STEPHENS, BRET; WARNER, STEVEN L; BEARSS, DAVID J; VANKAYALAPATI, HARIPRASAD; SHARMA, S. High-throughput virtual screening identifies novel N'-(1-phenylethylidene)-benzohydrazides as potent, specific, and reversible LSD1 inhibitors. **Journal of medicinal chemistry**, v. 56, n. 23, p. 9496–508, 12 dez. 2013.

STIERAND, K.; RAREY, M. Drawing the PDB: Protein-Ligand Complexes in Two Dimensions. **ACS medicinal chemistry letters**, v. 1, n. 9, p. 540–5, 9 dez. 2010.

SUKHWANI, B.; HERBORDT, M. C. Fast binding site mapping using GPUs and CUDA. Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on, 2010. 19-23 April 2010. p.1-8.

TAMILVANAN, T.; HOPPER, W. High-throughput virtual screening and docking studies of matrix protein vp40 of ebola virus. **Bioinformation**, v. 9, n. 6, p. 286–92, jan. 2013.

The Scripps Research (2016). Disponível em: <http://www.scripps.edu/>. Acessado em Novembro de 2016.

THOMSEN, R.; CHRISTENSEN, M. H. MolDock: a new technique for high-accuracy molecular docking. **Journal of medicinal chemistry**, v. 49, n. 11, p. 3315–21, 1 jun. 2006.

TOP500 Supercomputers (2016). Disponível em <www.top500.org/>. Acessado em Novembro de 2016.

TROTT, O.; OLSON, A. J. Software News and Update AutoDock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function, Efficient Optimization, and Multithreading. **Journal of Computational Chemistry,** v. 31, n. 2, p. 455-461, Jan 2010. ISSN 0192-8651.

VAN ZUNDERT, G. C. P.; RODRIGUES, J. P. G. L. M.; TRELLET, M.; SCHMITZ, C.; KASTRITIS, P. L.; KARACA, E.; MELQUIOND A. S. J.; VAN DIJK, M.; DE VRIES, S. J.; BONVIN, A. M. J. J. The HADDOCK2.2 Web Server: User-Friendly Integrative Modeling of Biomolecular Complexes. **Journal of Molecular Biology,** v. 428, n. 4, p. 720-725, Feb 2016.

WANG, R; FANG, X; LU, Y; WANG, S. The PDBbind database: Collection of binding affinities for protein-ligand complexes with known three-dimensional structures. **JOURNAL OF MEDICINAL CHEMISTRY**, v. 47, n. 12, p. 2977–2980, 2004.

WANG, WEI; HE, WANLIN; ZHOU, XI; CHEN, X. Optimization of molecular docking scores with support vector rank regression. **Proteins**, v. 81, n. 8, p. 1386–98, ago. 2013.

We-NMR (2016). Disponível em: <https://www.wenmr.eu/>. Acessado em Novembro de 2016.

WindowBuilder (2016). Disponível em: <http://www.eclipse.org/windowbuilder/>. Acessado em Novembro de 2016.

WLODAWER, A. Rational approach to AIDS drug design through structural biology. **Annual Review of Medicine,** v. 53, p. 595-614, 2002. ISSN 0066-4219.

WCG (2016). Disponível em <http://www.worldcommunitygrid.org>. Acessado em Novembro de 2016.

ZHANG, S. X.; KUMAR, K.; JIANG, X. H.; WALLQVIST, A.; REIFMAN, J. DOVIS: an implementation for high-throughput virtual screening using AutoDock. **Bmc Bioinformatics,** v. 9, Feb 2008. ISSN 1471-2105.

APÊNDICE A

Tabela com o tempo de execução de cada cálculo individual de docking no ambiente de grid computacional heterogêneo.

PDB	Worker	Tempo (Segundos)	PDB	Worker	Tempo (Segundos)	PDB	Worker	Tempo (Segundos)	PDB	Worker	Tempo (Segundos)
1A94	lqtm04-5	3332	**1QI0**	Pentium #2	542	**2JDY**	lqtm04-5	213	**3COW**	lqtm04-7	330
1AMW	lqtm04-6	270	**1R5Y**	lqtm04-4	40	**2OBF**	galaxy-3	1128	**3CKP**	lqtm04-3	534
1A30	lqtm04-2	412	**1RE8**	lqtm04-6	491	**2ON6**	lqtm04-8	127	**3D4Z**	lqtm04-5	62
1B8O	lqtm04-7	255	**1SL3**	lqtm04-8	551	**2OSF**	Pentium #1	71	**3DXG**	nexus-3	413
1B39	lqtm04-4	432	**1SLG**	lqtm04-4	2842	**2P15**	lqtm04-8	247	**3E8R**	lqtm04-8	406
1B7H	lqtm04-3	529	**1SV3**	lqtm04-7	41	**2P4Y**	Pentium #1	915	**3E93**	nexus-2	968
1B9J	lqtm04-8	488	**1SWR**	lqtm04-3	96	**2PGZ**	lqtm04-1	116	**3EHY**	lqtm04-6	81
1ADL	lqtm04-1	245	**1TRD**	lqtm04-7	60	**2POW**	lqtm04-5	101	**3EJR**	lqtm04-5	157
1BCU	Pentium #1	105	**1TSY**	lqtm04-7	106	**2PQ9**	lqtm04-1	147	**3F17**	lqtm04-6	126
1BGQ	Pentium #2	124	**1TXR**	lqtm04-3	212	**2PU2**	lqtm04-5	137	**3F3T**	Pentium #1	474
1C88	galaxy-3	423	**1U1B**	lqtm04-7	1051	**2QBP**	lqtm04-8	354	**3F80**	lqtm04-5	71
1C1V	galaxy-2	1516	**1UTO**	lqtm04-2	36	**2QE4**	lqtm04-4	152	**3FCQ**	lqtm04-6	51
1BXQ	galaxy-1	7862	**1V48**	lqtm04-3	157	**2QFT**	lqtm04-1	46	**3FK1**	lqtm04-4	51
1BXO	tablet-2	6447	**1VSO**	lqtm04-2	151	**2QM9**	lqtm04-5	196	**3G2Z**	lqtm04-1	48
1D7J	nexus-1	69	**1W3K**	lqtm04-6	306	**2QMJ**	lqtm04-1	1301	**3G32**	lqtm04-6	72
1E66	nexus-2	169	**1W3L**	Pentium #2	2432	**2QWB**	lqtm04-7	187	**3G35**	lqtm04-7	92
1DF8	nexus-3	270	**1X1Z**	lqtm04-3	127	**2QWD**	Pentium #2	349	**3GBA**	lqtm04-4	163
1FKB	nexus-1	2541	**1XGJ**	lqtm04-2	128	**2QWE**	lqtm04-4	207	**3G5D**	lqtm04-5	184
1FKI	Pentium #1	115	**1XH6**	galaxy-2	4536	**2R23**	lqtm04-3	142	**3GE7**	lqtm04-1	279
1FLR	Pentium #2	197	**1Y1Z**	lqtm04-8	40	**2RFH**	lqtm04-5	82	**3GNW**	nexus-1	1477
1FTM	nexus-2	207	**1Y6Q**	lqtm04-8	123	**2RKM**	lqtm04-7	253	**3GV9**	lqtm04-6	41
1G2K	Pentium #1	2980	**1YC1**	lqtm04-3	154	**2STD**	lqtm04-8	76	**3GY4**	lqtm04-7	51
1G74	lqtm04-1	258	**1ZEA**	lqtm04-2	5181	**2USN**	lqtm04-5	288	**3HEC**	nexus-3	1477
1GPK	lqtm04-7	66	**2ARM**	lqtm04-8	122	**2UWO**	nexus-1	1101	**3HZK**	lqtm04-6	704
1H23	lqtm04-6	669	**2AVQ**	lqtm04-6	2845	**2V00**	lqtm04-3	76	**3HZM**	lqtm04-8	208
1HFS	nexus-3	4523	**2B1V**	lqtm04-3	97	**2V2H**	lqtm04-4	40	**3IE3**	lqtm04-7	86
1HNN	lqtm04-7	55	**2B7D**	lqtm04-8	500	**2VL4**	lqtm04-8	61	**3IMC**	lqtm04-4	406
1IF7	Pentium #2	435	**2BRB**	lqtm04-1	148	**2VO5**	lqtm04-4	132	**3IWW**	lqtm04-5	203
1J37	lqtm04-7	56	**2BZ6**	lqtm04-3	305	**2VOT**	nexus-2	521	**3IUB**	lqtm04-3	102
1JQ8	nexus-2	3881	**2CET**	lqtm04-1	173	**2W66**	lqtm04-3	101	**3JVS**	lqtm04-7	319
1JYS	lqtm04-2	30	**2CGF**	lqtm04-1	61	**2WEC**	lqtm04-8	547	**3K8Q**	lqtm04-1	25
1KEL	galaxy-3	2016	**2CTC**	lqtm04-3	51	**2WN9**	Pentium #2	218	**3KEJ**	lqtm04-3	273
1LBK	lqtm04-4	203	**2D3U**	lqtm04-3	153	**2X00**	lqtm04-7	147	**3KME**	lqtm04-8	136
1LOL	lqtm04-7	183	**2D1O**	lqtm04-1	844	**2X91**	lqtm04-3	435	**3KEK**	lqtm04-1	303
1LOQ	lqtm04-2	142	**2EXM**	lqtm04-7	62	**2X96**	lqtm04-4	811	**3KV2**	Pentium #1	181
1LVU	lqtm04-8	116	**2F34**	lqtm04-8	177	**2XAB**	Pentium #1	689	**3L4U**	lqtm04-5	400
1MQ6	lqtm04-1	319	**2FLR**	lqtm04-7	171	**2XDL**	lqtm04-5	62	**3L4W**	lqtm04-8	111
1N2V	lqtm04-3	55	**2G5U**	lqtm04-3	71	**2XEG**	Pentium #2	6163	**3L7A**	lqtm04-7	193
1N5R	lqtm04-2	193	**2G71**	galaxy-3	954	**2XJ7**	lqtm04-7	86	**3L7D**	lqtm04-8	158
1NC1	lqtm04-3	51	**2G94**	lqtm04-3	1585	**2XMY**	galaxy-3	1840	**3LKA**	Pentium #1	102
1NJA	lqtm04-8	122	**2G9Q**	lqtm04-8	51	**2ZB0**	lqtm04-5	263	**3MFV**	lqtm04-4	76
1NJE	lqtm04-7	121	**2GSS**	lqtm04-7	91	**2ZCQ**	lqtm04-6	182	**3NEX**	lqtm04-3	238
1NVQ	lqtm04-4	330	**2HA3**	lqtm04-8	35	**2ZCR**	lqtm04-7	258	**3PCE**	lqtm04-1	46
1NWL	lqtm04-3	406	**2HB3**	lqtm04-8	636	**3ACW**	lqtm04-6	86	**3PCJ**	lqtm04-4	40
1O0M	lqtm04-8	127	**2I4J**	nexus-1	1561	**3ADV**	nexus-2	179	**3PCN**	lqtm04-1	56
1O3F	lqtm04-7	157	**2I4X**	lqtm04-7	1298	**3B3C**	nexus-3	127	**3STD**	Pentium #1	380
1O5B	Pentium #2	97	**2J47**	lqtm04-1	202	**3B7I**	lqtm04-3	46	**456C**	lqtm04-4	264
1OS0	lqtm04-2	552	**2J77**	Pentium #1	121	**3B92**	lqtm04-5	96	**4ER2**	nexus-2	5250
1P1Q	lqtm04-1	81	**2J78**	lqtm04-8	85	**3BFU**	lqtm04-8	66	**4TMN**	lqtm04-8	561
1PB8	lqtm04-8	50	**2J94**	lqtm04-1	313	**3BGS**	lqtm04-6	123	**4TIM**	lqtm04-1	66
1PBQ	Pentium #2	91	**2JBJ**	lqtm04-8	96	**3CFT**	lqtm04-7	51	**5ER2**	lqtm04-7	5255
1PS3	lqtm04-7	123	**2JDN**	Pentium #1	147	**3BRA**	lqtm04-3	51	**6CPA**	lqtm04-1	450
1Q84	lqtm04-1	955	**2JDU**	lqtm04-5	40	**3CJ2**	lqtm04-8	57	**6STD**	lqtm04-5	91
1Q8T	lqtm04-8	83									

