

UNIVERSIDADE DE BRASÍLIA
Instituto de Química
Curso: Química Tecnológica

Arthur Alves Nascimento

Implementação de GPU e automação para o AutoDock4 e AutoDock Vina usando Raccoon

Brasília
2025

Arthur Alves Nascimento

Implementação de GPU e automação para o AutoDock4 e AutoDock Vina usando Raccoon

Trabalho apresentado a Universidade de Brasília, Campus Darcy Ribeiro, como requisito para obtenção do título de Bacharel em Nome do Curso.

Orientador: Prof. Dr. João Batista Lopes Martins
Coorientadora: Profª. Dra. Érica Cristina Moreno Nascimento

BRASÍLIA
2025

À minha família.

AGRADECIMENTOS

Gostaria de agradecer ao meu pai Rubens, minha mãe Reuza e meu irmão Eduardo, que me deram suporte e motivação para estudar e buscar conhecer o motivo e o funcionamento do mundo.

Muito obrigado a todos os alunos do LQC da UnB, por terem me recebido e ajudado a entender melhor o mundo da química computacional e um agradecimento especial ao Rafael Campos, que fez algumas correções de bugs do Raccoon, e ao Leonardo que fez a parte de programação da junção do Raccoon+AD-GPU.

Gostaria também de expressar minha gratidão a professora Érica e o professor João Batista por terem me orientado durante o meu PIBIC, 2 TPs e o meu TCC.

Gostaria de agradecer ao Brenno, Dome, Bel, Guilherme e Yann por fazerem amizade comigo durante o nosso primeiro semestre da graduação, e aos meus amigos que fiz durante o correr do curso de química tecnológica, Marcelo, Pedro, Cadu, Ingrid, Paulo, Camila e Luana.

"Se não há um meio de vencer, crie um meio". – Mestre Arsenal

RESUMO

A química computacional, especificamente através do docking molecular, consolidou-se como uma ferramenta indispensável no desenvolvimento de fármacos. Entretanto, softwares tradicionais como o AutoDock4 apresentem limitações significativas de tempo de processamento e alternativas aceleradas, como o AutoDock-GPU, careçam de interfaces gráficas acessíveis. O presente trabalho teve como objetivo desenvolver, testar e validar um fluxo de trabalho e um ambiente computacional que integre a interface amigável do software Raccoon com a capacidade de processamento paralelo do AutoDock-GPU. Para a concretização deste objetivo, foi utilizada a tecnologia Docker para padronizar a instalação e criar um ambiente Linux acessível via navegador (browser), além do desenvolvimento de um script em linguagem Python para automatizar a listagem de ligantes e mapas. A metodologia aplicada consistiu na realização de testes de desempenho utilizando uma base de dados de 156 ligantes frente a dois receptores (as enzimas 3IK3 e hAChE), comparando-se os tempos de execução entre processamento em CPU e GPU em duas configurações de hardware distintas. Os resultados obtidos demonstraram uma redução drástica no tempo computacional, variando de milhares de minutos na execução via CPU para poucos minutos na execução via GPU, confirmando a eficiência da paralelização. Conclui-se que o ambiente desenvolvido associa com êxito a praticidade de configuração do Raccoon à velocidade do AutoDock-GPU, promovendo a acessibilidade e a usabilidade dessas ferramentas para pesquisadores com menor proficiência em programação, apesar de limitações pontuais na automação de múltiplos receptores simultâneos.

Palavras-chave: Docking molecular .AutoDock-GPU .Raccoon. Docker.

ABSTRACT

Computational chemistry, specifically through molecular docking, has established itself as an indispensable tool in drug development, although traditional software such as AutoDock4 presents significant processing time limitations and accelerated alternatives, such as AutoDock-GPU, lack accessible graphical interfaces. The present work aimed to develop, test, and validate a workflow and a computational environment that integrates the user-friendly interface of the Raccoon software with the parallel processing capacity of AutoDock-GPU. To achieve this objective, Docker technology was used to standardize the installation and create a Linux environment accessible via a browser, in addition to the development of a script in the Python language to automate the listing of ligands and maps. The applied methodology consisted of conducting performance tests using a database of 156 ligands against two receptors (the enzymes 3IK3 and hAChE), comparing execution times between CPU and GPU processing on two different hardware configurations. The results obtained demonstrated a drastic reduction in computational time, ranging from thousands of minutes in CPU execution to a few minutes in GPU execution, confirming the efficiency of parallelization. It is concluded that the developed environment successfully combines the configuration practicality of Raccoon with the speed of AutoDock-GPU, promoting the accessibility and usability of these tools for researchers with less proficiency in programming, despite specific limitations in the automation of multiple simultaneous receptors.

Key-words: Molecular Docking .AutoDock-GPU .Raccoon . Docker.

LISTA DE FIGURAS

Figura	Título	Página
Figura 1 -	Janela inicial do Raccoon	12
Figura 2 -	Aba “Ligands” (Ligantes) do Raccoon	12
Figura 3 -	Aba “Receptors” (Receptores) do Raccoon	13
Figura 4 -	Aba “Maps” (mapas) do Raccoon	13
Figura 5 -	Aba “Docking” do Raccoon	14
Figura 6 -	Aba “VS Generation” (geração de triagem virtual) do Raccoon	14
Figura 7 -	Estrutura de pastas criadas pelo Raccoon	15
Figura 8 -	Representação do modelo chave fechadura	18
Figura 9 -	Representação do modelo de ajuste induzido	19
Figura 10 -	Estrutura de pastas usadas pelo Raccoon+AD-GPU	28

TABELAS

Tabela 1 - Tipos de softwares de Docking Molecular. 21

Tabela 2 - Requisitos do AutoDock-GPU 23

Tabela 3 - Tempos de cálculo do AutoDock-GPU 26

LISTA DE ABREVIATURAS E SIGLAS

CADD	Computer-Aided Drug Design
AD4	AutoDock 4: Programa de docking automatizado que usa o Algoritmo Genético Lamarckiano (LGA) para prever interações ligante-macromolécula
VINA	AutoDock VINA: Programa de docking mais rápido que o AD4, que usa uma função de escore simplificada e otimização eficiente
ADT	AutoDockTool: Interface gráfica (GUI) para o AutoDock, utilizada para preparar arquivos (PDBQT), executar cálculos e analisar resultados de docking
LGA	Algoritmo Genético Lamarckiano
CCSB	Center for Computational Structural Biology
MGL	Molecular Graphics Laboratory
GPU	Unidade de Processamento Gráfico
AD_GPU	AutoDock-GPU: Implementação OpenCL do AutoDock4 acelerada por GPU, que usa arquitetura paralela para reduzir o tempo de execução
LGA	Algoritmo Genético Lamarckiano
CPU	Unidade Central de Processamento
RAM	Random Access Memory
WSL	Subsistema do Windows para Linux
CLI	Interface de linha de comando
HTS	Triagem de alto rendimento
grid	Refere-se a um conjunto de linhas e colunas que formam vários pontos
gpf	Grid parameters files
dpf	Docking parameters files
Mol2	Tipo de arquivo de estrutura tridimensional de uma molecula
grid	Refere-se a um conjunto de linhas e colunas que formam varios pontos

PDB	Tipo de arquivo usado para o Protein Data Bank, que descreve estruturas tridimensionais de biomoléculas.
PDBQT	Variação do PDB usada principalmente para simulações de docking
BCR-ABL	Proteína mutada da leucemia mieloide crônica
hAChE	Proteína acetilcolinesterase
E	Proteína
S	Substrato
P1	Parte 1
P2	Parte 2

