AUTOMODEL 0.5, UMA INTERFACE GRÁFICA PARA MODELAGEM DE PROTEÍNAS POR HOMOLOGIA: NOVO SUPORTE A REFINAMENTO DE LOOPS E ALINHAMENTO

JOÃO LUIZ DE ALMEIDA FILHO

UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO

CAMPOS DOS GOYTACAZES, RJ
MARÇO 2015

AUTOMODEL 0.5, UMA INTERFACE GRÁFICA PARA MODELAGEM DE PROTEÍNAS POR HOMOLOGIA: NOVO SUPORTE A REFINAMENTO DE LOOPS E ALINHAMENTO

JOÃO LUIZ DE ALMEIDA FILHO

Dissertação apresentada ao Centro de Biociências e Biotecnologia da Universidade Estadual do Norte Fluminense, como parte das exigências para obtenção do título de Mestre em Biociências e Biotecnologia.

Orientador: Dr. Jorge Hernandez Fernandez

UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE DARCY RIBEIRO

CAMPOS DOS GOYTACAZES, RJ
MARÇO 2015

João Luiz de Almeida Filho

AutoModel 0.5, uma interface gráfica para modelagem de proteínas por homologia: Novo suporte a refinamento de loops e alinhamento

Dissertação apresentada ao Centro de Biociências e Biotecnologia da Universidade Estadual do Norte Fluminense, como parte das exigências para obtenção do título de Mestre em Biociências e Biotecnologia.

Trabalho aprovado. Campos dos Goytacazes, RJ, 30 de Março de 2015:

Dr. Jorge Hernandez Fernandez
Orientador (UENF)

Dr. Alexandre Rossi Paschoal
Professor (UTFPR)

Dra. Annabell Del Real Tamariz
Professora (UENF)

Dr. Thiago Motta Venâncio
Professor (UENF)

Campos dos Goytacazes, RJ
Março 2015

Aos meus pais, irmão e toda minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida.

AGRADECIMENTOS

Primeiramente agradeço a Deus e a minha família, especialmente meus pais, por me apoiarem em todas as etapas na minha vida. Agradeço, também, a UENF e aos meus professores da pós-graduação por me fornecerem as ferramentas necessárias para o meu desenvolvimento acadêmico e futuramente profissional; ao meu orientador Jorge Hernandez por me apresentar e acreditar em mim no mundo da bioinformática; aos meus amigos e colegas de laboratório: Francielle, Carlos Eduardo, Heitor e Paula, que me apoiaram e me motivaram durante o meu mestrado. A todos vocês: muito obrigado, pois, sem vocês não chegaria até aqui.

SUMÁRIO

RESUMO

A modelagem molecular é uma vertente em amplo desenvolvimento dentro da bioinformática que está cada vez mais sendo utilizada em várias áreas como, por exemplo, na farmacologia. No entanto, os *softwares* de modelagem possuem uma interface de usuário complexa, dificultando a prática cotidiana de biólogos estruturais e bioquímicos, usuários não experientes e estudantes. Com o objetivo de facilitar a experiência dos pesquisadores com este tipo de modelagem foi desenvolvido o AutoModel, um serviço *on-line* semiautomático que permite ao usuário fazer modelagens com um maior controle nos parâmetros e com uma interface intuitiva. A peça central do AutoModel é o *software* Modeller, uma ferramenta de modelagem já conhecida no meio acadêmico. Como um diferencial do AutoModel a outros sistemas *on-line* de modelagem, foi adicionado um módulo de refinamento de loops, além de um novo sistema de alinhamento baseado no HMMER e novo sistema de comunicação.
Palavras-chaves: Bioinformática. Modelagem por homologia. AutoModel.

ABSTRACT

The molecular modeling is a bioinformatic research line in extensive development. It's use in increasing in many areas, such as, pharmacology. Still, the modeling softwares are complex and have a difficult user interface, for the daily practice of structural biologists and biochemists, not experienced users and students. In order to facilitate researcher's experience with this type of modeling we developed the AutoModel, a semi-automatic on-ine service that allows the user to perform the modeling with greater control of parameters and an intuitive interface. The engine of AutoModel is the Modeller software, a well-known modeling tool inside the academic environment. As a differential of AutoModel to other online modeling systems, a refinement loops module was added, together with a new alignment system based on HMMER and new communication system.

Key-words: Bioinformatic. Homology Modeling. AutoModel.

LISTA DE ABREVIATURAS E SIGLAS

BLAST	*Basic Local Alignment Search Tool*
DOPE	*Discrete Optimized Protein Energy*
FASTA	*FAST-All*
HPC	*High-performance computing*
INPI	Instituto Nacional da Propriedade Industrial
IP	*Internet Protocol* - Protocolo de Internet
LDH	Lactato Desidrogenase
LQFPP	Laboratório de Química e Função de Proteínas e Peptídeos
MDH	Malato Desidrogenase
MULTALIN	*Multiple sequence Alignment*
MUSCLE	*Multiple Sequence Comparison by Log-Expectation*
PDB	Protein Data Bank
Prosa-web	*Protein Structure Analysis*
RMN	Ressonância Magnética Nuclear
RPyC	Remote Python Call
VMD	*Visual Molecular Dynamics*

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

1 INTRODUÇÃO

1.1 BIOINFORMÁTICA

A bioinformática é um campo da ciência que emprega técnicas e ferramentas computacionais para o estudo de problemas e questões biológicas (ATTWOOD et al., 2011). Embora este campo seja relativamente novo, na década de 60, Levinthal já havia utilizado um software de computador para visualizar a estrutura de uma proteína. A biologia computacional teve uma maior importância no final da década de 90 devido ao projeto genoma e o aumento do poder computacional disponível (HOGEWEG, 2011; OUZOUNIS, 2012).

Os experimentos *in sílico*, como são chamados os experimentos utilizando o computador, possuem, em sua maioria, programação como parte da metodologia. Estes experimentos normalmente buscam tratar um grande volume de dados para a identificação de genes, resolução de estrutura tridimensional de proteínas, estudos com ligantes, montagem de árvores filogenéticas, sequênciamento de genomas, além de outras aplicações (GILISSEN et al., 2012; MOULT et al., 2014; CHANG; TOMMASO; NOTREDAME, 2014).

Além disso, a bioinformática é dependente da computação de alto desempenho, pois, várias aplicações necessitam de grande poder de processamento, sendo muitas vezes necessária a utilização de *grids* de computadores ou outras tecnologias de computação de alto desempeho (HPC - *High-performance computing*), como a utilização de *arrays* de placas de vídeos e servidores com vários processadores (MAREUIL et al., 2011; YANG et al., 2014).

Embora a bioinformática possua uma raiz acadêmica, ela tem auxiliado a indústria farmacêutica. A utilização das ferramentas de modelagem e caracterização da estrutura terciária de proteínas auxilia o desenho e planejamento de novas moléculas que podem ser utilizadas como fármacos (DU; XIE; HUANG, 2014).

1.2 PREDIÇÃO DA ESTRUTURA TRIDIMENSIONAL DE PROTEÍNAS ATRAVÉS DA MODELAGEM POR HOMOLOGIA

Caracterizar a estrutura e função de macromoléculas biológicas é de vital importância para a biologia. Exemplos destas macromoléculas são os ácidos nucleicos que contém a informação necessária para a regulação das funções na célula, carboi-

dratos que podem possuir função energética ou estrutural, e as proteínas que estão envolvidas em praticamente em todos processos celulares (VERLI, 2014).

As proteínas, para facilitar o estudo, a sua estrutura é dividida em 4 níveis hierárquicos de organização estrutural (figura 1) (MURRAY et al., 2003). Isto é, um nível superior depende da informação do nível inferior.