Tabela com o tempo de execução de cada cálculo individual de docking realizado pelo computador Xeon (lqtm04).

PDB	Tempo (Segundos)	PDB	Tempo (Segundos)	PDB	Tempo (Segundos)	PDB	Tempo (Segundos)
1A94	3332	**1QI0**	256	**2JDY**	213	**3COW**	330
1AMW	270	**1R5Y**	40	**2OBF**	131	**3CKP**	534
1A30	412	**1RE8**	491	**2ON6**	127	**3D4Z**	62
1B8O	255	**1SL3**	551	**2OSF**	40	**3DXG**	120
1B39	432	**1SLG**	2842	**2P15**	247	**3E8R**	406
1B7H	529	**1SV3**	41	**2P4Y**	439	**3E93**	277
1B9J	488	**1SWR**	96	**2PGZ**	116	**3EHY**	81
1ADL	245	**1TRD**	60	**2POW**	101	**3EJR**	157
1BCU	53	**1TSY**	106	**2PQ9**	147	**3F17**	126
1BGQ	70	**1TXR**	212	**2PU2**	137	**3F3T**	195
1C88	85	**1U1B**	1051	**2QBP**	354	**3F80**	71
1C1V	196	**1UTO**	36	**2QE4**	152	**3FCQ**	51
1BXQ	1119	**1V48**	157	**2QFT**	46	**3FK1**	51
1BXO	537	**1VSO**	151	**2QM9**	196	**3G2Z**	48
1D7J	30	**1W3K**	306	**2QMJ**	1301	**3G32**	72
1E66	70	**1W3L**	567	**2QWB**	187	**3G35**	92
1DF8	101	**1X1Z**	127	**2QWD**	172	**3GBA**	163
1FKB	737	**1XGJ**	128	**2QWE**	207	**3G5D**	184
1FKI	60	**1XH6**	397	**2R23**	142	**3GE7**	279
1FLR	95	**1Y1Z**	40	**2RFH**	82	**3GNW**	421
1FTM	86	**1Y6Q**	123	**2RKM**	253	**3GV9**	41
1G2K	642	**1YC1**	154	**2STD**	76	**3GY4**	51
1G74	258	**1ZEA**	5181	**2USN**	288	**3HEC**	426
1GPK	66	**2ARM**	122	**2UWO**	261	**3HZK**	704
1H23	669	**2AVQ**	2845	**2V00**	76	**3HZM**	208
1HFS	1233	**2B1V**	97	**2V2H**	40	**3IE3**	86
1HNN	55	**2B7D**	500	**2VL4**	61	**3IMC**	406
1IF7	206	**2BRB**	148	**2VO5**	132	**3IWW**	203
1J37	56	**2BZ6**	305	**2VOT**	166	**3IUB**	102
1JQ8	1063	**2CET**	173	**2W66**	101	**3JVS**	319
1JYS	30	**2CGF**	61	**2WEC**	547	**3K8Q**	25
1KEL	206	**2CTC**	51	**2WN9**	106	**3KEJ**	273
1LBK	203	**2D3U**	153	**2X00**	147	**3KME**	136
1LOL	183	**2D1O**	844	**2X91**	435	**3KEK**	303
1LOQ	142	**2EXM**	62	**2X96**	811	**3KV2**	85
1LVU	116	**2F34**	177	**2XAB**	105	**3L4U**	400
1MQ6	319	**2FLR**	171	**2XDL**	62	**3L4W**	111
1N2V	55	**2G5U**	71	**2XEG**	2639	**3L7A**	193
1N5R	193	**2G71**	130	**2XJ7**	86	**3L7D**	158
1NC1	51	**2G94**	1585	**2XMY**	251	**3LKA**	50
1NJA	122	**2G9Q**	51	**2ZB0**	263	**3MFV**	76
1NJE	121	**2GSS**	91	**2ZCQ**	182	**3NEX**	238
1NVQ	330	**2HA3**	35	**2ZCR**	258	**3PCE**	46
1NWL	406	**2HB3**	636	**3ACW**	86	**3PCJ**	40
1O0M	127	**2I4J**	461	**3ADV**	69	**3PCN**	56
1O3F	157	**2I4X**	1298	**3B3C**	51	**3STD**	170
1O5B	45	**2J47**	202	**3B7I**	46	**456C**	264
1OS0	552	**2J77**	15	**3B92**	96	**4ER2**	1714
1P1Q	81	**2J78**	85	**3BFU**	66	**4TMN**	561
1PB8	50	**2J94**	313	**3BGS**	123	**4TIM**	66
1PBQ	45	**2JBJ**	96	**3CFT**	51	**5ER2**	5255
1PS3	123	**2JDN**	75	**3BRA**	51	**6CPA**	450
1Q84	955	**2JDU**	40	**3CJ2**	57	**6STD**	91
1Q8T	83						