LISTA DE SÍMBOLOS

ΔG	Aproximação da Energia Livre

SUMÁRIO

1 INTRODUÇÃO 7
1.1 Contextualização 9
1.1.1 AutoDock4 9
1.1.2 AutoDock Tools 11
1.1.3 Raccoon 11
1.1.4 AutoDock-GPU 15
1.1.5 WSL 16
1.1.6 Docker 16
1.2 FORMULAÇÃO DO PROBLEMA 16
1.3 OBJETIVO GERAL 17
1.4 OBJETIVOS ESPECÍFICOS 17
2 REVISÃO TEÓRICA 18
2.1 Componentes Metodológicos do Docking Molecular 19
2.1.1. Algoritmos de Busca (Search Algorithms) 19
2.1.2 Funções de Pontuação (Scoring Functions) 20
2.2. Softwares e Aplicações Práticas 20
3 PROCEDIMENTOS EXPERIMENTAIS 22
4 RESULTADOS E DISCUSSÃO 24
4.1. Raccoon 25
4.2. Listador 25
4.3. AD-GP 26
4.3.1 Usando o AD-GPU 26
4.4 Raccoon+AD-GPU 26
5 CONSIDERAÇÕES FINAIS OU CONCLUSÃO 29
BIBLIOGRAFIA 30

1 INTRODUÇÃO

A química computacional é o uso de computadores em conjunto de equações matemáticas que são capazes de representar interações físico-químicas [1]. Esta área da química consolidou-se como ferramenta indispensável no desenvolvimento moderno de fármacos, permitindo a investigação detalhada de propriedades moleculares, conformações estruturais e interações em nível atômico e molecular. Entre suas diversas aplicações, destaca-se a capacidade de acelerar significativamente a triagem de moléculas bioativas contra alvos terapêuticos específicos, particularmente por meio da predição computacional da afinidade de ligantes por sítios proteicos. A utilização estratégica da modelagem molecular e suas técnicas, com ênfase no docking molecular, proporciona vantagens substantivas no *pipeline* de descoberta de fármacos, incluindo a redução de custos com reagentes, otimização de recursos energéticos e racionalização do tempo de pesquisa e desenvolvimento.[2]

O docking molecular configura-se como uma metodologia computacional automatizada projetada para prever, *in silico*, a interação molecular entre moléculas pequenas e seus alvos biomoleculares. Essa abordagem constitui o cerne do projeto de fármacos assistido por computador (Computer-Aided Drug Design, CADD), área que tem impulsionado continuamente o desenvolvimento e o refinamento de ferramentas computacionais especializadas para criar e estudar o desenvolvimento de novos fármacos. Dentre os softwares mais proeminentes deste contexto, destaca-se o pacote desenvolvido pelo Center for Computational Structural Biology (CCSB) sucessor do Molecular Graphics Laboratory (MGL), que inclui ferramentas amplamente adotadas pela comunidade científica, como AutoDock4 (AD4), AutoDock Vina (VINA) e AutoDockTools (ADT) [3]. O AutoDock4, por exemplo, emprega um algoritmo genético Lamarckiano (LGA) combinado com uma função de pontuação semiempírica de energia livre, demonstrando eficácia particular para ligantes contendo até aproximadamente dez ligações rotacionais flexíveis. [4, 5]

Quando comparado com outros programas de docking, como o GOLD [6], um software de docking desenvolvido pelo Cambridge Crystallographic Data Centre (CCDC), o AD4 sozinho é considerado computacionalmente lento, restringindo severamente sua aplicabilidade em triagens virtuais de larga escala, cenários que envolvem rotineiramente bibliotecas compostas por milhões de moléculas [7]. Porém

por ser um software livre, programas que permitem aos usuários executarem, copiarem, modificar e distribuir o código sem restrições, assim sendo dependente da comunidade para manter ele competitivo com seus concorrentes.

A resposta a estas limitações, têm surgido de esforços de otimização e automação dos cálculos de docking, com a implementação do programa Raccoon, desenvolvido na linguagem python, também desenvolvida pelo MGL. Este software oferece automação de vários cálculos de dockings por meio de uma interface gráfica, porém com um gargalo na execução dos cálculos, por realizar os cálculos de forma sequencial (em série).

Em contrapartida, a maneira de realizar o cálculo de docking molecular foi melhorado com a implementação do AutoDock-GPU (AD-GPU), que é capaz de realizar cálculos em paralelo, representando uma implementação avançada do AutoDock4, reprogramada em OpenCL e CUDA para aproveitar a arquitetura massivamente paralela de unidades de processamento gráfico (GPUs) e de unidades centrais de processamento (CPU). Essa adaptação permite ao AD-GPU ter ganhos de desempenho notáveis, podendo reduzir o tempo de execução de docking em até 350 vezes em comparação com a execução *single-threaded* tradicional [7]. Porém, o AD-GPU somente está disponível para computadores com sistema operacional baseados em Linux. Sendo este entrave parcialmente resolvido com o uso do WSL (Subsistema do Windows para Linux).

Apesar da disponibilidade online dessas ferramentas otimizadas, o Raccoon apresenta limitações em relação a velocidade de cálculo de docking e o AD-GPU apresenta barreiras para usuários novos e pouco experimentes em programação e execução de programas mais robustos, dependo do WSL, linhas de comando no terminal e conhecimentos avançados em informática. Essa barreira técnica limita substancialmente a acessibilidade e a usabilidade por pesquisadores das áreas químicas e farmacológicas, que podem não possuir formação computacional avançada, destacando a necessidade premente de soluções que conciliem alta eficiência computacional com interfaces intuitivas e acessíveis.

Um programa capaz de unir a interface gráfica e praticidade do Raccoon com a velocidade e eficiência do AD-GPU seria ideal, porém devido à grande variedade de hardware possível em computadores isto se torna um desafio. Uma das possíveis maneiras de conseguir este programa é usando o Docker, uma

plataforma de código aberto que permite criar pacotes de programas capazes de padronizar a instalação e o uso deste programa.

1.1 Contextualização

O docking molecular é metodologia computacional que explora o comportamento de moléculas pequenas em sítios de ligações de receptores e tem como objetivo prever a orientação e conformação preferencial de duas ou mais moléculas ao formarem um complexo estável, além de estimar a afinidade de ligação entre elas [8]. Usando funções de pontuação (score) para avaliar a energia de interação dos complexos, e explorar possíveis novas moléculas que podem desempenhar um papel importante no combate de alguma doença ou na melhor compreensão de algum fenômeno biológico. Na era do desenho racional de fármacos, esta técnica consolidou-se como uma ferramenta pivotal, permitindo agilizar e baratear o processo de descoberta e desenvolvimento de novos medicamentos, que tradicionalmente consome bilhões de dólares e possui uma alta taxa de insucesso. [8]

1.1.1 AutoDock4

O AD4 é um conjunto de ferramentas de acoplamento molecular automatizado desenvolvido pelo MGL, projetado para prever como pequenas moléculas interagem com um receptor com estrutura tridimensional conhecida. Consistindo em basicamente 2 programas: "autodock4.exe” que calcula o docking do ligante em uma área (grid) que represente parte da proteína, e o "autogrid4.exe” que pré-calcula o grid.

O grid é calculado a partir das dimensões, das coordenadas e do centro da “caixa” onde os mapas são calculados, estes parâmetros são armazenados em um arquivo do tipo “.gpf” onde são atribuídos um mapa de potencial para cada átomo do ligante. Cada tipo de átomo tem seu próprio arquivo “.map”, um arquivo “.fld” é usado pelo AutoDock para verificar se todos os mapas foram atribuidos corretamente.