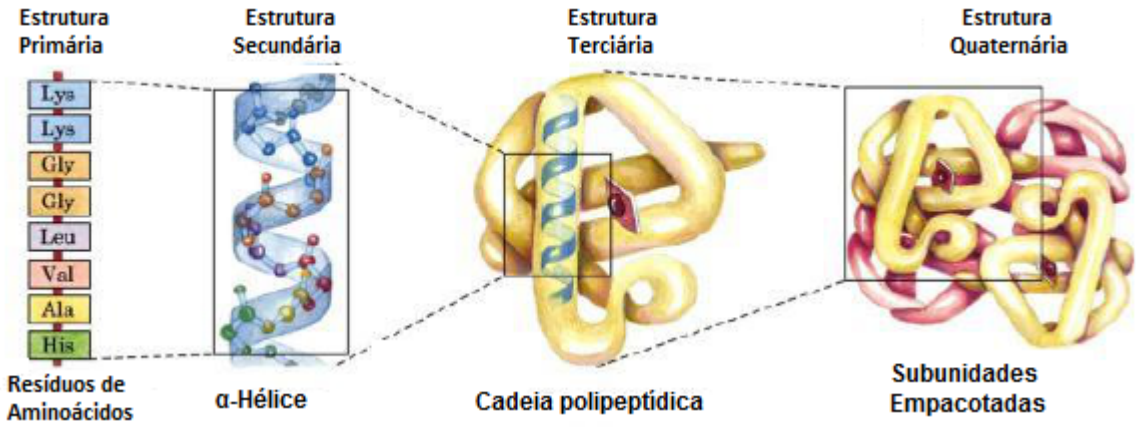


Figura 1 – Os 4 níveis de organização estrutural de uma proteína. Modificado de Lehninger, Nelson e Cox (2005)

O primeiro nível de organização, conhecido como estrutura primária, consiste em uma fita unidimensional de resíduos de aminoácidos que compõe uma proteína. Embora em menor nível de complexidade, ela fornece uma grande quantidade de informações sobre a estrutura e função da proteína. Estas informações normalmente são obtidas ao se buscar padrões na estrutura primária de diferentes proteínas (GERALD, 2005). Estes padrões podem estar associados a características funcionais ou estruturais, e podem ser utilizados no estudo de outras proteínas que possuem pouca informação estrutural ou funcional (VERLI, 2014). Além disto, a determinação destes padrões permite o estudo da evolução de famílias de proteínas e organismos (DAVID, 2001). Normalmente, cada resíduo que compõe a estrutura primária é representado por 1 ou 3 letras.

Alguns padrões de aminoácidos na estrutura primária formam estruturas espaciais denominadas estruturas secundárias. Esta estrutura espacial é obtida quando acontecem interações entre os resíduos destas sequências e o meio . Por exemplo, as cadeias laterais de cisteínas tendem a formar pontes dissulfeto, e em proteínas globulares, os resíduos polares tendem expor suas cadeias laterais em meio aquoso, enquanto que os resíduos apolares tendem a se agrupar maximizando as interações hidrofóbicas e expulsando a água do seu centro.

É relativamente pequena a quantidade de estruturas secundárias conhecidas. Além disto, uma estrutura secundária pode ser formada por padrões diferentes de sequências de resíduos. As estruturas secundárias mais frequentes são as alfa-hélices, folhas-beta e *loops*. Comparadas com as duas primeiras, os *loops* são estruturas

comumente mais flexíveis, podendo possuir várias conformações, alterando inclusive a posição e orientação de outras estruturas secundárias na proteína. Além disso, os *loops* são as regiões mais suscetíveis à mutações, isto é, são suscetíveis a adição, substituição ou deleção de resíduos por outro, podendo acarretar na alteração de sua flexibilidade ou função.

As estruturas secundárias contidas em uma proteína se organizam na estrutura terciária através de um processo de enovelamento. Este processo, é uma combinação de forças que convergem para que a biomolécula adote uma conformação mais estável no meio biológico alvo. Na estrutura terciária, há uma tentativa de aproximação mútua de resíduos que são hidrofóbicos que tentam se esconder da água (colapso hidrofóbico, ocasionando a expulsão deste solvente na região central da proteína). Com isso, os resíduos polares são expostos ao solvente, e interações interresíduos são estabelecidas. Estas interações podem ser tanto de origem covalente como pontes dissulfidicas como interação não covalente. Assim, a estrutura está enovelada, nativa.

A estrutura nativa (terciária) de uma proteína é única, e na maioria dos casos é determinada somente pela sua sequência de resíduos. No processo de enovelamento a estrutura desnovelada da proteína tende a se enovelar em uma estrutura termodinamicamente mais estável que é a sua forma nativa. Isso é conhecido como hipótese termodinâmica de Anfinsen . Toda a informação necessária para o enovelamento está contido nos aminoácidos que constituem a estrutura primária da proteína. Ou seja, é a totalidade das interações entre os aminoácidos que constituem a estrutura primária que determina a estrutura nativa da proteína.

O último nível de organização, estrutura quaternária são agregados peptídicos, que algumas proteínas podem constituir. E elas podem adotar estados oligoméricos, ou seja por 2 ou mais subunidades necessárias para realizar determinada função (VERLI, 2014).

No entanto, predizer a estrutura terciária de uma proteína não é uma tarefa simples, são utilizados tradicionalmente para este fim métodos *in vitro*. Destes, os mais utilizados são a ressonância magnética nuclear (RMN) e a difração de raios-X. A vantagem na utilização destes métodos é que eles produzem modelos com ótima resolução. No entanto, ambos os métodos possuem algumas desvantagens. A RMN, por exemplo, tem dificuldade de determinar a estrutura de proteínas que possuem mais de 120 resíduos de aminoácidos. E algumas proteínas, como proteínas de membrana, não se cristalizam de forma satisfatória, impossibilitando o uso da difração de raios-X (KRIEGER; NABUURS; VRIEND, 2003; TOPF et al., 2006). Além disso, ambos os métodos são custosos e demandam de tempo, e consequentemente dificultam a aplicação em larga escala.

O sequenciamento da estrutura primária de uma proteína é uma tarefa relativamente mais simples do que a resolução da estrutura terciária pelos métodos *in vitro*

(FILHO; ALENCASTRO, 2003). Além do mais, como consequência do axioma termodinâmico proposto por Anfinsen, a estrutura tridimensional da maioria das proteínas é determinada por sua sequência de aminoácidos e condições ambientais na qual esta sujeita. Assim, métodos de experimentação *in silico* estão sendo desenvolvidos para a determinação da estrutura tridimensional de uma proteína utilizando somente informações de sua sequência de resíduos de aminoácidos. Um exemplo deste tipo de abordagem é a modelagem por homologia. (FLOUDAS, 2007).

1.3 MODELAGEM POR HOMOLOGIA

Utilizando somente o axioma de Anfinsen, o método *ab initio* consegue determinar satisfatóriamente a estrutura tridimensional apenas de peptídeos com menos de 100 resíduos (ROY; KUCUKURAL; ZHANG, 2010; ZHANG, 2014). O motivo para isto, é que não se conhece ainda um algoritmo que seja eficiente na simulação das forças envolvidas no empacotamento e enovelamento das proteínas. Este é um dos maiores desafios da biologia estrutural atualmente (LIWO et al., 1999; KIHARA et al., 2001; ZHANG, 2014; XU et al., 2014).

A modelagem por homologia é outra metodologia *in silico* que tem se mostrado mais eficiente. Assim como a modelagem *ab initio*, esta metodologia tenta resolver a estrutura tridimensional de uma proteína (proteína-*target*) através de sua sequência de resíduos. Contudo, esta metodologia utiliza também informações estruturais de outras proteínas conhecidas como *templates*. Estes *templates* são proteínas que devem possuir duas características: sua estrutura tridimensional deve ser resolvida através de métodos *in vitro* E devem ser homólogas a proteína-*target* (FILHO; ALENCASTRO, 2003). Estas características são utilizadas pelos softwares de modelagem para a criação do modelo tridimensional.

A modelagem por homologia é dividida em 4 etapas distintas: busca e seleção de *template* (proteína molde), alinhamento, modelagem e a validação do modelo gerado (HILLISCH; PINEDA; HILGENFELD, 2004). Normalmente a realização de cada etapa é feita através de um software especifico e resumidamente possui os seguintes passos (figura 2):

Inicialmente é necessária a escolha de uma proteína que será o *template*. Os candidatos a proteínas *templates* normalmente são encontrados em repositórios de estruturas de proteínas, como o PDB (*Protein Data Bank*) (BERMAN; HENRICK; NAKAMURA, 2003). Caso o usuário não conheça nenhuma proteína que possa ser utilizada como *template*, pode-se usar softwares de busca como o BLAST (*Basic Local Alignment Search Tool*) (ALTSCHUL et al., 1990) e o HMMER (FINN; CLEMENTS; EDDY, 2011). Estes programas normalmente pesquisam nestes bancos de dados, e retornam uma lista de possíveis *templates* classificados de acordo com o grau de identidade com