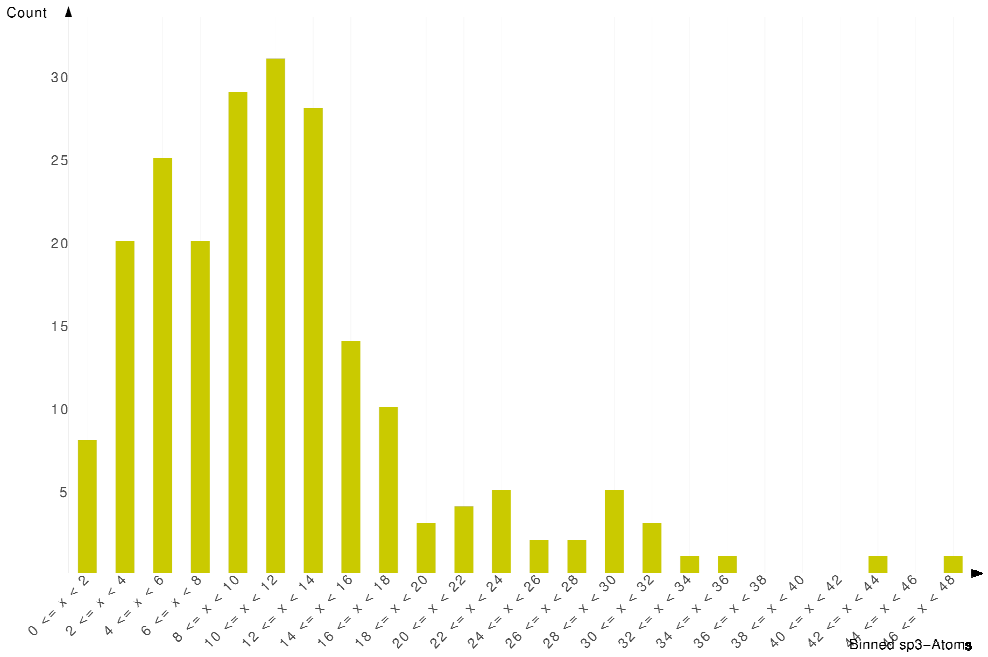


Gráfico da distribuição do número de átomos sp3 entre os 213 ligantes do conjunto de moléculas selecionado.

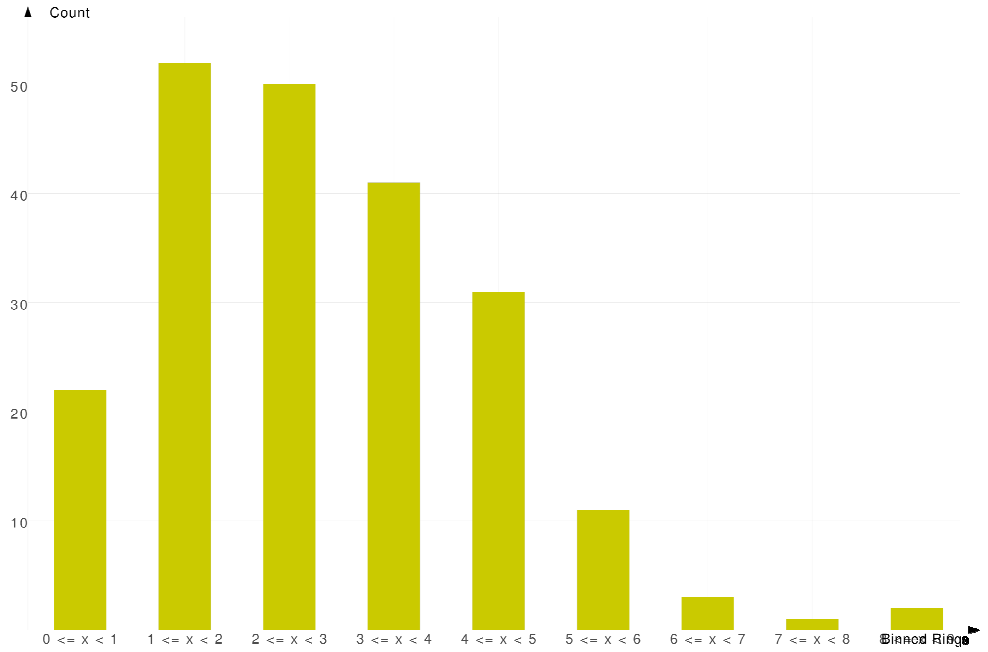


Gráfico da distribuição do número de anéis aromáticos entre os 213 ligantes do conjunto de moléculas selecionado.

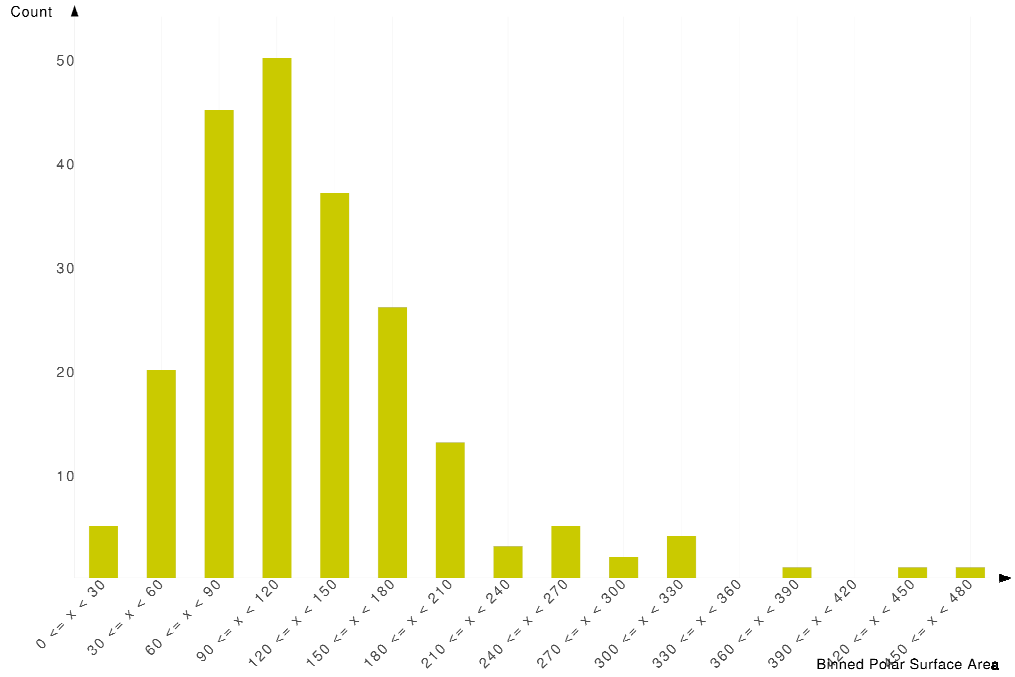


Gráfico da distribuição da área de superfície polar entre os 213 ligantes do conjunto de moléculas selecionado.