O AutoDock tem implementado um campo de força com uma aproximação da energia livre usando parâmetros semiempíricos para avaliar conformações durante simulações. Este campo de força foi parametrizado usando uma ampla base de dados de complexos proteína-inibidor para os quais tanto a estrutura quanto as constantes de inibição são conhecidas.[7]

O campo de força avalia as interações em 2 passos, sendo o ponto de partida tanto o ligante quanto a proteína em uma conformação não ligada, onde a energia intramolecular é estimada para transição do estado não ligante para o estado e conformação ligante. A segunda etapa é avaliar a energia intramolecular do ligante com a proteína na conformação de interação. A equação 1 abaixo representa o cálculo da energia livre usando parâmetros semiempíricos.

$$\Delta G = (V_{bound}^{L-L} - V_{unbound}^{L-L}) + (V_{bound}^{P-P} - V_{unbound}^{P-P}) + \left(V_{bound}^{P-L} - V_{unbound}^{P-L} + \Delta S_{conf}\right) \quad (1)$$

em que L se refere ao ligante e P a proteína, V são avaliações ΔS_{conf} OBJ é uma estimativa da $V = W_{vdw} \sum_{1,j} \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}\right) + W_{hbond} \sum_{i,j} E(t) \left(\frac{C_{ij}}{r_{ij}^{12}} - \frac{D_{ij}}{r_{ij}^{10}}\right) +$

$W_{elec} \sum_{i,j} \frac{q_i q_j}{e(r_{ij}) r_{ij}} + W_{sol} \sum_{i,j} \left(S_i V_j + S_j V_i\right) e^{\left(-\frac{r_{ij}^2}{2\sigma^2}\right)}$ (2)

Onde as constantes de parametrização (W) foram otimizadas para calibrar a energia livre empírica com base em um conjunto de constantes de ligações determinadas experimentalmente. O primeiro termo é um potencial típico Lennard-Jones de tipo 6/12 para interações de dispersão/repulsão. Os parâmetros são baseados no campo de força Amber [9]. O segundo termo representa e mensura a energia da ligação de hidrogênio direcional baseado em um potencial de Lennard-Jones do tipo de 10/12. Os parâmetros C e D são atribuídos para fornecer uma profundidade máxima de poço de 5 kcal/mol a 1,9 Å para ligações de hidrogênio com oxigênio e nitrogênio, e uma profundidade de poço de 1 kcal/mol a 2,5 Å para ligações de hidrogênio com enxofre. Para cada elemento atômico alguns parâmetros são adaptados para melhor representação.

A função E(t) fornece peso baseado no angulo direcional da geometria ideal de ligação de hidrogênio. O terceiro termo é um potencial de Coulomb blindado para as interações do tipo eletrostáticas. O termo final ($W_{sol} \sum_{i,j} \left(S_i V_j + S_i V_j\right) e^{\left(\frac{-r_{ij}^2}{2\sigma^2}\right)}$) é um potencial de solvatação baseado no volume de átomos (V) que circundam um determinado átomo e o protegem do solvente, ponderado por um parâmetro de solvatação (S) e um termo exponencial com fator de ponderação de distância σ = 3,5 Å. [9]

É possível adicionar carga nos elementos usando o programa ADT, sendo elas cargas de Gasteiger. Estas cargas são calculadas utilizando o método do *Parial Equalization of Orbital Electronegativities* (PEOE), tem como base a eletronegatividade, transferência iterativa, ajuste dinâmico e convergência.

1.1.2 AutoDockTools

O ADT é uma ferramenta gráfica feita para facilitar a configuração de triagens virtuais de docking, possibilitando o preparo, execução e análise dos estudos. Este software permite atribuir e modificar a carga dos átomos dos ligantes e dos receptores, também possibilita definir a flexibilização das ligações químicas tanto do ligante quanto dos resíduos do receptor. Por meio de sua interface gráfica, permite a configuração dos arquivos “.gpf” e “.dpf” de maneira facilitada, permitindo a visualização, criação e configuração do grid com a proteína. Para criar o grid basta clicar na aba grid e selecionar o receptor, depois selecionar o ligante, após isso clique em grid options e configurar as dimencoes da caixa e o dentro dela, depois disso salve o arquivo “.gpf”. [9]

1.1.3 Raccoon

O Raccoon é um programa desenvolvido na linguagem Python3 e de fácil instalação, ele é uma interface gráfica para processamento de bibliotecas de ligantes em diferentes formatos (“.mol2”, “.pdb” e “pdbqt”), múltiplos receptores e resíduos flexíveis, para gerar todos os arquivos necessários para executar uma triagem virtual usando o AutoDock. Apesar de sua versatilidade na montagem do setup de estudos de virtual screening, o Raccon não traz como padrão a possibilidade de paralelização dos cálculos. O programa ainda depende de outro software para gerar os arquivos “.gpf” e “.dpf”, apesar de ter uma opção padrão para esses dois arquivos, não sendo capaz também de ler os arquivos “.dlg” e depende do executável doAD4 para realizar os cálculos do docking.

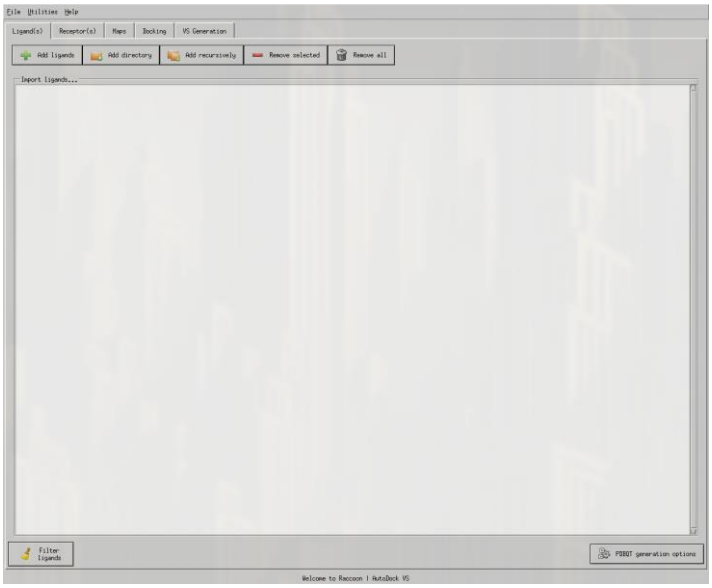


Figura 1. Janela Principal do Programa Raccoon.

O programa é estruturado em abas (seções), conforme descrito em seu manual técnico:

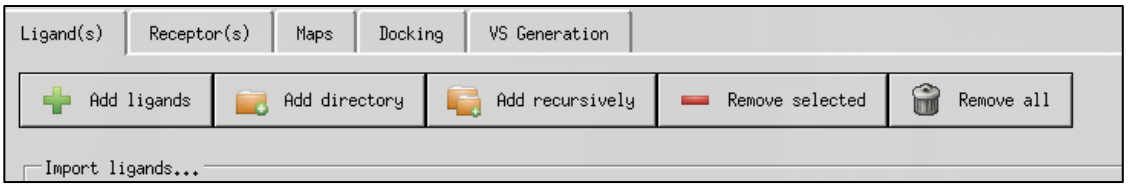


Figura 2. Programa Raccoon: Aba "Ligands" (Ligantes).