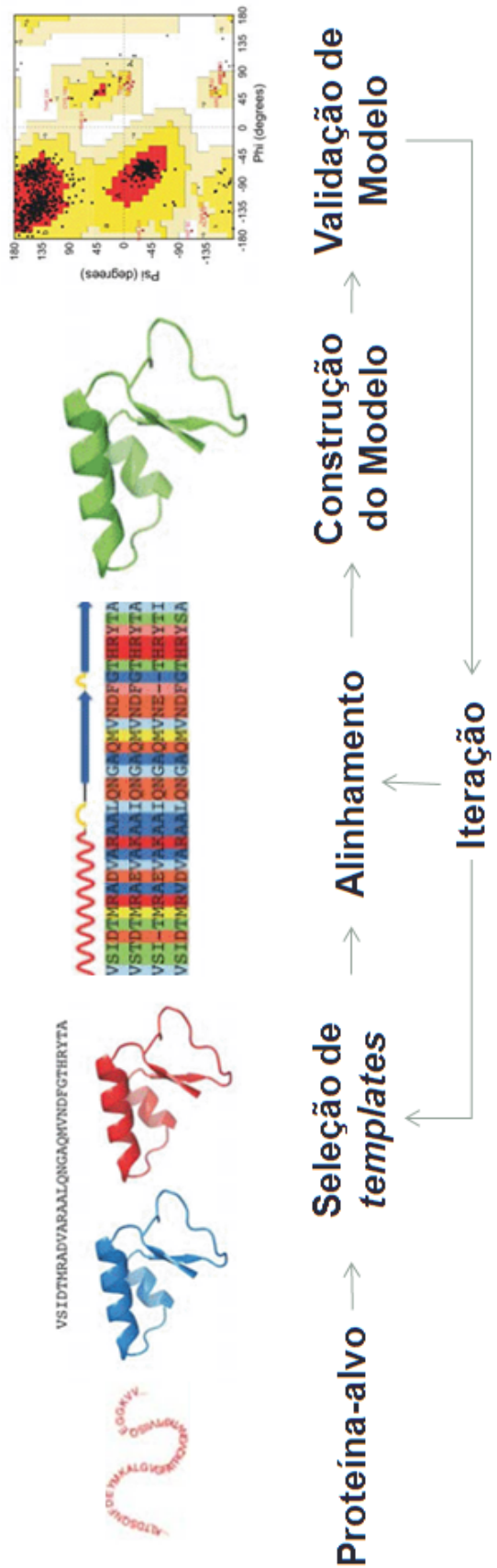


Figura 2 – Algoritmo da modelagem por homologia

a sequência da proteína-*target* (FILHO; ALENCASTRO, 2003).

A segunda etapa corresponde ao alinhamento da sequência do *Template* escolhido com a sequência da proteína-*target*. No alinhamento é possível distinguir as regiões que estão estruturalmente conservadas das regiões variáveis. O software de modelagem utiliza estas informações para a construção do modelo tridimensional. Os softwares de alinhamento mais conhecidos são o BLAST, FASTA (*FAST-All*) (PEARSON; LIPMAN, 1988), CLUSTAL (LARKIN et al., 2007) e o MULTALIN (*Multiple sequence Alignment*) (CORPET, 1988). Esses programas estão disponíveis na internet e são de uso em ambiente acadêmico.

Após o alinhamento, utiliza-se então um software de modelagem para criar o modelo tridimensional da proteína-*target*. Este programa utiliza informações provenientes do alinhamento e da estrutura do *template*. O programa mais conhecido para modelagem é o Modeller (WEBB; SALI, 2014) que utiliza o método de modelagem por restrições espaciais. O modelo gerado pode ser visualizado através de um programa de visualização molecular como o VMD (*Visual Molecular Dynamics*) (HUMPHREY; DALKE; SCHULTEN, 1996) e o PyMol (DELANO, 2002)

Com o modelo pronto é necessário validá-lo, pois, erros podem ter sido introduzidos durante as etapas anteriores da modelagem. Isto pode ser realizado através da avaliação de energia livre por resíduo, torções impróprias de ângulos phi e psi etc (KRIEGER; NABUURS; VRIEND, 2003). Para esta etapa, utilizam-se softwares de avaliação como o PROCHECK (LASKOWSKI et al., 1993) e o MolProbity (CHEN et al., 2009). Esses programas geram relatórios sobre a estereoquímica de modelos. Além disso, alguns programas de modelagem também possuem métricas de avaliação de modelo. Como por exemplo, o Modeller pode gerar gráficos sobre energia de cada resíduo do modelo gerado (ŠALI et al., 1995). Caso, seja encontrado algum problema no modelo, o usuário pode retornar uma ou mais etapas para tentar corrigir algum parâmetro.

A modelagem por homologia pode ser feita com a utilização de softwares específicos para cada etapa, como por exemplo, o HMMER ou o Blast que são softwares que realizam alinhamento e, o PROCHECK que pode ser utilizado para a validação do Modelo. Contudo, com o aumento da demanda por este tipo de modelagem foram desenvolvidas ferramentas que são capazes de realizar a modelagem completa como é o caso do Modeller.

O Modeller é um software que possui um alto custo computacional. Aliado a não possuir uma interface gráfica nativa, dificulta o uso diário e aprendizado de pesquisadores. Neste contexto, foram desenvolvidos vários sistemas de modelagem no qual seu uso é feito pela internet. Exemplo destes softwares são o SWISS-MODEL (SCHWEDE et al., 2003), o ModWeb (ESWAR et al., 2003), Phyre2 (KELLEY; STERNBERG, 2009) e o ESyPred3D (LAMBERT et al., 2002).

Embora estes softwares facilitem o uso diário da modelagem por homologia, eles não conseguem criar modelos mais complexos, como por exemplo, com heteroátomos diferentes daqueles especificados no *template* e no alinhamento. Assim é necessário que o usuário recorra à ferramentas de uso mais complexos como o Modeller.

1.4 MODELLER

O Modeller é uma ferramenta computacional conhecida e utilizada para a modelagem de proteínas por homologia. Ele é capaz de realizar todas as etapas deste tipo de modelagem, sem a necessidade de outras ferramentas. No entanto, sua maior utilização é na etapa de modelagem.

O Modeller guia sua predição com o uso de restrições espaciais da sequência de aminoácidos e ligantes. Estas restrições são obtidas no alinhamento entre a proteína-alvo e a proteína molde e foram escolhidas a partir do trabalho de ŠAli e Overington (1994) no qual foi feita uma análise estatística em um banco de dados de alinhamentos de estruturas homólogas que possuíam a estrutura terciária conhecida. Essas restrições compreendem ângulos diedrais, distâncias entre átomos, regras sobre empacotamento de estrutura secundária, além de outras restrições que são utilizadas pelo Modeller como funções de densidade de probabilidade (PDF - probability density functions). Estas funções são combinadas em uma função objetiva de energia da qual o Modeller tenta minimizar no espaço cartesiano (WEBB; SALI, 2014).

No final da predição, o Modeller retorna com um modelo tridimensional da proteína-alvo em um arquivo no formato PDB (ESWAR et al., 2003), o qual pode ser visualizado através de um visualizador de molecular como o VMD e o PyMOL .

Nativamente o Modeller não possui um ambiente de modelagem gráfico, sendo o controle da modelagem feito através de scripts (Figura 3) escritos na linguagem Python. Estes scripts são arquivos de texto contendo comandos que são executados sequencialmente por este software e, que devem ser alterados pelo usuário. Aliado a isto, a execução deste script é feita através da linha de comando. Isso dificulta o uso cotidiano e a aprendizagem. Contudo, o Modeller fornece a possibilidade de personalizar a modelagem de forma mais efetiva do que outros softwares (ESWAR et al., 2003). Nesse contexto, nós estamos desenvolvendo o AutoModel. Uma nova ferramenta que permitirá o pesquisador a fazer experimentos de modelagens que dificilmente são realizadas nas ferramentas on-line.

```
from modeller import *

env = environ()
aln = alignment(env)
for (pdb, chain) in (('1b8p', 'A'), ('1bdm', 'A'), ('1civ', 'A'),
                     ('5mdh', 'A'), ('7mdh', 'A'), ('1smk', 'A')):
    m = model(env, file=pdb, model_segment=('FIRST:'+chain, 'LAST:'+chain))
    aln.append_model(m, atom_files=pdb, align_codes=pdb+chain)
aln.malign()
aln.malign3d()
aln.compare_structures()
aln.id_table(matrix_file='family.mat')
env.dendrogram(matrix_file='family.mat', cluster_cut=-1.0)
```

Figura 3 – Exemplo de Script do Modeller

1.5 AUTOMODEL

O AutoModel é uma ferramenta on-line que permite ao usuário fazer predições da estrutura tridimensional de proteínas. O AutoModel difere de outras ferramentas por realizar a modelagem de forma semi-automática, isto é, permitindo que o usuário controle parâmetros importantes do processo de predição quando necessário. Além disto, a versão 0.1 do AutoModel conta com a capacidade de escolha e alteração do heteroátomos durante a modelagem, uso de *templates* personalizáveis e relatórios estereoquímicos do modelo predito (FILHO, 2013). Como o AutoModel realiza modelagem por homologia, sua interface foi dividida em 4 seções principais: busca por proteína-molde, alinhamento, modelagem e análise do modelo gerado. Em cada seção existem configurações ou parâmetros que podem ser ajustados de acordo com a necessidade do usuário. Como dado de entrada o usuário carrega a estrutura primária da proteína alvo em um arquivo no formato fasta. Após isto, o usuário tem a opção de carregar uma proteína molde (*template*) ou buscar possíveis candidatos à proteína molde no banco de dados do AutoModel. Antes da etapa de alinhamento, é possível realizar algumas modificações na proteína molde, como por exemplo, escolher qual cadeia e heteroátomos que serão utilizados na modelagem da proteína. Estes heteroátomos poderão ser modificados após o alinhamento e antes da etapa de modelagem. Ao final da predição, o AutoModel permite a avaliação do modelo através de um relatório estereoquímico ou através da análise do gráfico de energia por resíduo chamado de DOPE (*Discrete Optimized Protein Energy*), um potencial estatístico que o motor do AutoModel utiliza para avaliar a energia dos modelos gerados. Neste gráfico, é possível avaliar regiões do modelo estrutural que estão energeticamente desfavoráveis quando comparado com a proteína molde. Esta análise parte do principio termodinâmico que as proteínas tendem a ter a menor energia livre possível (SHEN, 2006).