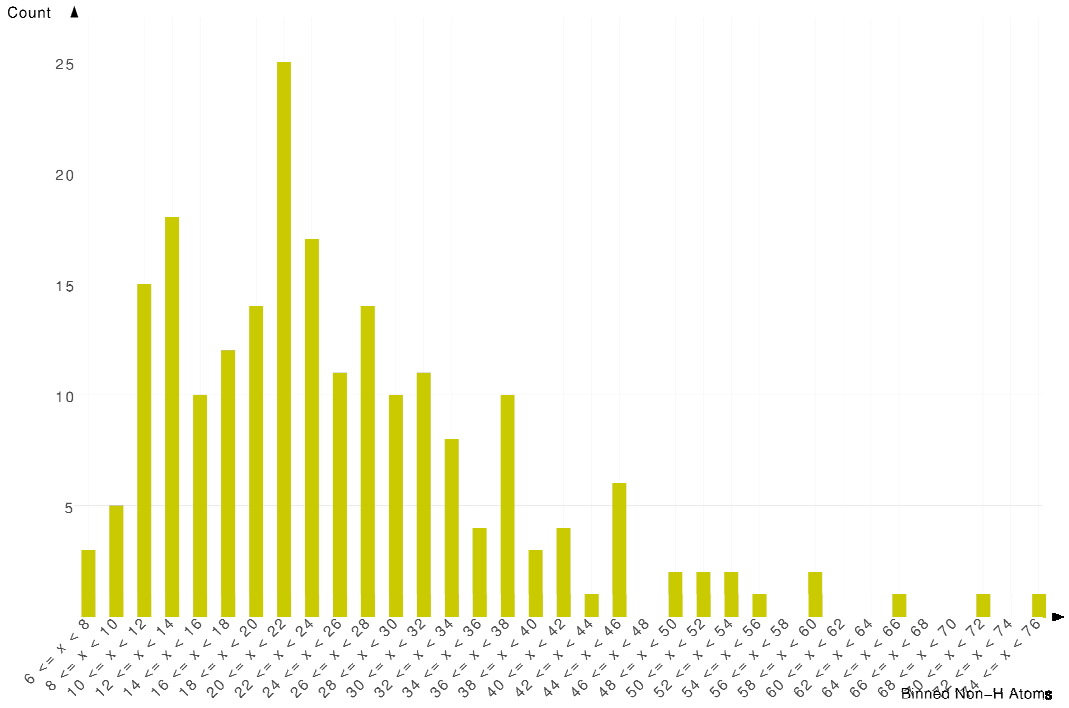


Gráfico da distribuição do número de átomos não hidrogênio entre os 213 ligantes do conjunto de moléculas selecionado.

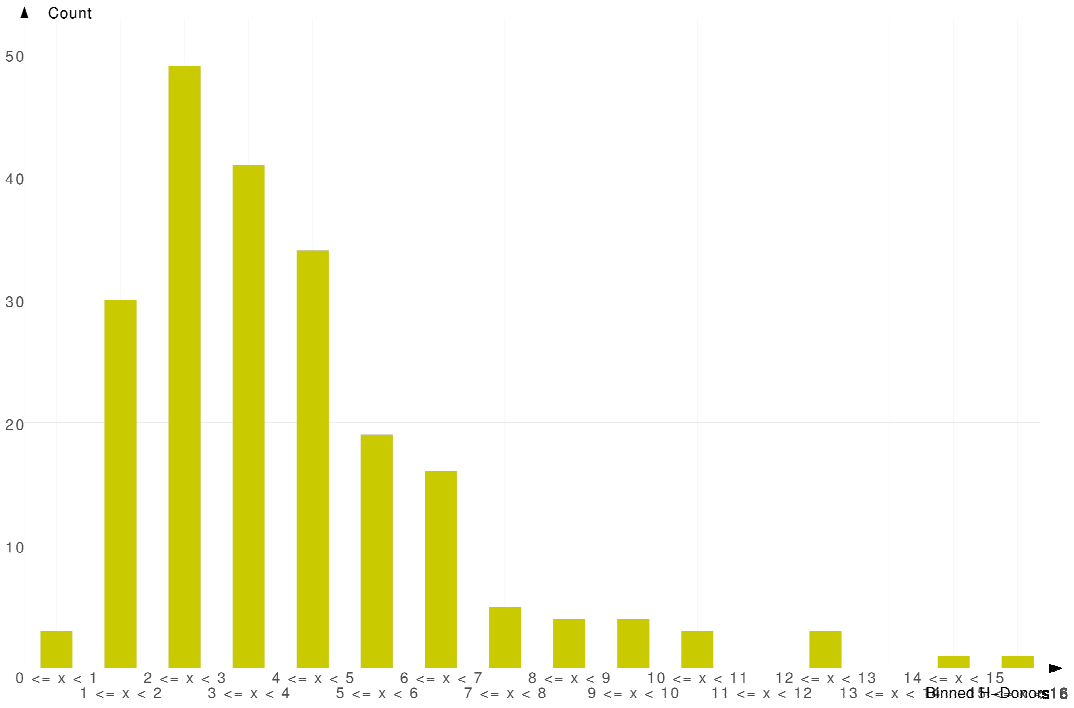


Gráfico da distribuição do número de doadores de hidrogênios entre os 213 ligantes do conjunto de moléculas selecionado.

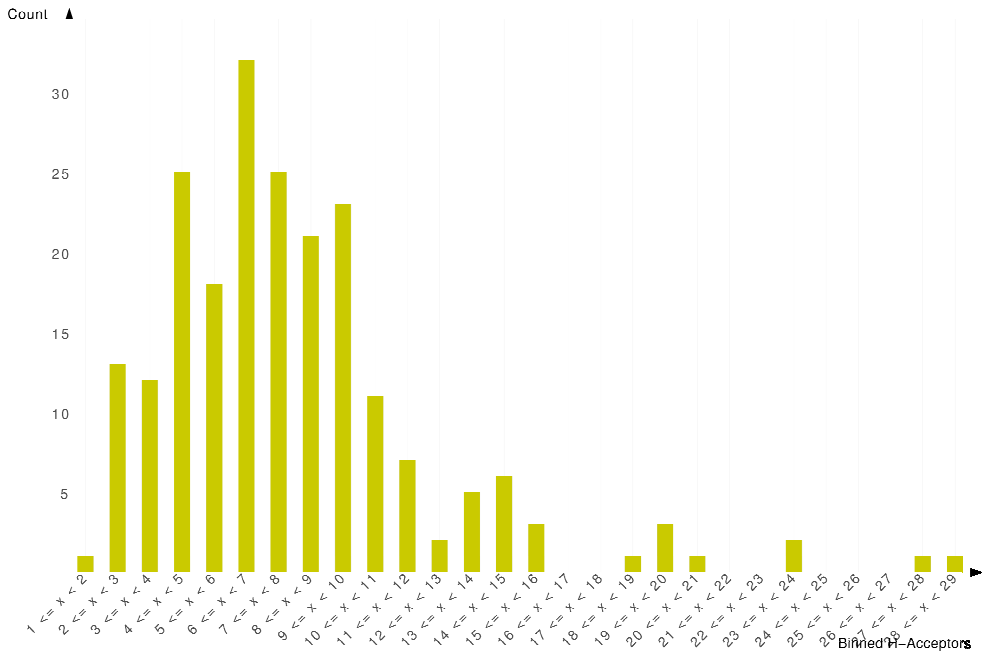


Gráfico da distribuição do número de aceitadores de hidrogênios entre os 213 ligantes do conjunto de moléculas selecionado.