Permite ao usuário carregar os arquivos dos ligantes, que devem estar nos formatos “.mol2”, “.pdb” ou “.pdbqt”. Nesta seção, o Raccoon é capaz de processar os ligantes para alterar sua flexibilidade (converter de rígido para flexível e vice-versa), adicionar cargas e reparar eventuais problemas estruturais.



Figura 3. Programa Raccoon: Aba "Receptors" (Receptores).

NA aba “Receptors” feita a inserção de um ou mais receptores, aceitando os mesmos formatos de arquivo (“.mol2”, “.pdb”, “.pdbqt”). A ferramenta permite a definição de resíduos flexíveis; contudo, esta opção não está disponível para mais de uma proteína ficando limitada a uma única proteína por triagem. Também é possível modificar o receptor, realizando ações como adição de cargas, adição ou remoção de hidrogênios, remoção de moléculas de água e seleção de cadeias proteicas específicas.



Figura 4. Programa Raccoon: Aba "Maps"(mapas).

Nesta etapa, o usuário deve fornecer os arquivos do grid. É possível indicar um arquivo de parâmetros de grid (“.gpf”) pré-existente (gerado em outro programa ou editado manualmente) ou apontar o local onde os mapas já calculados estão armazenados. Também é necessário indicar o local do executável autogrid4.exe (dis-

ponível no mesmo repositório do AutoDock). A ferramenta oferece a opção de executar o AutoGrid individualmente para cada docking ou executá-lo uma única vez para todo o conjunto de simulações.

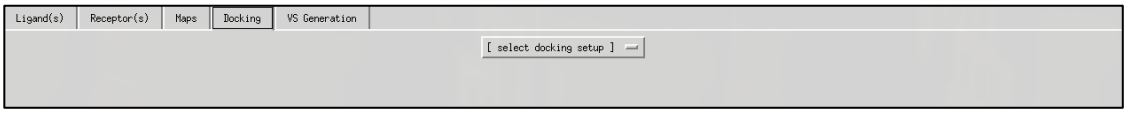


Figura 5. Programa Raccoon: Aba "Docking".

Na aba ‘Docking’ define os parâmetros da simulação de docking. O usuário pode selecionar um arquivo de parâmetros (“.dpf”) padrão, editá-los manualmente na própria interface ou carregar um arquivo “.dpf” customizado.



Figura 6. Programa Raccoon: Aba "VS Generation" (Geração da Triagem Virtual).

Esta é a etapa final de configuração. A aba apresenta um resumo do protocolo de docking e solicita ao usuário o diretório de destino onde o Raccoon deve gerar os arquivos de saída. Adicionalmente, permite a seleção do sistema operacional (SO) de destino, o que é relevante para ambientes como clusters Linux, onde é possível definir limitações e configurações de execução específicas.

Os arquivos de saída gerados pelo Raccoon seguem um padrão de organização específico, apresentados no esquema da Figura 7.

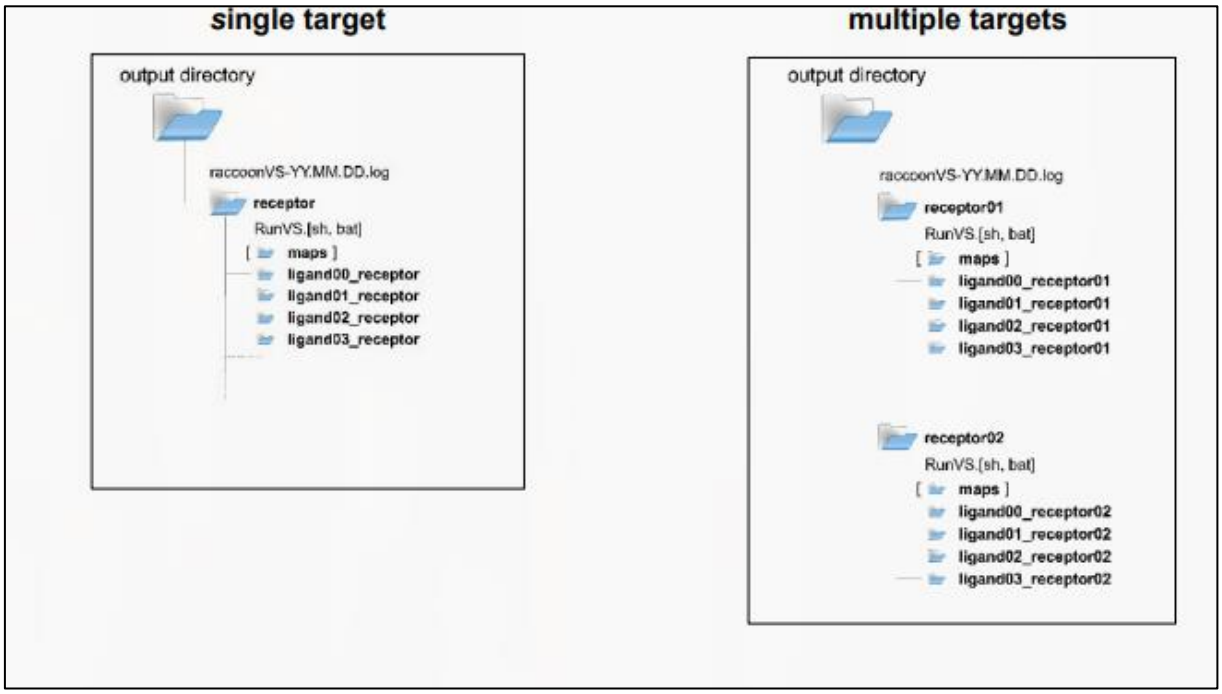


Figura 7. Estrutura das pastas criadas pelo Raccoon, imagem retirada do manual de usuário. Figura retirada do manual do Raccoon.

Para iniciar os cálculos, basta executar o script gerado (*RunVS.sh* para Linux/MacOS ou *RunVS.bat* para Windows), seja por execução direta ou via terminal.[9]

1.1.4 AutoDock-GPU

O AD-GPU é um programa capaz de paralelizar e automatizar os cálculos de docking usando o AD4, sem interface gráfica e apenas disponível para computadores com os sistemas operacionais baseados em Linux e MacOS. Para computadores que usam sistema operacional Windows sua utilização é viabilizada através do WSL (Subsistema do Windows para Linux). Este programa também é incapaz de ler e visualizar arquivos ".dlg", porém ainda é dependente de outros programas para calcular os arquivos ".maps". Além de possuir uma instalação complicada que necessita conhecimento específicos em computação. O repositório oficial do AD-GPU está disponível no GitHub [11].

1.1.5 WSL

O WSL é um recurso do Windows que permite executar um ambiente Linux computadores Windows, sem a necessidade de uma máquina virtual separada ou inicialização dupla. O WSL 2 usa tecnologia de virtualização para executar um kernel do Linux dentro de uma máquina virtual. Para sua instalação basta executar o comando "*wsl --install*" no terminal do Windows.[12]

1.1.6 Docker

O Docker é um projeto de código aberto usado para automatizar a implantação de aplicativos como contêineres portáteis e autônomos que podem ser executados na nuvem ou localmente. Contêineres do Docker podem ser executados nativamente no Linux e no Windows, porém imagens Linux podem ser executadas em hosts Linux e em hosts do Windows (usando uma VM Linux Hiper-V).[13]

1.2 FORMULAÇÃO DO PROBLEMA

Atualmente, existe uma diversidade de programas para docking molecular, porém o acesso amplo e gratuito a essas ferramentas mostra-se limitado a algumas opções. Muitos desses softwares livres dependem fortemente do desenvolvimento pela comunidade, o que pode impactar seu desenvolvimento, continuidade e suporte.