Internamente o AutoModel utiliza como motor o software Modeller e o banco de dados PDB. O Modeller fornece uma interface de desenvolvimento Python que permitiu

a criação do AutoModel, pois, além de facilitar o desenvolvimento da ferramenta, a linguagem Python possui várias bibliotecas que facilitam a construção de interfaces gráficas. O banco de dados do PDB é utilizado como uma base para *templates*.

Para avaliação do Modelo, o AutoModel utiliza o software PROCHECK. Este programa gera um relatório estereoquímico do modelo gerado. Além do PROCHECK, o AutoModel pode utilizar o próprio Modeller para gerar o gráfico de energia livre por resíduo (DOPE).

1.5.1 Arquitetura cliente-servidor do AutoModel

Para diminuir os custos computacionais inerentes a modelagem por homologia, o AutoModel utiliza a arquitetura distribuída cliente-servidor. Utilizando esta arquitetura, todas as etapas que requerem um alto poder de processamento são realizadas pelo servidor: O AutoModel Server. O AutoModel Server fica localizado em um computador no LQFPP (Laboratório de Química e Função de Proteínas e Peptídeos) na UENF. O AutoModel Server possui todas as ferramentas necessárias para modelagem por homologia. O cliente, o AutoModel Client, é responsável por criar as janelas e receber os comandos dos usuários. Estes comandos são enviados para o AutoModel Server via internet. A arquitetura cliente-servidor também evita que o usuário tenha que instalar as ferramentas utilizadas durante a modelagem em seu computador, e toda comunicação, entre o AutoModel Client e AutoModel Server, é realizada através da internet utilizando a biblioteca RPyC (figura 4).

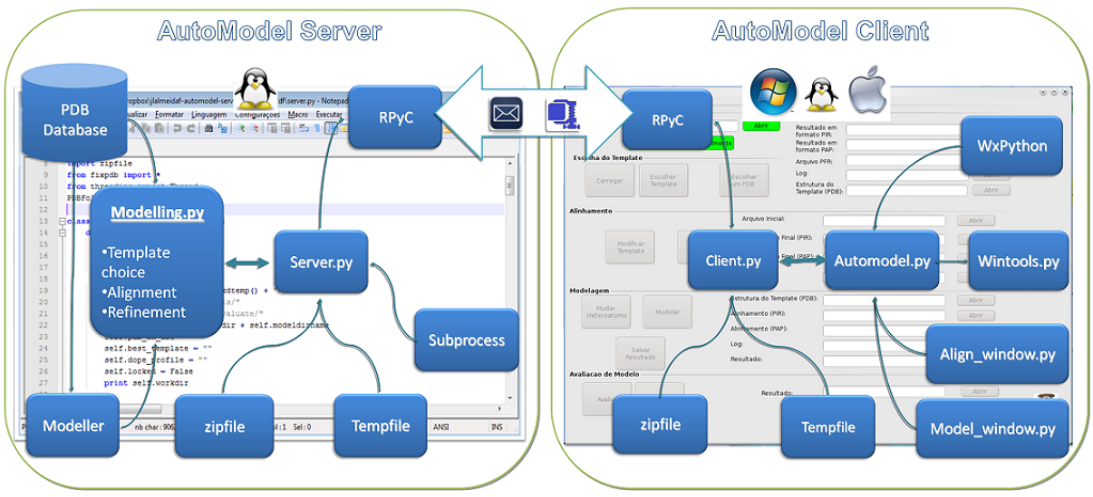


Figura 4 – Comunicação entre o AutoModel Server e AutoModel Client

O AutoModel Server foi desenvolvido de forma a ser modular, permitindo assim adicionar novos módulos conforme a necessidade. Isso significa que todas as etapas da modelagem por homologia são módulos. O módulos do AutoModel necessitam possuir 2 comandos: create_script_in_folder() e run(). Estes 2 comandos são responsáveis por chamar o software Modeller para fazer a modelagem.

1.6 HMMER

HMMER é um pacote de ferramentas computacionais que permite analisar sequências biológicas, que podem ser sequências de resíduos de uma proteína ou de nucleotídeos de DNA, utilizando para isto modelos probabilísticos somados a computação de alta velocidade sem sacrificar o desempenho computacional (EDDY, 2011).

O HMMER pode ser usado para busca por homólogos de proteínas em banco de dados ou ainda para alinhar sequências. Quando utilizado para busca de homólogos membros de uma família de sequências em um banco de dados, o HMMER consegue localizar com maior precisão sequências homólogas distantes do que as ferramentas tradicionais de busca e alinhamento como o BLAST e o FASTA (PEARSON; LIPMAN, 1988).

O HMMER faz o uso de modelos probabilísticos conhecidos como Perfis HMM. Estes perfis contêm informações de cada coluna do alinhamento que são utilizadas para prever quais resíduos são os mais prováveis para estar naquela coluna. Assim o HMMER pode pontuar inserções, deleções, substituições de uma maneira mais efetiva ao utilizar uma base probabilística para estabelecer os valores de pontuação, ao contrário das ferramentas tradicionais de alinhamento e busca de banco de dados que utilizam algoritmos como o Smith/Waterman (SMITH; WATERMAN, 1981) e outras heurísticas que pontuam as inserções, deleções e substituições utilizando pontuação simples (EDDY, 2011).

A versão HMMER3 trouxe avanços sobre o HMMER2 devido a otimizações feitas em seu algoritmo. Esses avanços trouxeram muito mais velocidade, tornando-o tão rápido quanto o BLAST, o que o levou a ser adotado pelos maiores banco de dados de sequências de proteínas como o PFAM (BATEMAN et al., 2004) e o InterPro (FINN; CLEMENTS; EDDY, 2011). No entanto, ao contrário da versão anterior, o HMMER3 só realiza alinhamentos locais (EDDY, 2011). Para o alinhamento de proteínas o HMMER possui os seguintes comandos:

i. HMMBUILD – cria um perfil HMM de um alinhamento múltiplo;

ii. HMMSEARCH – Busca um perfil HMM contra um banco de dados de proteínas, como o PDB por exemplo.;

iii. HMMSCAN – Busca uma sequência em um banco de dados HMM;

iv. PHMMER – Busca uma única proteína em um banco de dados, isto é, tem a mesma função da ferramenta BLASTP. Para fazer a busca a sequência é, internamente, convertida no formato HMM. O usuário pode utilizar uma sequência em formato fasta com o PHMNER;

v. HMMALIGN – Faz um alinhamento de múltiplas sequências contra um perfil HMM criado pelo HMMBUILD;

1.7 MUSCLE

O MUSCLE (*Multiple Sequence Comparison by Log-Expectation*) é uma ferramenta de alinhamento que pode ser utilizada para o alinhamento múltiplo de proteínas e nucleotídeos. Ao realizar um alinhamento o MUSCLE executa um algoritmo que é divido em 3 fases: Inicialmente é feito um alinhamento rápido das sequências. Nesta fase chamada o seu resultado é uma arvore binária. Após isto, é feito uma reestimação deste alinhamento utilizando uma matriz de distancias Kimura, melhorando assim a qualidade do alinhamento. Por final é realizado uma refinamento do alinhamento gerado (EDGAR, 2004b). O tempo de complexidade das duas primeiras fases é O(N2 + NL + L2), sendo a última etapa possuir um tempo de complexidade O(N3L) . Normalmente o Muscle é mais veloz do que outra ferramentas de alinhamentos múltiplos como o Clustal. Produzindo inclusive alinhamento melhores (EDGAR, 2004a). O Muscle é utilizado como plug-ins em várias ferramentas como, por exemplo, o UGENE (OKONECHNIKOV et al., 2012) e no site do EMBL-EBI (GOUJON et al., 2010).

O Muscle está disponível através de seu site, como também um aplicativo compatível com Windows, Linux e Mac OS. Este aplicativo funciona por meio de linha de comando. Principais parâmetros na tabela 1.

Tabela 1 – Lista de parâmetros do MUSCLE

Parâmetro	Detalhes
-in nome_do_arquivo	Arquivo com as sequências para alinhar no formato FASTA.
-out nome_do_arquivo	Nome do arquivo de alinhamento no formato FASTA.
-log nome_do_arquivo	Escreve um arquivo contendo informações sobre o alinhamento.
-version	Mostra versão do Muscle.