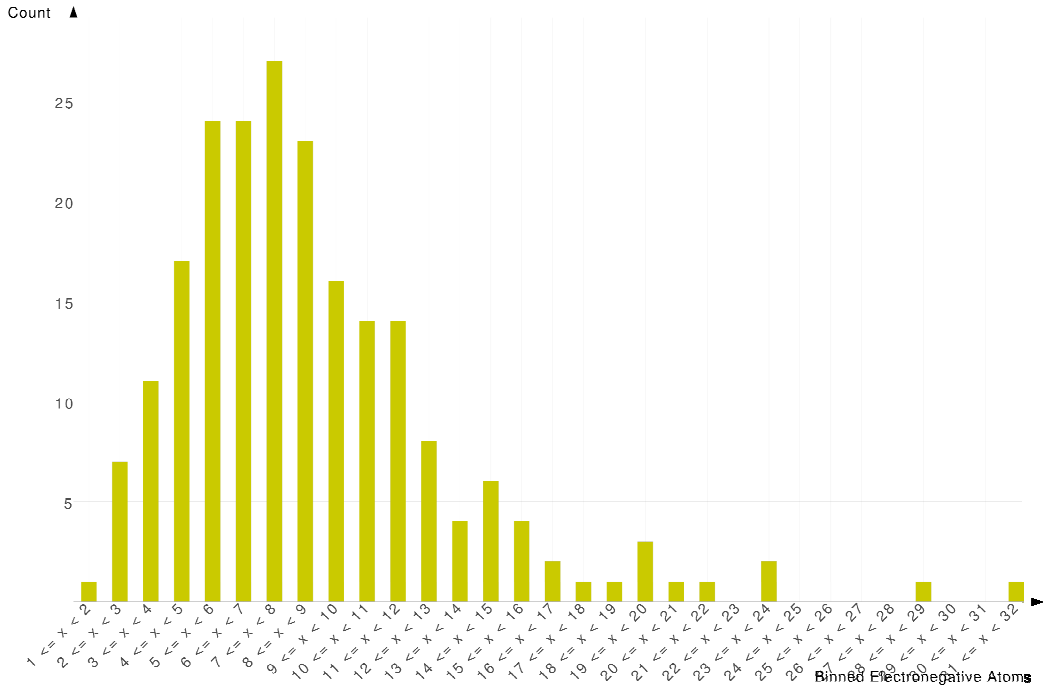


Gráfico da distribuição do número de átomos eletronegativos entre os 213 ligantes do conjunto de moléculas selecionado.

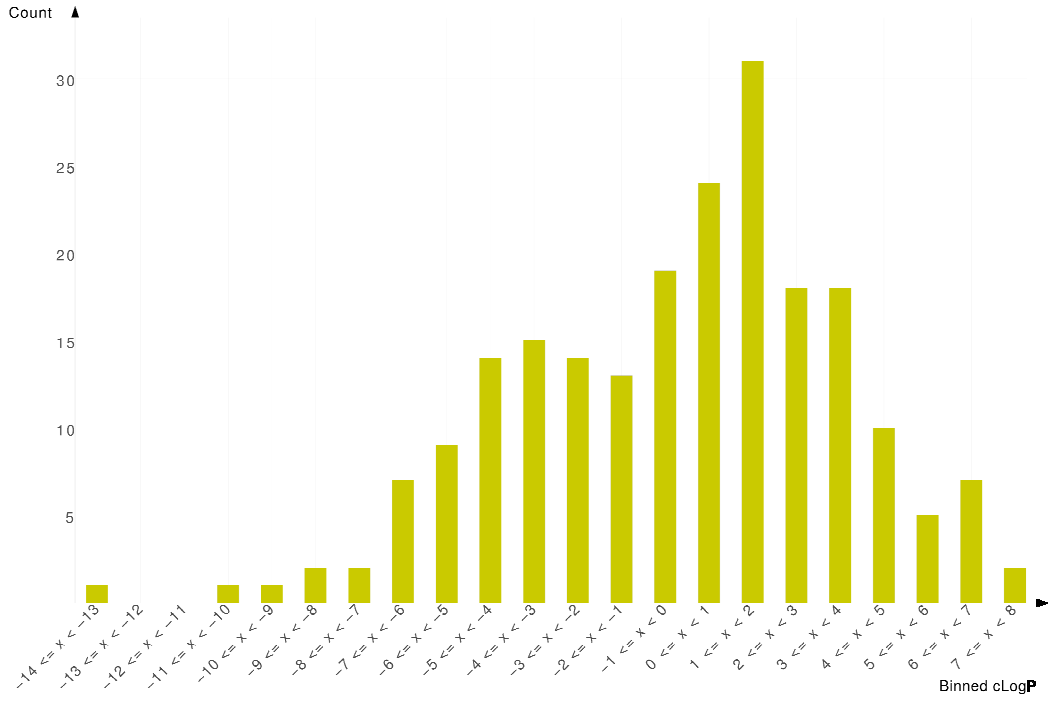


Gráfico da distribuição dos valores do coeficiente de partição (LogP) entre os 213 ligantes do conjunto de moléculas selecionado.

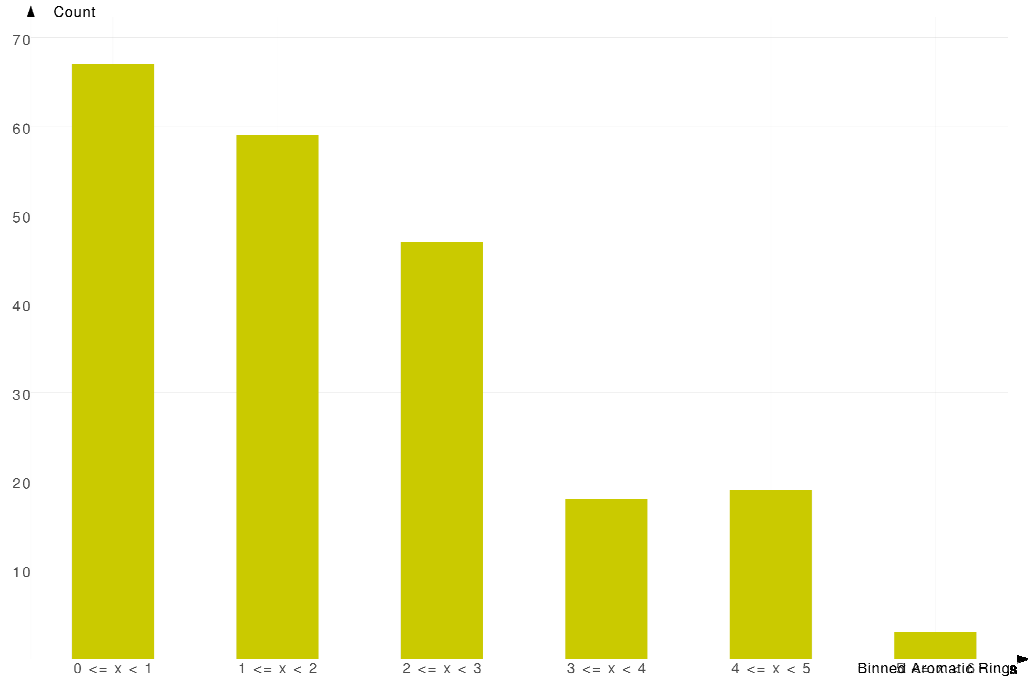


Gráfico da distribuição do número de anéis aromáticos entre os 213 ligantes do conjunto de moléculas selecionado.