Entre estas opções, destaca-se os programas desenvolvidos pela The Scripps Research Institute, o AutoDock 4 e AutoDock Vina, que possui interface gráfica acoplada externamente (AutoDockTools) - um ambiente que permite a interação com o computador por meio de elementos visuais como ícones, botões, menus e janelas. No entanto, o programa apresenta limitações operacionais: não é capaz de executar cálculos em paralelo, nem de operar de maneira automatizada em larga escala, dependendo de scripts de terminal para realizar grandes quantidades de simulações de docking. Além disso, o processo consome considerável quantidade de tempo, variando conforme as configurações específicas do docking. Esta limitação temporal deriva fundamentalmente da arquitetura computacional implementada pelo Scripps Research (anteriormente CCRB), que não permite a utilização de mais de um core da CPU por simulação individual.

O Scripps Research disponibiliza outro programa com interface gráfica denominado Raccoon, capaz de organizar, converter e calcular múltiplos estudos de docking. Contudo, mesmo esta ferramenta opera os cálculos em série, permanecendo incapaz de aproveitar múltiplos cores da CPU para processamento paralelo.

Uma alternativa mais eficiente do mesmo desenvolvedor é o AutoDock-GPU, que realiza cálculos de docking de maneira paralela, aproveitando plenamente os recursos de processamento. Entretanto, esta versão carece de interface gráfica e está disponível exclusivamente para sistemas operacionais baseados em Linux. Adicionalmente, o AutoDock-GPU não calcula os mapas de energia necessários para as simulações, dependendo que estes sejam previamente gerados por outros programas.

A ausência de interface gráfica, a restrição a sistemas Linux e a operacionalização via terminal limitam significativamente a adoção dessas ferramentas por usuários com menor proficiência em química e modelagem computacional. Paralelamente, a baixa eficiência computacional do Raccoon, em relação ao tempo de cálculo, o torna menos elegível entre as alternativas disponíveis, criando um cenário onde os usuários precisam escolher entre facilidade de uso e eficiência computacional para realização de virtual screening de milhares de moléculas.

1.3 OBJETIVO GERAL

Desenvolver, testar e validar uma ferramenta e um workflow usando a linguagem Phyton para associar o Raccoon com o AD-GPU, para usabilidade e acessibilidade e popularização do uso deste tipo de ferramenta.

1.4 OBJETIVOS ESPECÍFICOS

1. Delimitar a funcionalidade do Raccoon.
2. Analisar a funcionalidade e potencialidade do AD-GPU.
3. Desenvolver ambiente multi-software entre o Raccoon e o AD-GPU: docker.
4. Estudar a viabilidade do ambiente desenvolvido.
5. Testar a exequibilidade do ambiente desenvolvido por meio de estudo de docking molecular usando 156 ligantes que foram estudados no Laboratório de Química Computacional (LQC) e 2 proteínas também estudadas no LQC.

6. Preparar manuais para o uso das ferramentas computacionais utilizadas e desenvolvidas.

2 REVISÃO TEÓRICA

O docking molecular é uma metodologia *in sílico* de modelagem molecular usada para simular o processo natural de reconhecimento molecular, no qual biomoléculas interagem seletivamente para desencadear eventos biológicos cruciais, como transcrição, sinalização celular e catálise enzimática. Para entender esse processo, diferentes modelos teóricos foram propostos [14]:

- Modelo "Chave e Fechadura" (Lock and Key): Proposto por Emil Fischer em 1890, postula que o ligante (chave) possui uma forma complementar e rígida ao sítio ativo do receptor (fechadura).[14]

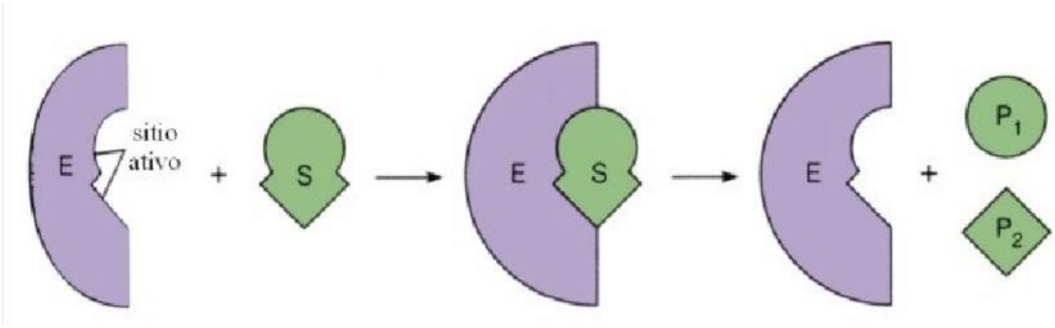


Figura 8: Modelo de chave e fechadura, onde E representa a proteína, S representa um substrato, P1 e P2 são parte 1 e parte 2 respectivamente. Figura retirada do Pereira, Francisco. (2010). BIOCHEMISTRY IN A CHEMICAL APPROACH. 10.13140/RG.2.2.17251.14889.

- Modelo do Ajuste Induzido (Induced Fit): Introduzido por Daniel Koshland em 1958, este modelo é mais dinâmico, sugerindo que tanto o ligante quanto o receptor sofrem ajustes conformacionais mútuos para alcançar um encaixe ideal, analogamente a uma "mão em uma luva".[15]

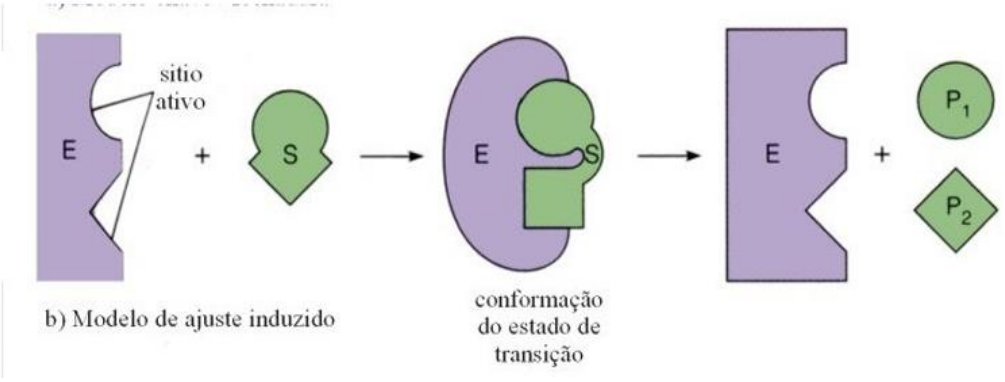


Figura 9: modelo de ajuste induzido, onde E representa a proteína, S representa um substrato, P1 e P2 são parte 1 e parte 2 respectivamente. Figura retirada do Pereira, Francisco. (2010). BIOCHEMISTRY IN A CHEMICAL APPROACH. 10.13140/RG.2.2.17251.14889.

- Modelo do Conjunto Conformacional (Conformational Ensemble): Modelo mais recente que considera as proteínas como um conjunto de estados conformacionais pré-existentes. A plasticidade proteica permite a transição entre esses estados, sendo a ligação com o ligante um processo de seleção dentro desse conjunto. [7]

Esses modelos não são mutuamente exclusivos, mas focam em aspectos diferentes e complementares do processo de reconhecimento, sendo o modelo de "Ajuste Induzido" o mais comumente aplicado em programas de docking modernos para explicar a flexibilidade molecular. [15]

2.1 Componentes Metodológicos do Docking Molecular

Um experimento de docking bem-sucedido apoia-se em dois pilares computacionais: o algoritmo de busca (search algorithm) e a função de pontuação (scoring function).