1.8 REFINAMENTO DE *LOOPS*

As proteínas dentro de uma família proteica que possuem estruturas semelhantes podem possuir diferenças funcionais. Essas diferenças geralmente estão situadas na superfície das proteínas e nas regiões de *loops*, estruturas proteicas que normalmente ligam outras duas estruturas secundárias (DALL'AGNO; SOUZA, 2013). Estas regiões são mais suscetíveis à deleções, inserções e substituições de resíduos com o mínimo de alterações na estrutura tridimensional da proteína. Além disso, os *loops* podem está intimamente relacionados com a especificidade da proteína. Por exemplo, os *loops* podem está relacionados para os sítios ativo e de ligações ao sítio ativo da

proteína (FISER; DO; ŠALI, 2000). Portanto, é importante que o modelo gerado possua esta região bem modelada.

No entanto, ao se utilizar a modelagem por homologia, as informações utilizadas para criar o modelo são as informações estruturais provenientes da proteína-*template* e do alinhamento. No alinhamento os *loops* normalmente estão situados em regiões de *gaps*, ou seja, trechos do alinhamento representado por traços, significando que nesta região não houve similaridade estrutural entre o *template* e a proteína *target* (FISER; ŠALI, 2003).

Sem uma informação estrutural disponível sobre a região de loop, os softwares de modelagem necessitam utilizar alguma estratégia para a modelagem destas regiões. Essas estratégias vão desde a modelagem *ab initio*, minimização de energia local, métodos probabilísticos até a utilização de banco de dado (KRIEGER; NABUURS; VRIEND, 2003).

O Modeller realiza a modelagem de loop através da minimização da sua função energética. Ele é capaz de modelar *loops* com ligantes e vários *loops* seguidos. Para isso ele tenta utilizar informações estruturais, como ligações químicas e ângulos de ligação, interações entre átomos, e outros parâmetros estereoquímicos (ESWAR et al., 2003; FISER; DO; ŠALI, 2000).

2 JUSTIFICATIVA

O Modeller é um software que realiza modelagem por homologia de forma altamente customizável. A modelagem utilizando este programa é feita através de scripts escritos na linguagem Python (ESWAR et al., 2002). A customização é feita ao se alterar parâmetros destes scripts. Assim seu poder de customização contrasta com a sua espartana interface de usuário, necessitando do usuário um profundo conhecimento da linguagem computacional. Esta é uma característica muito comum em várias outras ferramentas computacionais cientificas e muitas vezes afastam alunos e pesquisadores da experimentação *in silico*.

Assim, para facilitar o uso e aprendizagem destas ferramentas computacionais é necessário muitas vezes o desenvolvimento de interface gráficas mais simples e de fácil manuseio pra estes softwares. Apesar disso, estas interfaces gráficas tendem ser simplistas, não permitindo o usuário fazer a alteração de parâmetros importantes dos softwares que elas controlam.

No caso da modelagem de proteínas por homologia, várias ferramentas estão disponíveis dentre elas o SWISS-MODEL, ModWeb e Phyre2. No entanto, estas ferramentas são simples, ao ponto de não permitir o pesquisador customizar a modelagem de forma que ele gostaria. Nestes casos, o pesquisador necessita usar ferramentas mais complexas como o Modeller. Neste contexto, foi desenvolvido o AutoModel uma ferramenta que permite pesquisadores, de todas partes do mundo, façam modelagens por homologia complexas através de uma interface gráfica amigável.

Neste trabalho, nós atualizamos o AutoModel de forma a permitir que pesquisadores, de todas partes do mundo, façam modelos complexos e mais precisos através do suporte a refinamento de *loop* e um novo módulo de alinhamentos utilizando o HMMER.

3 OBJETIVOS

3.1 OBJETIVO GERAL

O objetivo deste projeto é atualizar e disponibilizar a ferramenta AutoModel, permitindo-o criar modelos mais precisos através de refinamento de *loops* e alinhamento com o HMMER.

3.2 OBJETIVOS ESPECÍFICOS

i. Registrar o AutoModel como uma ferramenta de uso biotecnológico;

ii. Tornar disponível mundialmente o acesso ao AutoModel à pesquisadores, implementando uma arquitetura cliente-servidor;

iii. Implementar módulo de refinamento de *loop*;

iv. Implementar módulo de validação do refinamento de *loop*;

v. Implementar módulo de alinhamento com o HMMER;

vi. Melhorar a qualidade de experiência de uso;

vii. Treinamento de usuários;

4 MATERIAIS E MÉTODOS

4.1 DESENVOLVIMENTO DE NOVOS MÓDULOS

Os novos módulos foram desenvolvidos utilizando a linguagem de programação Python versão 2.7.2 seguindo o guia de codificação PEP8 (PSF, 2013). Este guia contém normas de codificação que facilitam a leitura do código por outras pessoas e também a correção de possíveis erros.

Além disso, seguindo o modelo de desenvolvimento no AutoModel 0.1, todos os módulos desenvolvidos neste trabalho seguiram o modelo de desenvolvimento incremental. O modelo incremental consiste em ciclos de desenvolvimento, onde no final de cada ciclo uma nova versão do programa está disponível contendo uma nova funcionalidade ou a correção de um problema (*bug*). No AutoModel, o controle destas versões é feito por meio do software Git (TORVALDS; HAMANO, 2010). O Git permite o controle das alterações do programa e monta um histórico destas alterações de tal forma que é possível voltar para uma versão anterior do AutoModel caso algo de errado aconteça no desenvolvimento. Além do controle de versões, foi utilizada a ferramenta de testes automatizados.

Teste automatizado consiste em uma ferramenta que realiza testes de forma a garantir que o módulo ou biblioteca que está sendo desenvolvida faça exatamente o que foi projetado durante a etapa de engenharia de software que ocorre no início de cada ciclo do modelo incremental. A utilização de testes é de grande importância, pois, ajuda a evitar e corrigir bugs, além de garantir que a manutenção do código seja feita de forma mais segura, pois, após cada alteração é possível testar se aquela alteração causou uma contradição no código do programa. No AutoModel o software utilizado na realizaçãos destes testes foi o should_dsl (TAVARES, 2012).

Além do Python, no desenvolvimento foi utilizado: a biblioteca RPyC versões 3.2.0 e 3.2.3 para a comunicação entre o AutoModel Client e AutoModel Server, biblioteca WxPython (RAPPIN; DUNN, 2006) versão 3.0 para o desenvolvimento das interfaces gráficas no AutoModel Client, software Modeller versão 9v9 para realização de etapas de modelagem no AutoModel Server.

4.2 DESENVOLVIMENTO DO MÓDULO REFINAMENTO DE *LOOP* NO AUTOMODEL

O módulo de refinamento de *loop* foi desenvolvido de acordo com a seção 4.1. No entanto, no AutoModel Server, o módulo de refinamento de *loop* foi baseado no script

loop_refine.py do tutorial avançado do *software* Modeller (figura 5). A característica principal do loop_refine.py é que ele utiliza o comando loopmodel. Este comando captura do modelo gerado a conformação inicial do *loop* e vai alterando a posição de cada resíduo aleatoriamente em 5Å em todas as direções do plano Cartesiano, Após isto o *loop* é otimizado. Esta otimização inicialmente é feita no *loop* e depois é feita considerando o sistema completo.

```
# Loop refinement of an existing model
from modeller import *
from modeller.automodel import *

log.verbose()
env = environ()

# directories for input atom files
env.io.atom_files_directory = './:../atom_files'

# Create a new class based on 'loopmodel' so that we can redefine
# select_loop_atoms (necessary)
class MyLoop(loopmodel):
    # This routine picks the residues to be refined by loop modeling
    def select_loop_atoms(self):
        # 10 residue insertion
        return selection(self.residue_range('273', '283'))

m = MyLoop(env,
           inimodel='TvLDH-mult.pdb', # initial model of the target
           sequence='TvLDH')          # code of the target

m.loop.starting_model= 1           # index of the first loop model
m.loop.ending_model  = 10          # index of the last loop model
m.loop.md_level = refine.very_fast # loop refinement method; this yields
                                   # models quickly but of low quality;
                                   # use refine.slow for better models

m.make()
```

Figura 5 – *Script* de refinamento de *loops* do Modeller

4.3 DESENVOLVIMENTO DO MÓDULO DE VALIDAÇÃO DO REFINAMENTO DE *LOOP*

Para o desenvolvimento do módulo de validação do refinamento de *loop* foram utilizadas a biblioteca Pylab (HUNTER, 2007) versão 0.98, responsável por plotar o gráfico de qualidade dos modelos gerados. Este módulo também foi baseado no módulo de validação de modelo já desenvolvido para o AutoModel.