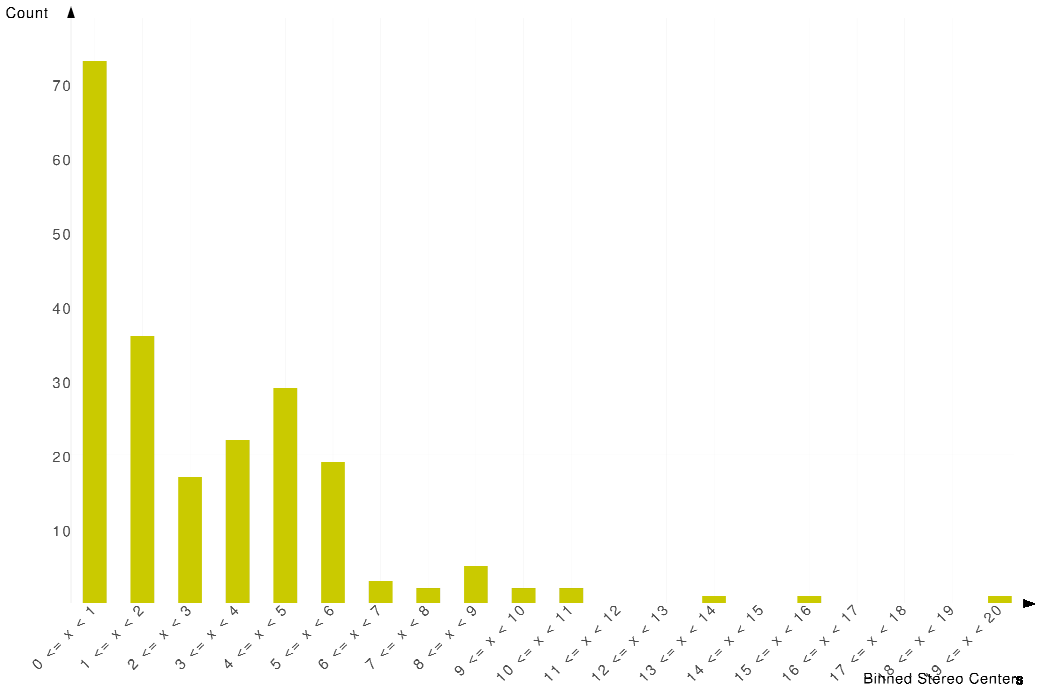
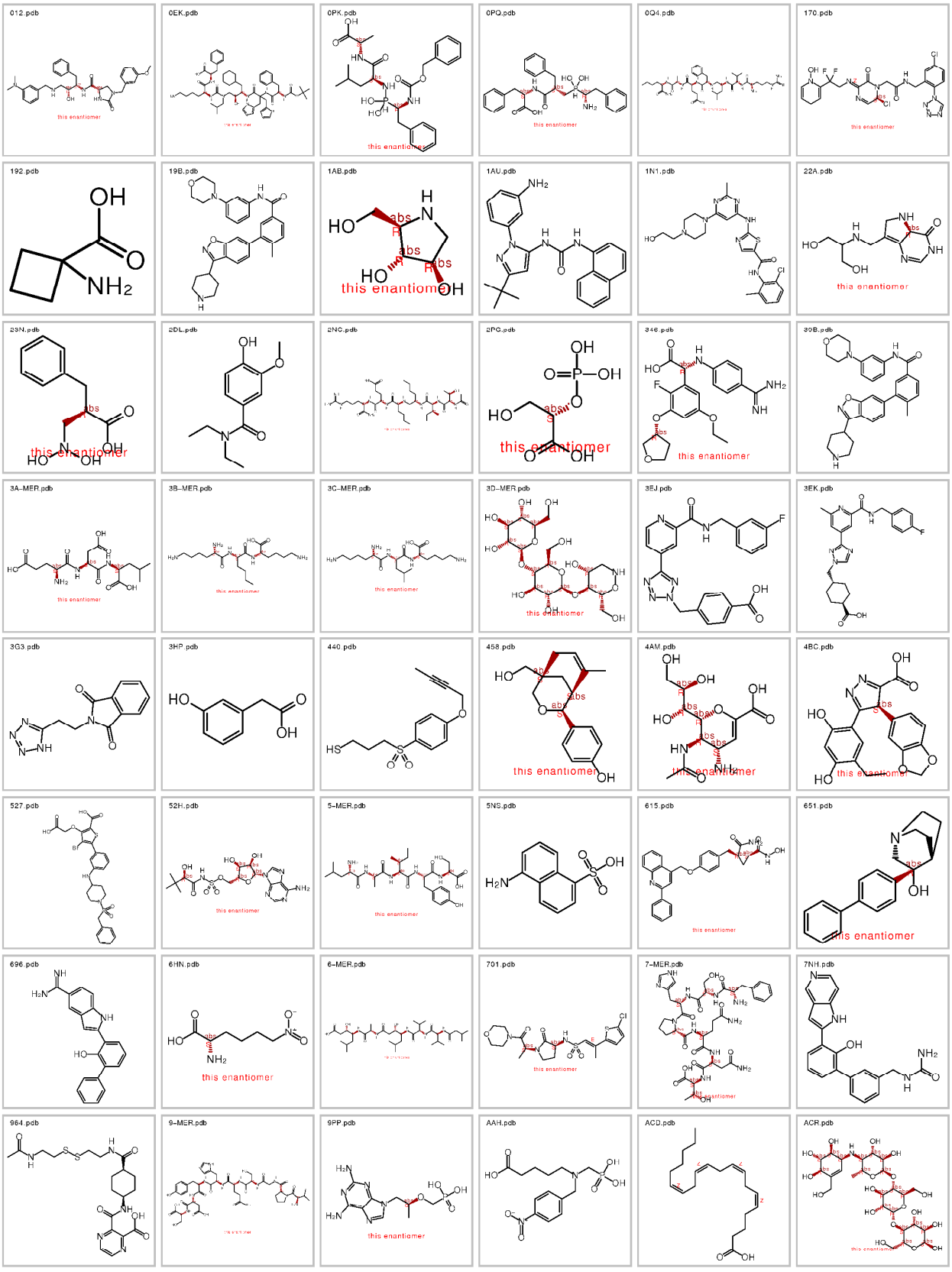
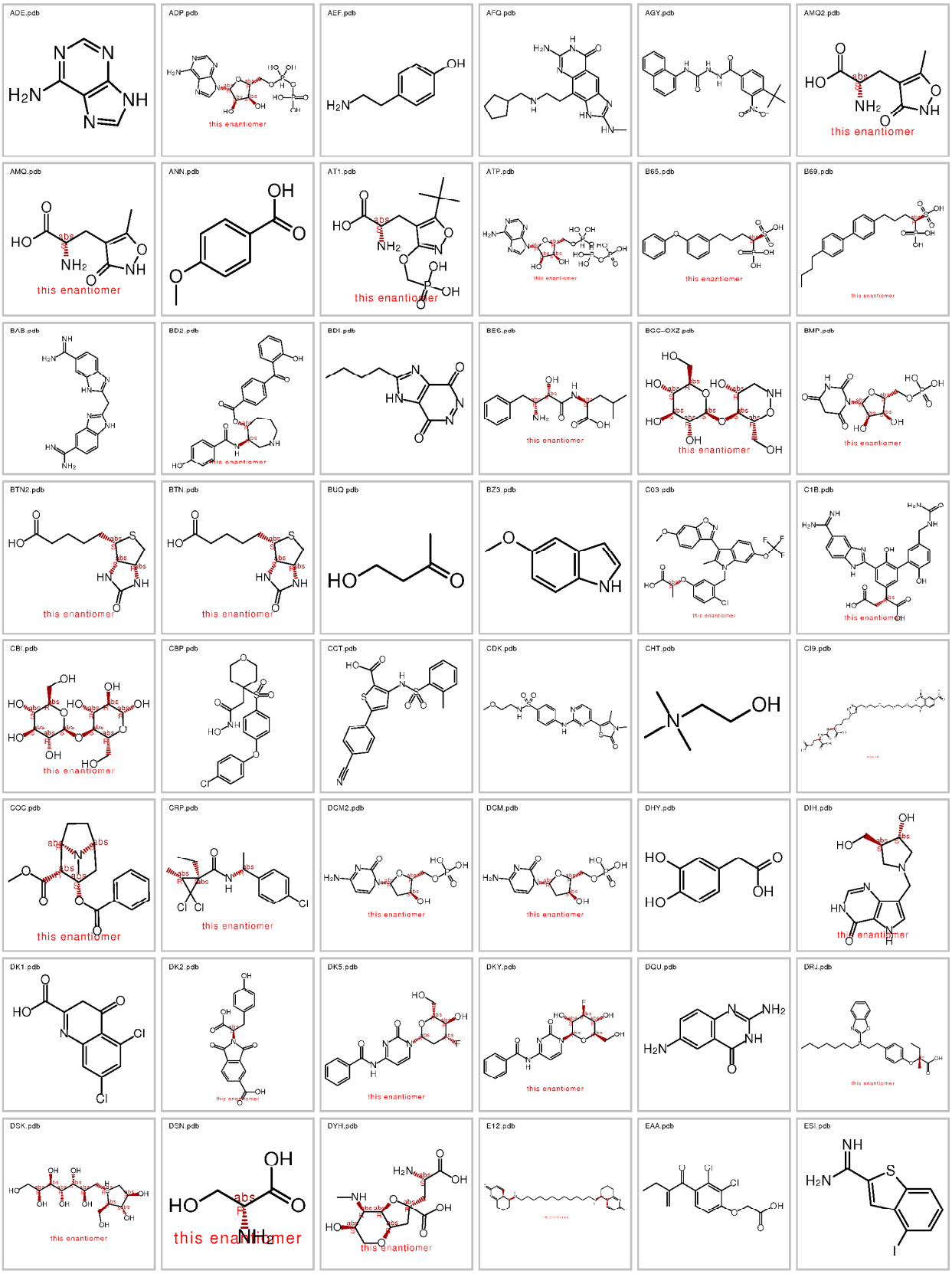
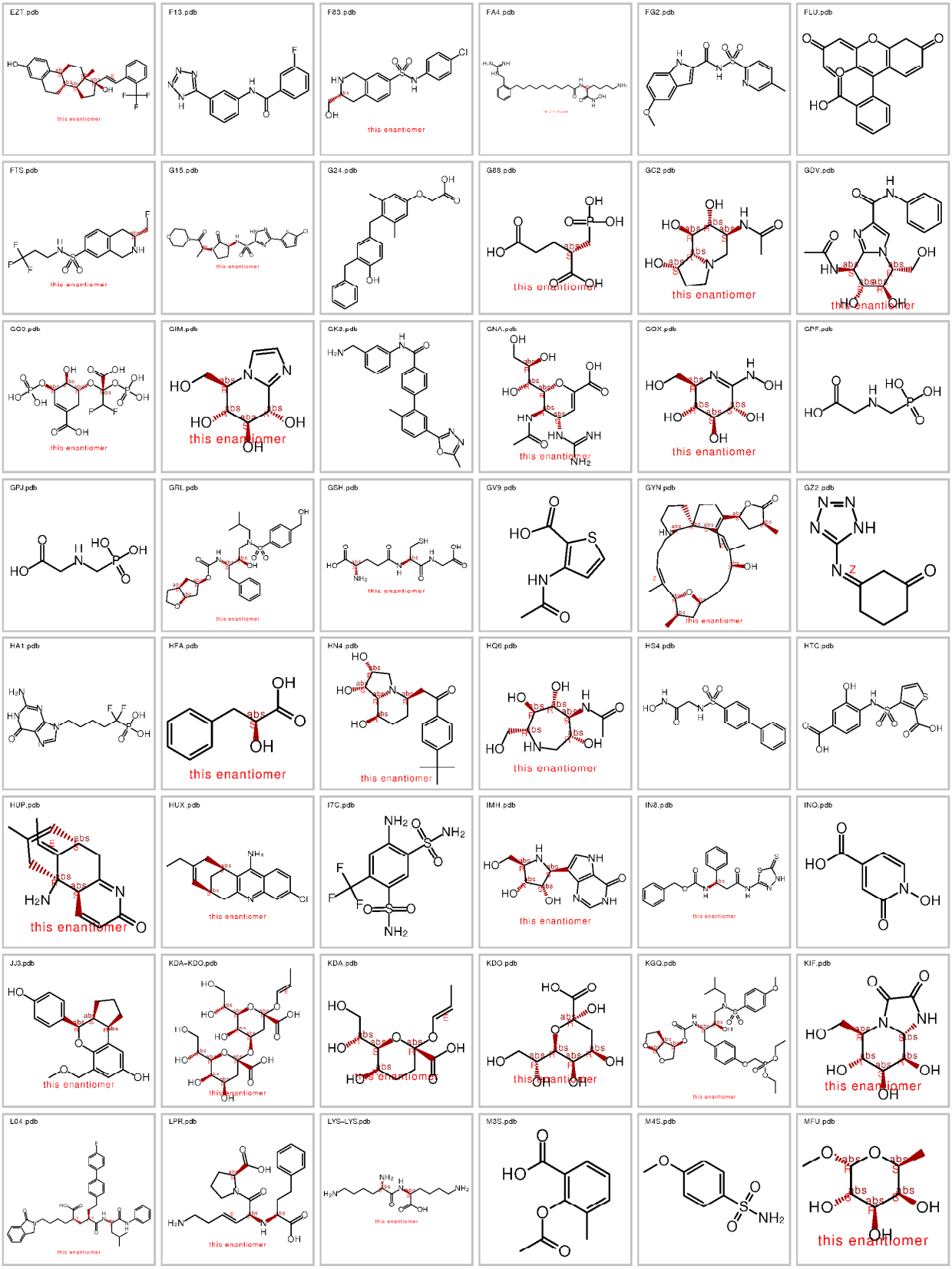