2.1.1. Algoritmos de Busca (Search Algorithms)

O algoritmo de busca é a parte responsável por explorar o espaço conformacional que incluem orientações, translações e conformações do ligante, e gerar diversas poses (modos de ligação) candidatas. Dada a vastidão desse espaço, diferentes estratégias foram desenvolvidas:

- Algoritmo de Correspondência Geométrica (Matching Algorithm): Trata as moléculas como corpos rígidos e busca complementaridade de forma. É a base, por exemplo, do programa DOCK, que utiliza esferas para mapear a superfície do sítio de ligação e do ligante.[16]
- Construção Incremental (Incremental Construction): O ligante é fragmentado e reconstruído passo a passo dentro do sítio ativo. É utilizado por programas como FlexX e DOCK 4.0. [17]
- Algoritmo Genético (Genetic Algorithm): Aplica princípios de evolução (mutação, cruzamento e seleção) para "evoluir" poses em direção a um mínimo de energia global. É empregado pelos softwares GOLD e AudoDock 4. [23]
- Algoritmo de Monte Carlo (Monte Carlo): Gera mudanças aleatórias nos graus de liberdade do ligante, aceitando ou rejeitando novas poses com base em probabilidades termodinâmicas. É usado pelo AutoDock Vina e ICM. [18]

2.1.2 Funções de Pontuação (Scoring Functions)

Após a geração das diversas possíveis conformações (poses), a função de pontuação avalia e classifica cada complexo proteína-ligante predito, estimando uma aproximação da energia livre de ligação (ΔG). Cada programa que utiliza energia livre faz aproximações com o intuito de acelerar os cálculos tentando perder o mínimo de precisão. Uma função robusta deve ser capaz de ranquear corretamente as poses e discriminar ligantes ativos dos inativos. As principais categorias são [19]:

- Baseadas em Campo de Força (Force Field-based): Calculam a energia de interação usando termos eletrostáticos e de van der Waals, derivados de campos de força como AMBER e CHARMm. Exemplos: AutoDock, DOCK. [18, 16]
- Empíricas (Empirical): Utilizam parâmetros derivados de regressões lineares a partir de dados experimentais de estruturas conhecidas. Exemplos: ChemScore, GlideScore (SP/XP). [19]
- Baseadas em Conhecimento (Knowledge-based): Empregam potenciais estatísticos inferidos da análise da frequência de contatos atômicos em um

grande banco de dados de complexos, como o Protein Data Bank. Exemplos: PMF, DrugScore. [7]

2.2. Softwares e Aplicações Práticas

A escolha do programa de docking depende do objetivo do estudo, pois cada software possui algoritmos e funções de pontuação distintas. Estudos comparativos indicam que programas como GLIDE, GOLD e SURFLEX estão entre os que apresentam maior precisão na predição de poses.[20]

Tabela 1. Tipos de softwares de Docking Molecular.

Software	*Tipo de Licença*	*Algoritmo de Busca Principal*	*Função de Pontuação*	*Função E_{bind}*
AutoDock / Vina	Código Aberto	Algoritmo Genético, Monte Carlo	Empírica	Semi-empirical Free Energy Dorce Field/ Vina Score
DOCK6	Acadêmica Gratuita	Correspondência Geométrica	Campo de Força, empírica	Grid Score / Amber Score
GOLD	Comercial	Algoritmo Genético	Empírica (GoldScore, ChemScore)	GoldScore, ChemScore, ASP, ChemPLP
Glide 14	Comercial	Busca Exaustiva	Empírica (SP, XP)	ClideScore (SP eXP)
FlexX	Comercial	Construção Incremental	Empírica	FlexX Score (Böhm adaptation)
MOE	Comercial	Múltiplos Algoritmos	London ΔG, Affinity ΔG	London ΔG, GBVI/WSA ΔG
DockThor	Gratuito (servidor Web)	Algoritmo próprio (MF-score, IF-score)	Empírica	DockTScore (MF0score, IF-score)
LeDock	Acadêmica Gratuita	Algoritmo de Monte Carlo	Empírica (simplificado)	LeDock Score
rDock	Código Aberto	Algoritmo Genético	Empírica (ChemScore, SF)	RDock Score(SF), ChemScore
SwissDock	Gratuito (Servidor Web)	Algoritmo de Busca Global	Campos de Forca CHARMM	FullFitness (CHARMm)

Surflex	Comercial	Fragmentação e Construção Incremental baseado em "Protomol"	Empírica (Protomol-Based Scoring)	Mammerhead / Protomol-Based

As aplicações do docking são vastas na descoberta de fármacos como a otimização de compostos Líderes (Lead Optimization), o estudo de estruturas e mecanismos de ação de fármacos. Uma outra aplicação de docking é a Triagem Virtual (Virtual Screening) que permite o rastreamento computacional de grandes bibliotecas de compostos (milhões de unidades moleculares) contra um alvo proteico, identificando moléculas promissoras ("hits") de forma rápida e econômica, servindo como alternativa ou complemento a ensaios experimentais de alto rendimento (HTS) [17]. Otimização de Ligantes (Lead Optimization), análise detalhada do modo de ligação (pose) de um ligante líder permite aos químicos medicinas modificar sua estrutura racionalmente para melhorar a afinidade, seletividade e propriedades farmacocinéticas. Estudos de mecanismos de ação, o docking ajuda a elucidar mecanismos moleculares de inibição enzimática e a compreender as bases estruturais de interações proteína-ligante.[22]

3 PROCEDIMENTOS EXPERIMENTAIS

Testar o funcionamento e as limitações do Raccoon, dispondo de uma larga base de dados de ligantes (303429 moléculas, que foram retiradas do banco de dados do ChEMBL), em um alvo enzimático. Medir e avaliar o tempo de cálculo que o Raccoon leva em uma base de dados menor (de 156 ligante) com potencial de inibição frente a duas enzimas: a BCR-ABL T315I (código 3IK3) e a hAChE (código 6O4W) ambas depositadas no **Protein Data Bank**.

Os cálculos foram realizados em 2 computadores sendo que um deles tem um processador AMD Ryzen 7 2700X com 8 core de 3.70 GHz, 64,0 GB de memória RAM, uma placa de vídeo NVIDIA GeForce GTX 1060 3 GB (computador-1). O outro computador tem processador AMD Ryzen 9 5950X 16 de core e 3.40 GHz, 128 GB de memória RAM e uma placa de vídeo NVIDIA GeForce RTX 3060 12GB (computador-2). O sistema operacional usado em ambas as máquinas foi o Windows com o

WSL usando uma distribuição do Ubuntu. Utilizando o AutoDock 4.2.26 para os dockings que usaram CPU e o AD-GPU para os cálculos feitos com a GPU.

Como pré-requisito de hardware para o AD-GPU:

Tabela 2: Requisitos AD-GPU.

Sistema Operacional	*CPU*	*GPU*
CentOS 6.7 & 6.8 / Ubuntu 14.04 & 16.04	Intel SDK for OpenCL 2017	OpenCL / CUDA >= 11
macOS Catalina 10.15.1	Apple / Intel	Apple / Intel Iris, Radeon Vega 64, Radeon VII

Estes sistemas e configurações foram apenas as listadas do GitHub deles, como está relatado outros sistemas e configurações provavelmente funcionam, apenas não foram testadas.