4.4 ACESSO EXTERNO AO AUTOMODEL SERVER

Para permitir o acesso externo, o AutoModel Server foi instalado em um servidor IBM x3550 que possui 2 processadores Xeon com quatro núcleos cada um, rodando sob o Ubuntu Linux 12.04 LTS. Esta infraestrutura está localizada no LQFPP (Laboratório de Química e Função de Proteínas e Peptídeos) na UENF e ligado a rede interna da universidade utilizando o endereço de IP : 172.20.2.203.

4.5 SITE DO AUTOMODEL

Para tornar o AutoModel disponível ao público foi criado um site utilizando o Google Sites (https://sites.google.com/site/uenfautomodel/). Neste site, além do link para download do AutoModel foi disponibilizado o manual de usuário do AutoModel.

4.6 DESENVOLVIMENTO DO NOVO MÓDULO DE ALINHAMENTO COM O HMMER

O desenvolvimento do novo módulo de alinhamento foi feito de acordo com a seção 4.1. Além disto, no desenvolvimento foram utilizados os softwares hmmfetch, hmmsearch e hmmalign do pacote HMMER versão 3.1b1. Apresentados na seção 1.6 desta dissertação.

Além disso, é necessário a conversão entre os formatos FASTA, utilizado pelo HMMER, e PIR, utilizado pelo Modeller. Para esta conversão foi utilizado o módulo bio.SearchIO da biblioteca BioPython (COCK et al., 2009) versão 1.61. O mesmo módulo também foi utilizado para análisar os arquivos gerados pelo HMMER.

Para o funcionamento do HMMER é necessário a utilização de um banco de dados de famílias de proteínas. Para este projeto foi escolhido o banco de dados Pfam-A 27.0 (FINN et al., 2013). O Pfam é um banco de dados que contém uma vasta coleção de famílias de proteínas. Neste banco de dados, cada família é representada por modelos ocultos de Markov (HMM) e que podem ser lidos pelos programas do HMMER.

4.7 DESENVOLVIMENTO DO NOVO MÓDULO DE ALINHAMENTO COM O MUSCLE

O módulo de alinhamento com o MUSCLE foi desenvolvido baseando-se em uma versão modificada do módulo de alinhamento com o HMMER. Nesta nova versão,

o software HMMER foi substituído pelo MUSCLE, e não há mais a necessidade da utilização do banco de dados PFAM-A.

4.8 REALIZAÇÃO DE TESTES COMPARATIVOS ENTRE DIFERENTES VERSÕES DO AUTOMODEL

Estes testes têm como objetivo avaliar se os novos módulos de alinhamento permitem o AutoModel gerar modelos com melhor qualidade, comparando-se com as versões anteriores do AutoModel.

Estes teses consistiram na utilização de quatro versões do AutoModel conforme a tabela abaixo:

Tabela 2 – Ferramenta responsável pelo alinhamento em cada versão do AutoModel

Versão do AutoModel	Ferramenta de alinhamento
AutoModel 0.4	Modeller versão 9.4
AutoModel 0.4	Modeller versão 9.9
AutoModel 0.5	HMMER
AutoModel 0.5M	MUSCLE

Cada versão do AutoModel foi testada com quatro proteínsa diferentes (tabela abaixo). Nestes testes, foram avaliados o tempo necessário para a modelagem, tempo necessário para a realização do alinhamento e a qualidade do modelo gerado. A qualidade destes modelos foi avaliada utilizando os gráficos gerados pelo servidor Prosa-web (*Protein Structure Analysis*) (https://prosa.services.came.sbg.ac.at/prosa.php) .

Tabela 3 – Sequencias utilizadas para a comparação entre versões do AutoModel

Sequencia	Tamanho	Proteína molde	Tamanho	Similaridade (%)
gi|21913852|	458	1uij	416	29
TvLDH	336	1bdmA	216	45
gi|157108525|	359	1iknD	236	40

5 RESULTADOS

5.1 VERSÕES DO AUTOMODEL

Para cada etapa de desenvolvimento do AutoModel neste trabalho foi criado uma versão (tabela 4). A versão original, àquela datada em janeiro de 2012, foi nomeada como AutoModel 0.1, esta versão foi utilizado como base no desenvolvimento das versões posteriores do AutoModel.

Tabela 4 – Recursos adicionados ao AutoModel em cada versão

Versão do AutoModel	Fase	Comentário
AutoModel 0.1	Alpha	Versão Inicial do AutoModel
AutoModel 0.2	Alpha	Adicionado o recurso de refinamento de loops.
AutoModel 0.3	Alpha	Nova versão do sistema de chamadas de procedimentos remot
AutoModel 0.4	Alpha	Adicionado o recurso que permite a modelagem off-line.
AutoModel 0.5	Beta	Adicionado o alinhamento com o HMMER

Todas as versões do AutoModel estão em fase de desenvolvimento alpha, isto é, versões em desenvolvimento e não indicado para o uso cotidiano, portanto não disponível ao público. A versão 0.5, no entanto, está em fase beta que indica que o mesmo está em fase de testes e disponível ao público através do site do AutoModel (https://sites.google.com/site/uenfautomodel/).

5.2 MÓDULO DE REFINAMENTO DE *LOOP*

Com a adição deste módulo, após a modelagem o usuário possui a opção de melhorar a qualidade do modelo gerado através do refinamento das regiões de loops. Este módulo consiste na janela Refinamento de Loops no AutoModel Client e o módulo loop_refinement no AutoModel Server.

5.2.1 Módulo de refinamento de *loop* no AutoModel Client

A janela Refinamento de Loops (figura 6) permite que o usuário selecione a região do *loop* que deseja refinar através dos campos resíduo inicial e resíduo final. Além disto, a janela também possui o botão Ok que quando pressionado envia o modelo para o AutoModel Server e o instrui, via internet, a fazer o processo através do módulo loop_refinement.

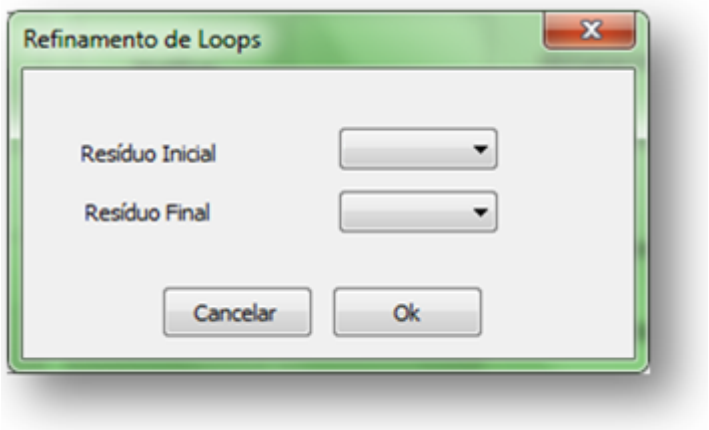


Figura 6 – Janela de refinamento de *loops* do AutoModel 0.5

Esta janela pode ser acessada através do botão Refinar Loops que está em uma seção com o mesmo nome na janela principal do AutoModel Client. Além deste botão, a seção Refinar Loops possui um botão que permite a avaliação do novo modelo gerado.

A janela Refinamento de *loops* foi desenvolvida utilizando a biblioteca WxPython com o nome interno de Refinamento_de_Loop(). Ao abrir esta janela o modelo é carregado pela classe Pdb_File, que através do métodos start_residue() e end_residue() é possível determinar o resídulo inicial e final do modelo gerado. Estes dados são utilizados pela janela Refinamento_de_Loop() para popular os botões resíduo inicial e final. Ao pressionar o botão OK é instanciada a classe LoopModelStep com informações sobre o modelo, instancia da classe Client e região do loop a ser refinado.

A classe LoopModelStep foi desenvolvida em python para gerenciar a tarefa de refinamento de *loops* no AutoModel Client. Ela herda de ModellingStep a interface para a comunicação com o GUI (*Graphic User Interface* - Interface Gráfica de usuário) e comunicação com o AutoModel Server através de uma instancia da classe Client (figura 7). Nesta comunicação é enviado ao AutoModel Server o modelo e a região de *loop* que será remodelado e através de uma *thread* é executada a etapa. Esta *thread* evita o bloqueio da GUI durante a modelagem .