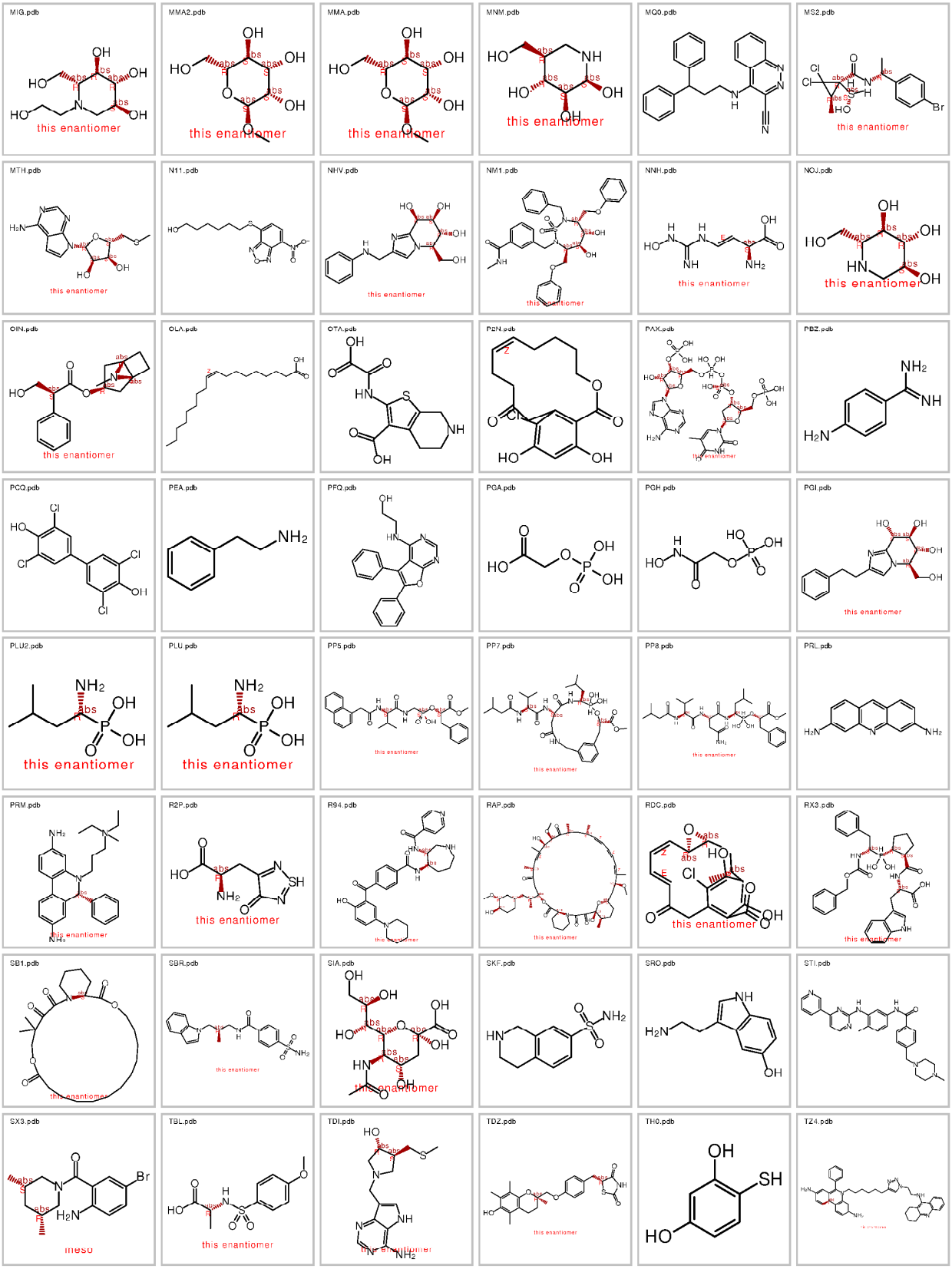
Gráfico da distribuição do número de centros estereogênicos entre os 213 ligantes do conjunto de moléculas selecionado.