Por ser uma ferramenta de linha de comando (CLI), sem interface gráfica, todo o processo de instalação, configuração e execução do AD-GPU depende de comandos no terminal. O processo de instalação pode variar conforme as especificações do sistema.

O procedimento de instalação para hardware *Nvidia* consiste na instalação prévia do CUDA e do CUDA Toolkit. Em seguida, é necessário definir as variáveis de ambiente *GPU_INCLUDE_PATH* e *GPU_LIBRARY_PATH* para os diretórios corretos e, por fim, compilar o código-fonte do programa.

Após a compilação, o executável (ex: *autodock_gpu_128wi*) localiza-se no diretório AutoDock-GPU/bin/. A sintaxe básica de execução é a seguinte:

```
./bin/autodock_<tipo>_<N>wi

--ffile .maps.fld

--lfile .pdbqt
```

```
--nrun
```

Onde <tipo> refere-se a gpu ou cpu, e <N> (ex: 128) indica o tamanho do grupo de trabalho (work-group). A ferramenta permite a inclusão de múltiplos argumentos, como a importação de listas de ligantes, a utilização de arquivos de parâmetros “file.dpf” e a alteração de parâmetros do algoritmo genético local (LGA), como o tamanho da população.

Ao término da simulação, os resultados são gerados nos formatos “.xml” e “.dlg”. A análise e interpretação desses arquivos de saída requerem o uso de software adicional, como o AutoDock Tools ou programas de visualização molecular.[10]

Foi testado e instalado o AD-GPU em ambiente WSL e as limitações que ele possui. Foi verificado o tempo de cálculo que o AD-GPU leva em uma base de dados de 156 ligantes sendo eles e 2 proteínas a 3ik3-B e a hAChE em computadores Linux/WSL (computador 1 e 2).

Foi desenvolvido um Workflow que utilize o Raccoon e o AD-GPU com o foco em praticidade e tempo de cálculo da máquina. Para facilitar a usabilidade foi riado um programa em python que liste os ligantes e mapas para o AD-GPU.

O protocolo de docking modelo para o estudo que foi usado como em todos os cálculos foi com ligantes flexíveis, o receptor 3IK3rígido e o receptor hAChE (com flexibilização da cadeia lateral do aminoácido Tyr 337). O GridBox da proteína 3IK3 gerado tem dimensões X, Y e Z: 60, 80 e 50 pontos respectivamente, com espaçamento de 0,375 Å e centrado nas coordenadas de 4,652 Å; 31,172 Å e 15,575 Å. Já para a proteína o GridBox hAChE foi desenhado nas dimensões X, Y e Z, 68, 76 e 54 pontos respectivamente, com espaçamento de 0,375 Angstrom e centro X, Y e Z de –2,554 Å; 113,497 Å e 28,759 Å. O número de indivíduos por população foi de 150, o número máximo de avaliações energéticas de 2500000, o número de gerações foi de 27000, para ambas as proteínas.

4 RESULTADOS E DISCUSSÃO

O workflow proposto neste trabalho é dividido em 5 partes, sendo elas:

I. Preparo do arquivo “.gpf”.
II. Processamento dos ligantes, receptores e a criação dos mapas.
III. Criação da Lista com os endereços dos ligantes e dos mapas.
IV. Cálculo dos estudos de dockings
V. Análise dos resultados

As etapas I pode ser feita no ADT seguindo o processo padrão descrito no manual do ADT. Esta escolha foi feita pois ele é capaz de ajudar visualmente na criação do gridbox.

4.1. Raccoon

O Raccoon apresentou um limite no computador-1 de 5939 ligantes, sendo o limitador o espaço de armazenamento do computador.

Neste protocolo o Raccoon agiu como um organizador e processador tanto dos ligantes quanto das proteínas e mapas.

Usando a aba “Ligands” (Figura 2) para configurar a flexibilidade e a carga dos ligantes, este passo tem que ser feito antes de selecionar as moléculas para aplicar as mudanças. A aba “Receptors” (Figura 3) para selecionar e configurar a(s) proteína(s). A aba “Maps” (Figura 4) para pre-calcular os arquivos mapas. A aba “Doking” (Figura 5) neste protocolo não possui utilidade, pois o docking será feito utilizando o AD-GPU. Na aba “VS Generation” é escolhida a pasta vazia “resultado-rAD4”. Após todas as etapas basta clicar no botão “Generate” (Figura 6).

4.2. Listador

Para trabalhar com o AD-GPU de maneira eficiente foi preciso usar a flag “*--filelist*” que permite usar uma lista com os endereços dos mapas e dos ligantes. Como o Raccoon organiza os ligantes e mapas em uma mesma pasta foi criado um programa em python que pede o endereço onde das pastas criadas pelo Raccoon e gera a lista automaticamente, a lista é criada na pasta indicada. Assim faltando apenas mover a lista para a pasta bin do AutoDock-GPU.

A estrutura que o listador segue para criar as listas é sempre primeiro o arquivo “.*maps.fld*" em seguida dos ligantes e seus respectivos nomes, porém devido a essa estrutura simples ele ainda não é capaz de criar uma lista com mais de um re-

ceptor pois ele irá colocar os mapas em sequência quando o desejado seria colocar um mapa primeiro seguido pelos ligantes e depois colocar outro mapa seguido dos mesmos ligantes.

4.3. AD-GP

O AD-GPU se demonstrou extremamente útil principalmente quando usado com uma lista com vários ligantes, esta lista tem que ser um arquivo de texto que indique primeiro o endereço do arquivo *"maps.fld*", nas próximas linhas os endereços dos ligantes e os respectivos nomes de saída.

A capacidade de aceleração do tempo de cálculo do AD-GPU é muito bem documentada e de acordo com os dados apresentados na tabela 2 [6]

Tempo de cálculo CPU e GPU:

Tabela 3: Tempo (em minutos) de cálculo dos 156 ligantes nas 2 proteínas e nos 2 computadores, cálculos feitos na GPU e na CPU.

Tempo	*Computador-1* *3ik3*	*Computado-1* *hAChE*	*Computador-2* *3ik3*	*Computador-2* *hAChE*
CPU	4981	4130	4805	4011
GPU	38,05	21,81	4,85	2,78

4.3.1 Usando o AD-GPU

Para tirar melhor proveito da automação do AD-GPU se faz necessário um arquivo que liste os ligantes e os mapas. Para isso foi criado um programa em python que pede de entrada um caminho para o diretório onde se encontra as pastas criadas pelo Raccoon e nele são identificados todos os arquivos "*.pdbqt*" e o arquivo "*maps.fld*", é escrito uma lista de acordo com os padrões pedidos pelo AD-GPU. O arquivo da lista é gerado na pasta indicada anteriormente, apenas tendo que ser movida para a pasta *do /home/usuario/~/AutoDockGPU/bin* e então executado com o comando padrão do AD-GPU como base.

4.4 Raccoon+AD-GPU

Para trabalhar com os 2 programas foi utilizado a parte da interface do Raccoon e a capacidade de cálculo do AD-GPU, o Raccoon continua fazendo a etapa de organizar e processar as moléculas de acordo com as configurações do usuário. Ele também foi usado para calcular os mapas, e o AD-GPU irá usar a estrutura criada pelo Raccoon indicada na lista criada pelo listador para calcular os dockings.

A junção do Raccoon e do AD-GPU em um único ambiente computacional foi arquitetada em parceria com estudante de engenharia elétrica da UnB, para tal feito foi usado o Docker para padronizar a instalação e o ambiente com o WebTop que é capaz de simular um ambiente Linux usando o navegador de internet como plataforma, porém sem depender de conexão com a internet. Junto do arquivo para criar a imagem tem a estrutura de pastas para a entrada de ligantes, proteínas e do arquivo ".gpf", a pasta de resultados do Raccoon e do AD-GPU, o pacote MGLtools e o listador.