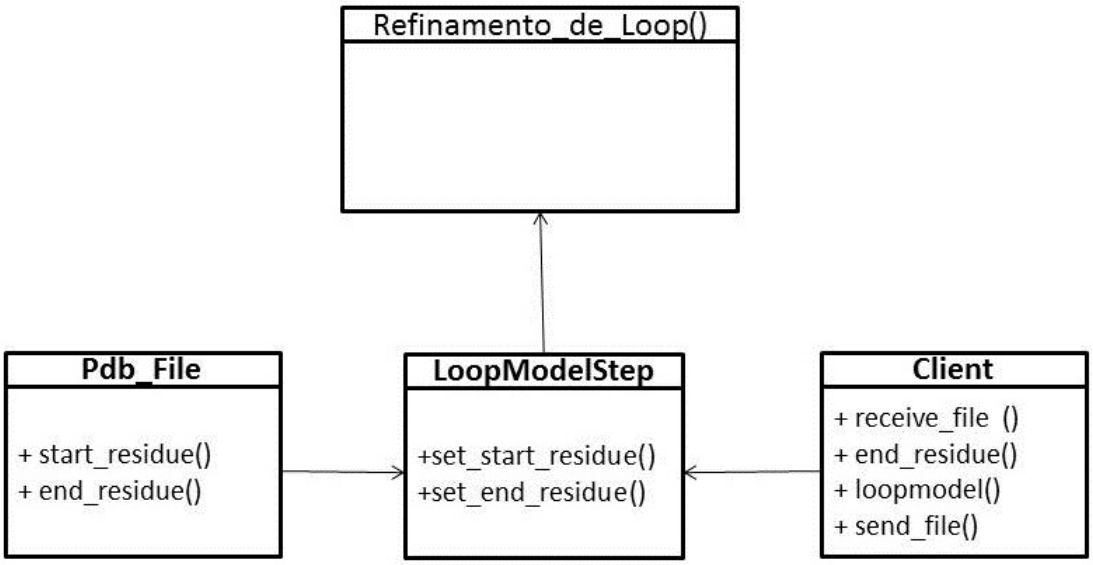


Figura 7 – Diagrama de classes do módulo de refinamento de *loops*

5.2.2 Módulo refinamento de *loop* no AutoModel Server

O script loop.py executa o loop_refinement e com o auxilio do *software* Modeller para gerar o modelo. O AutoModel foi configurado para gerar 5 modelos. Além disso, o módulo loop_refinement também avalia estes novos modelos para que o melhor seja enviado para o AutoModel Client via internet, o que possibilita uma posterior visualização pelo usuário.

O critério de avaliação utilizado pelo módulo é o parâmetro *modeller objective function* que está presente no cabeçalho de cada modelo gerado (figura 8). Esta função é o resultado da minimização da função objetiva que o Modeller utiliza para as restrições. Por si, só este valor não contem nenhuma informação importante, mas é útil na comparação entre modelos.

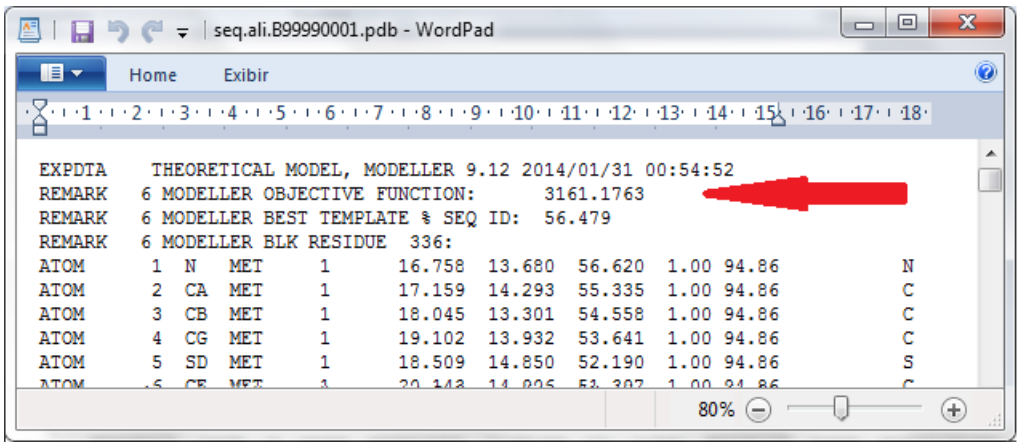


Figura 8 – A seta vermelha aponta ao parâmetro *modeller objective function* de um modelo gerado pelo Modeller.

5.3 MÓDULO DE VALIDAÇÃO DO REFINAMENTO DE *LOOP*

Como o refinamento de *loop* gera um novo modelo é necessário verificar se a qualidade deste modelo está realmente melhor que o original. Embora o AutoModel permita analisar o modelo através dos relatórios gerados pelo *software* PROCHECK (acessíveis pelo botão avaliar do AutoModel), o PROCHECK não permite a comparação dos modelos. Logo, foi desenvolvido o módulo evaluate_loop_model que gera um gráfico utilizando como parâmetro a pontuação DOPE de cada resíduo do *template*, modelo e modelo com o *loop* refinado. Este módulo será executado automaticamente após o refinamento de *loop* e será exibido o gráfico automaticamente no computador do usuário.

O módulo evaluate_loop_model foi desenvolvido para o AutoModel Server realize a tarefa em 2 etapas: na primeira etapa o módulo utiliza a função assess_dope do Modeller para gerar um arquivo com a extensão .profile contendo o perfil de energia DOPE de cada resíduo para o *template*, modelo e o modelo com o loop refinado.

Na segunda etapa, utilizando os arquivos gerados da etapa anterior e a biblioteca pylab o módulo evaluate_loop_model gera o gráfico que ficará disponível para download pelo o AutoModel Client. Este download é realizado no final do refinamento de loop.

5.4 ACESSO EXTERNO AO AUTOMODEL SERVER

O AutoModel foi projetado para funcionar em uma estrutura cliente servidor via internet. O AutoModel Server, *software* responsável por realizar o processamento da modelagem, recebe requisições de uma ou mais instâncias do AutoModel Client. Estas instâncias geralmente estão sendo executadas em um computador diferente do qual está sendo executado o AutoModel Server, assim, para haver a interoperabilidade entre os 2 *softwares* é necessário a comunicação de ambos via internet, utilizando a porta 18661.

Para que haja o acesso das instâncias do AutoModel Client a este servidor, foi necessário que a Assessoria de Rede/UENF concedesse o acesso externo através de um de seus *gateways*, dispositivo que interliga uma rede interna à internet. Este gateway possui o IP externo (de internet) 200.20.228.47 e foi configurado pela Assessoria de Rede/UENF para que todos os pacotes que chegarem a porta 18661 sejam redirecionados para o AutoModel Server (figura 9).

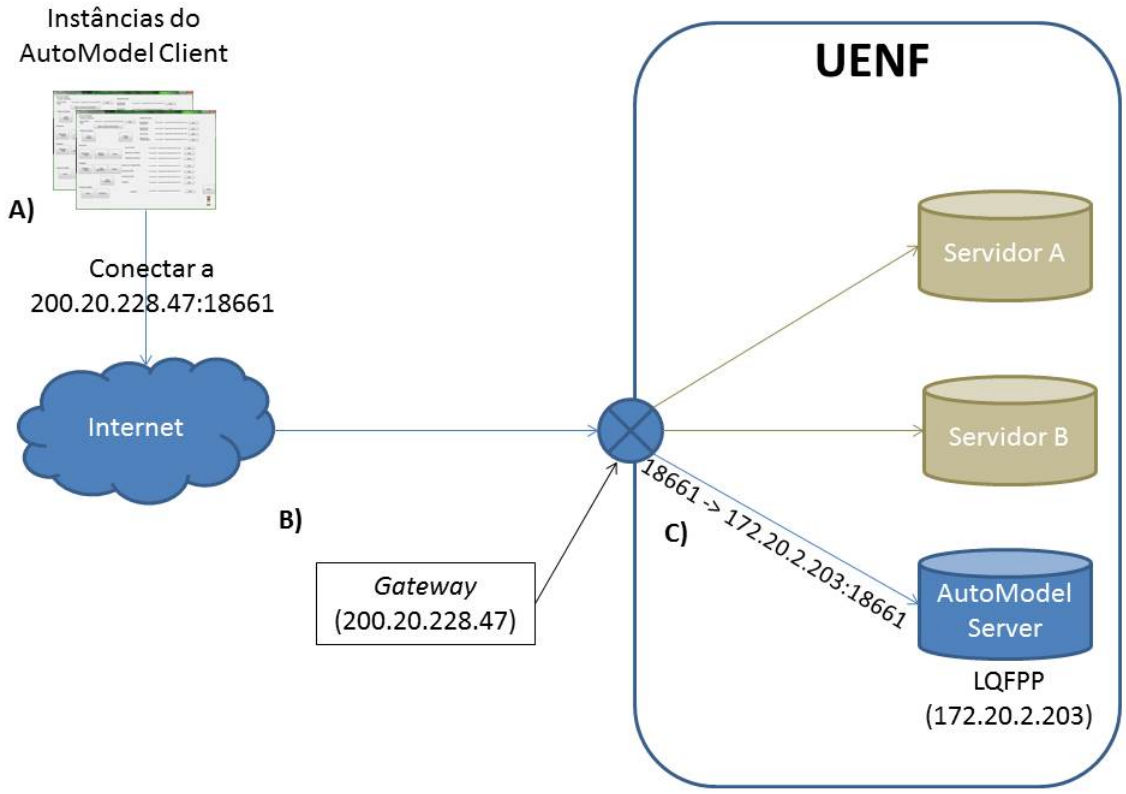


Figura 9 – Esquema de funcionamento da comunicação do AutoModel 0.5: (A) Uma ou mais instâncias do AutoModel faz uma requisição ao AutoModel Server via internet (200.20.228.47:18661); (B) *Gateway* da UENF recebe a requisição e a redireciona para o servidor do AutoModel Server (C).