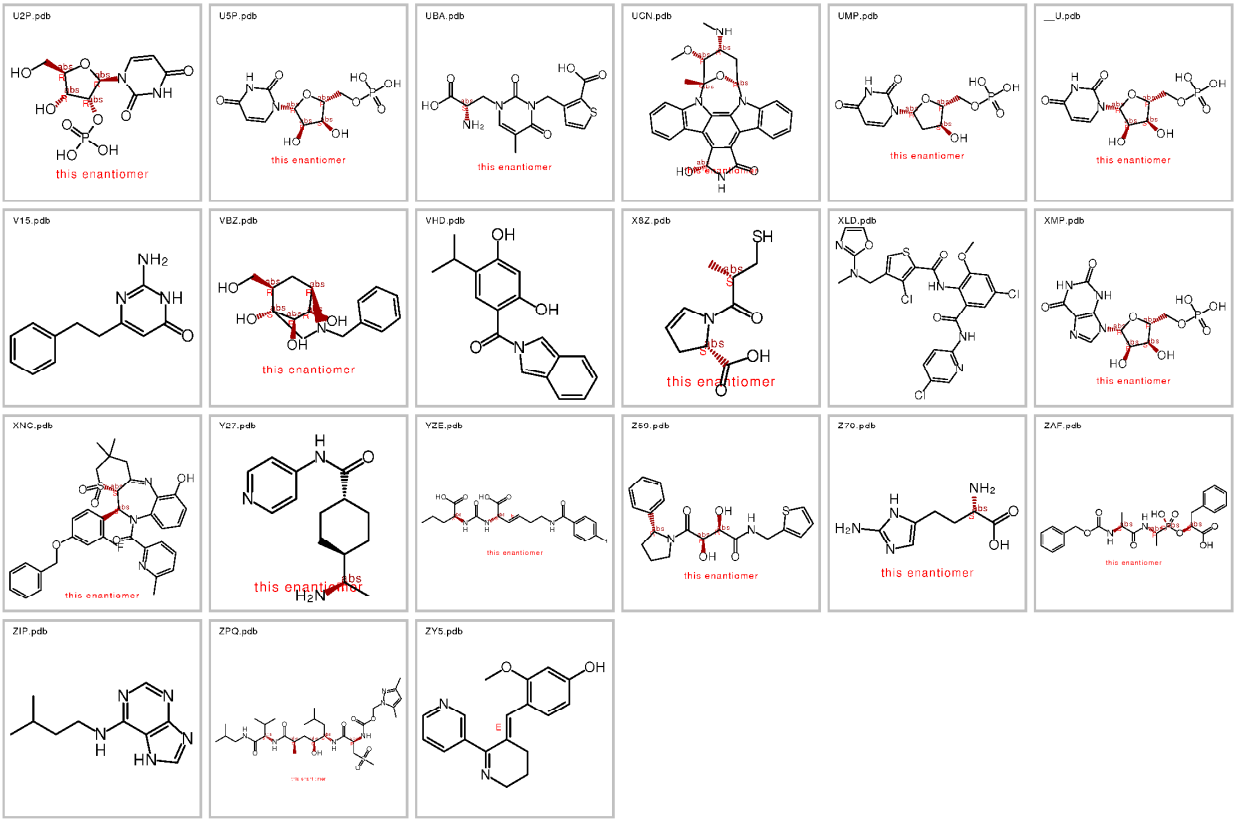
Tabela contendo o desenho em 2D de todos os 213 ligantes utilizados neste trabalho.




ADE.pdb
ADP.pdb
this enantiomer
AEF.pdb
AFQ.pdb
AGY.pdb
AMQ2.pdb
this enantiomer
AMQ.pdb
this enantiomer
ANN.pdb
AT1.pdb
this enantiomer
ATP.pdb
this enantiomer
B65.pdb
this enantiomer
B69.pdb
this enantiomer
BAB.pdb
BD2.pdb
this enantiomer
BDI.pdb
BEC.pdb
this enantiomer
BGC-OXZ.pdb
this enantiomer
BMP.pdb
this enantiomer
BTN2.pdb
this enantiomer
BTN.pdb
this enantiomer
BUQ.pdb
BZ3.pdb
C03.pdb
this enantiomer
C1B.pdb
this enantiomer
CBI.pdb
this enantiomer
CBP.pdb
CCT.pdb
CDK.pdb
CHT.pdb
CI9.pdb
COC.pdb
this enantiomer
CRP.pdb
this enantiomer
DCM2.pdb
this enantiomer
DCM.pdb
this enantiomer
DHY.pdb
DIH.pdb
this enantiomer
DK1.pdb
DK2.pdb
this enantiomer
DK5.pdb
this enantiomer
DKY.pdb
this enantiomer
DQU.pdb
DRJ.pdb
this enantiomer
DSK.pdb
this enantiomer
DSN.pdb
this enantiomer
DYH.pdb
this enantiomer
E12.pdb
EAA.pdb
ESI.pdb


EZT.pdb
F13.pdb
F83.pdb
FA4.pdb
FG2.pdb
FLU.pdb
FTS.pdb
G15.pdb
G24.pdb
G88.pdb
GC2.pdb
GDV.pdb
GG9.pdb
GIM.pdb
GK3.pdb
GNA.pdb
GOX.pdb
GPF.pdb
GPJ.pdb
GRL.pdb
GSH.pdb
GV9.pdb
GYN.pdb
GZ2.pdb
HA1.pdb
HFA.pdb
HN4.pdb
HQ6.pdb
HS4.pdb
HTC.pdb
HUP.pdb
HUX.pdb
I7C.pdb
IMH.pdb
IN8.pdb
INO.pdb
JJ3.pdb
KDA-KDO.pdb
KDA.pdb
KDO.pdb
KGQ.pdb
KIF.pdb
L04.pdb
LPR.pdb
LYS-LYS.pdb
M3S.pdb
M4S.pdb
MFU.pdb
this enantiomer


MIG.pdb
this enantiomer
MMA2.pdb
this enantiomer
MMA.pdb
this enantiomer
MNM.pdb
this enantiomer
MQ9.pdb
MS2.pdb
this enantiomer
MTH.pdb
this enantiomer
N11.pdb
NHV.pdb
this enantiomer
NM1.pdb
this enantiomer
NNH.pdb
this enantiomer
NOJ.pdb
this enantiomer
OIN.pdb
this enantiomer
OLA.pdb
OTA.pdb
P2N.pdb
PAX.pdb
PBZ.pdb
PCQ.pdb
PEA.pdb
PFQ.pdb
PGA.pdb
PGH.pdb
PGI.pdb
this enantiomer
PLU2.pdb
this enantiomer
PLU.pdb
this enantiomer
PP5.pdb
this enantiomer
PP7.pdb
this enantiomer
PP8.pdb
this enantiomer
PRL.pdb
PRM.pdb
this enantiomer
R2P.pdb
this enantiomer
R94.pdb
this enantiomer
RAP.pdb
this enantiomer
RDC.pdb
this enantiomer
RX3.pdb
this enantiomer
SB1.pdb
this enantiomer
SBR.pdb
this enantiomer
SIA.pdb
this enantiomer
SKF.pdb
SRO.pdb
STI.pdb
SX3.pdb
meso
TBL.pdb
this enantiomer
TDI.pdb
this enantiomer
TDZ.pdb
this enantiomer
TH0.pdb
TZ4.pdb


U2P.pdb
this enantiomer
U5P.pdb
this enantiomer
UBA.pdb
this enantiomer
UCN.pdb
UMP.pdb
this enantiomer
__U.pdb
this enantiomer
V15.pdb
VBZ.pdb
this enantiomer
VHD.pdb
X8Z.pdb
this enantiomer
XLD.pdb
XMP.pdb
this enantiomer
XNC.pdb
this enantiomer
Y27.pdb
this enantiomer
YZE.pdb
this enantiomer
Z50.pdb
this enantiomer
Z70.pdb
this enantiomer
ZAF.pdb
this enantiomer
ZIP.pdb
ZPQ.pdb
ZY5.pdb