Para criar a imagem do contêiner docker foi usado o comando "*docker build –t raccoon-webtop*." no software Docker. Após a criação da imagem foi usado iniciador universal em seguida uma janela do navegador padrão é aberta com uma área de trabalho Linux e o Raccoon já aberto na janela dos ligantes. Foi mantido o procedimento padrão do Raccoon de adicionar ligantes, proteína, grid, configurações do "*.dpf*", indicação do local. Ao final quando a janela do Raccoon for fechada ele abria automaticamente a janela do AD-GPU, que serve para construir de maneira gráfica a linha de comando do AD-GPU. O listador age assim que o Raccoon for fechado, criando a lista já na pasta padrão "resultadosAD4".

As pastas que são de acesso compartilhado pelo docker e pelo computador são estruturadas de maneira a ter uma entrada, onde são colocados os ligantes, proteínas e o arquivo "*.dpf*". Uma pasta de saída onde é criado os arquivos do Raccoon, a lista e uma pasta para os resultados do AD-GPU.

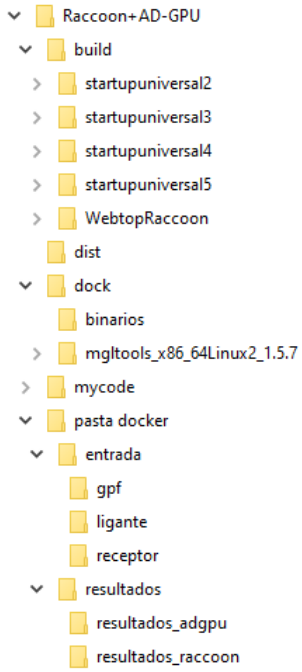


Figura 10: estrutura de pastas criadas pelo Raccoon+AD-GPU.

5 CONSIDERAÇÕES FINAIS OU CONCLUSÃO

O pacote de docking e virtual screening usando o Raccoon e o ADT-GPU desenvolvido é capaz de melhorar o tempo dos cálculos de docking molecular e bem como a acessibilidade para usuários inexperientes em computação, como demonstrados nos tempos de cálculos, onde o cumputador 1 foi capaz de reduzir o tempo de cálculo de milhares de minutos para algumas dezenas e o cumputador 2 reduziu de milhares de minutos para alguns minutos, para 156 ligantes os 2 computadores foram capazes de reduzir o tempo de docking e no manual do aplicativo. A junção do Raccoon com o AD-GPU usando o docker se mostrou capaz de manter a praticidade do Raccoon e a velocidade de conta do AD-GPU.

O listador não apresentou problemas para criar listas com 1 receptor, porém ele ainda é incapaz de gerar elas para vários receptores em um único arquivo. A estrutura de pastas pré-definidas precisa que o usuário sempre esvazie a pasta dos resultados do Raccoon.

BIBLIOGRAFIA

1. CHAUDHARY, K. K.; MISHRA, N. A review on molecular docking: novel tool for drug discovery. **JSM Chemistry**, [S. l.], v. 4, n. 3, p. 1029, 2016.
2. AGU, P. C. *et al*. Molecular docking as a tool for the discovery of molecular targets of nutraceuticals in diseases management. **Scientific Reports**, [S. l.], v. 13, p. 13745, 2023.
3. SANTOS-MARTINS, D. S. *et al*. Accelerating AutoDock4 with GPUs and Gradient-Based Local Search. **Journal of Chemical Theory and Computation**, [S. l.], v. 17, p. 1060–1073, 2021.
4. BISSANTZ, C.; FOLKERS, G.; ROGNAN, D. Protein-based virtual screening of chemical databases. 1. Evaluation of different docking/scoring combinations. **Journal of Medicinal Chemistry**, [S. l.], v. 43, n. 25, p. 4759–4767, 2000.
5. MOLECULAR Docking: Challenges, Advances and its Use in Drug Discovery Perspective. **Current Drug Targets**, [S. l.], v. 20, n. 5, p. 501–521, 2019.
6. JONES, G. et al. Development and validation of a genetic algorithm for flexible docking. Journal of Molecular Biology, [S. l.], v. 267, n. 3, p. 727–748, 1997.
7. PINZI, L.; RASTELLI, G. Molecular docking: shifting paradigms in drug discovery. **International Journal of Molecular Sciences**, [S. l.], v. 20, p. 4331, 2019.
8. MENG, X. Y. *et al*. Molecular docking: A powerful approach for structure-based drug discovery. **Current Computer-Aided Drug Design**, [S. l.], v. 7, p. 146–157, 2011.
9. MORRIS, G. M. et al. AutoDock4 and AutoDock Tools4: automated docking with selective receptor flexiblity. Journal of Computational Chemistry, [S. l.], v. 30, n. 16, p. 2785–2791, 2009.
10. THE SCRIPPS RESEARCH INSTITUTE. **AutoDock 4**. Disponível em: https://autodock.scripps.edu. Acesso em: 28 nov. 2025.
11. THE SCRIPPS RESEARCH INSTITUTE. **AutoDock-GPU**. 2021. Disponível em: https://github.com/ccsb-scripps/AutoDock-GPU. Acesso em: 28 nov. 2025.
12. MICROSOFT. **Windows Subsystem for Linux**. 2024. Disponível em: https://learn.microsoft.com/pt-br/windows/wsl/about. Acesso em: 28 nov. 2025.zzz
13. MICROSOFT. **O que é o Docker?**. 2024. Disponível em: https://learn.microsoft.com/pt-br/dotnet/architecture/microservices/container-docker-introduction/docker-defined. Acesso em: 28 nov. 2025.
14. HALPERIN, I. et al. Principles of docking: An overview of search algorithms and a guide to scoring functions. Proteins, [S. l.], v. 47, p. 409–443, 2002.
15. KOSHLAND JR, D. E. Correlation of structure and function in enzyme action. **Science**, [S. l.], v. 142, p. 1533–1541, 1963.

16. KUNTZ, I. D. *et al*. A geometric approach to macromolecule–ligand interactions. **Journal of Molecular Biology**, [S. l.], v. 161, p. 269–288, 1982.
17. RAREY, M. et al. A fast flexible docking method using an incremental construction algorithm. Journal of Molecular Biology, [S. l.], v. 261, n. 4, p. 470–489, 1996.
18. TROTT, O.; OLSON, A. J. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading. Journal of Computational Chemistry, [S. l.], v. 31, p. 455–461, 2010.
19. FRIESNER, R. A. et al. Glide: a new approach for rapid, accurate docking and scoring. 1. Method and assessment of docking accuracy. Journal of Medicinal Chemistry, [S. l.], v. 47, n. 7, p. 1739–1749, 2004.
20. WANG, Z. et al. Comprehensive evaluation of ten docking programs on a diverse set of protein-ligand complexes: the prediction accuracy of sampling power and scoring power. Physical Chemistry Chemical Physics, [S. l.], v. 18, p. 12964–12975, 2016.
21. JOURNAL OF MOLECULAR DOCKING. **Journal of Molecular Docking**, [S. l.], v. 2, n. 2, 2022.
22. KITCHEN, D. B. et al. Docking and scoring in virtual screening for drug discovery: Methods and applications. **Nature Reviews Drug Discovery**, [S. l.], v. 3, n. 11, p. 935–949, 2004.