5.5 NOVA VERSÃO DO SISTEMA DE CHAMADAS DE PROCEDIMENTOS REMOTOS

O uso do AutoModel 0.1 na internet foi seriamente prejudicado devido a problemas de comunicação externa inerentes a rede interna e a internet. Assim, esta versão comumente congelava durante a modelagem, o que levava a perda da experimentação *in silico*. Desta forma, para sanar este problema uma nova abordagem de comunicação entre o AutoModel Server e Client foi implementada.

Na primeira versão do AutoModel, todos os aspectos da modelagem eram gerenciados pelo AutoModel Server, assim, quando o AutoModel Client precisava de uma informação, como por exemplo, o local de armazenamento dos arquivos de alinhamento, era necessário que o AutoModel Client fizesse uma requisição ao AutoModel Server (figura 10).

Além disso, a conexão entre o AutoModel Server e AutoModel Client era mantida durante toda a modelagem na primeira versão do AutoModel, caracterizando assim uma conexão síncrona. Caso esta conexão fosse perdida por algum motivo

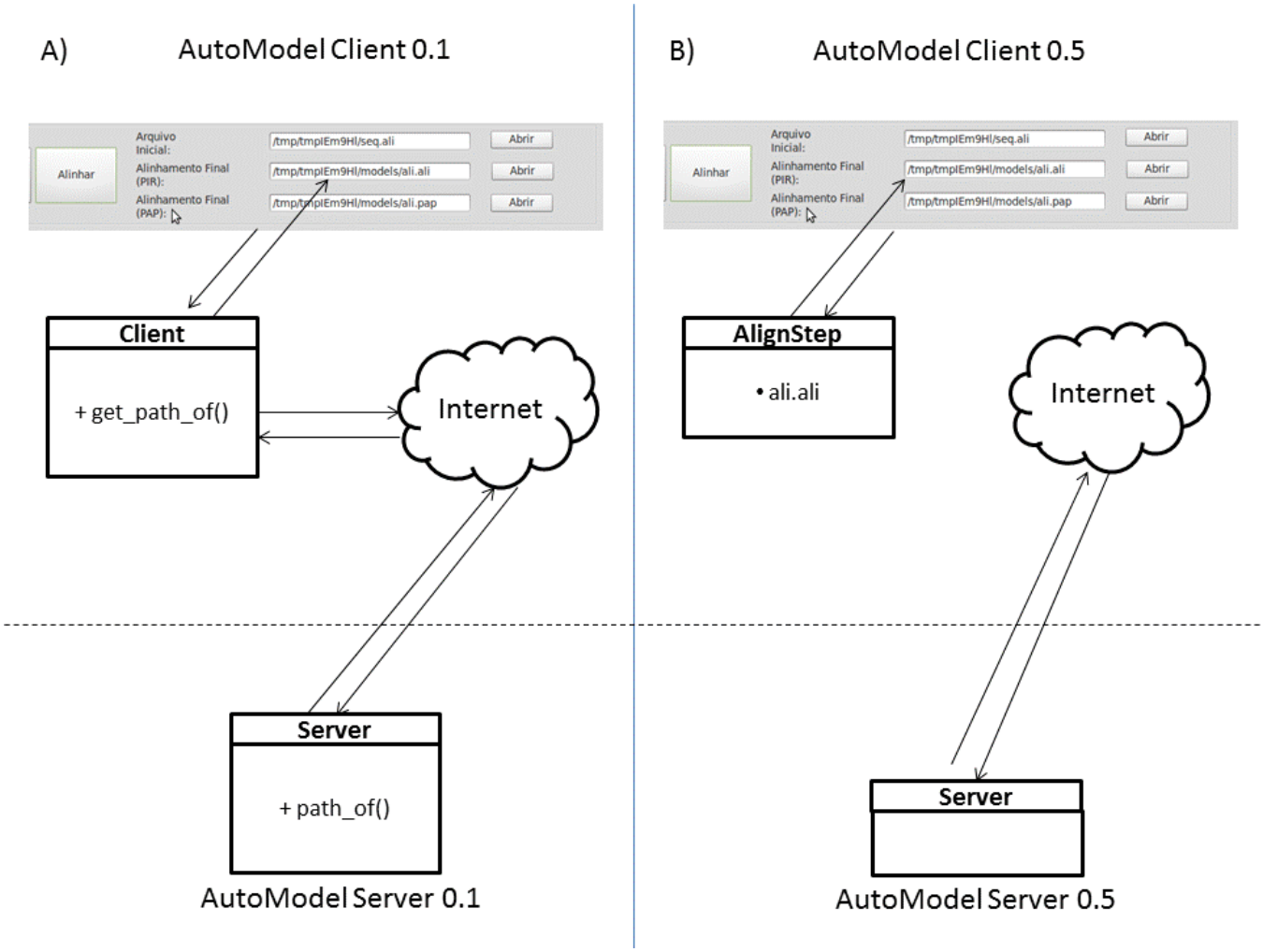


Figura 10 – Exemplo comparativo do gerenciamento de dados do AutoModel: (A) No AutoModel 0.1 a informação sobre o caminho de arquivos fica armazenado no AutoModel Server. (B) Na versão atual do AutoModel, todos os dados da modelagem são gerenciados pelo pacote ModelingStep no AutoModel Client.

externo a modelagem não poderia continuar. Neste contexto, devido a infraestrutura que o AutoModel está sujeito, a comunicação via internet entre o cliente e o servidor do AutoModel era muito instável.

No AutoModel 0.3 a arquitetura de rede continua baseada na cliente-servidor, contudo a principal mudança está na forma de gerenciamento das informações. Nesta versão, o controle de toda a modelagem é feito através do AutoModel Client, realizando assim, conexões pontuais ao AutoModel Server. O objetivo foi manter o controle no AutoModel Client e o processamento no AutoModel Server. Assim quando o AutoModel necessita realizar uma etapa que necessita de processamento, os arquivos necessários para esta etapa de modelagem são enviados para o AutoModel Server e este faz o processamento. Após o envio dos dados os 2 softwares se desconectam e voltam a reconectar novamente após o fim do processamento da etapa. Ao fim da etapa os

arquivos gerados são enviado de volta ao AutoModel Client.

Internamente, o AutoModel continua utilizando a biblioteca RPyC, no entanto, sua forma de trabalho foi modificada. A classe Client que no AutoModel 0.1 era responsável por várias tarefas da modelagem, na versão 0.3 ela foi otimizada, ficando responsável apenas pela comunicação entre o AutoModel Client e AutoModel Server e tarefas de rede.

Além do novo sistema de comunicação foi criado o módulo ModellingStep (figura 11), que solicita a execução de cada etapa da modelagem por homologia ao AutoModel Server, utilizando a classe cliente. O módulo ModellingStep fornece uma interface para o ambiente gráfico, e permitindo o controle de cada etapa da modelagem. Na figura 12 é apresentado o diagrama de classes deste novo módulo.

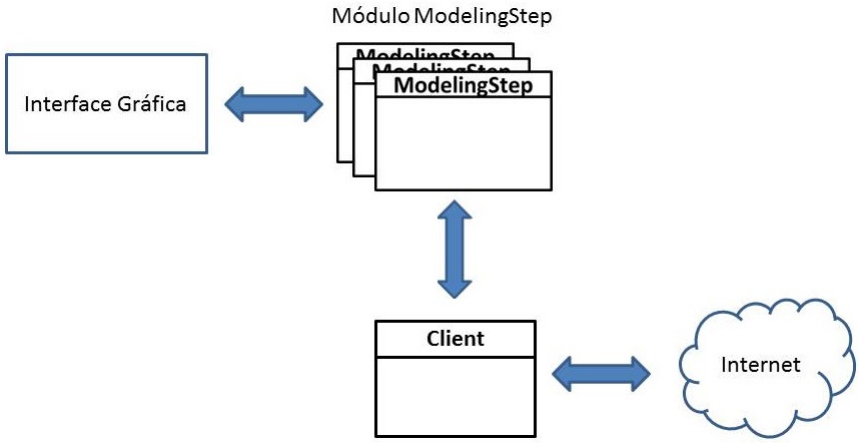


Figura 11 – Esquema de funcionamento do módulo ModellingStep

No AutoModel Server, as mudanças ocorreram somente no módulo server. Este módulo no AutoModel 0.1 era responsável por manter a conexão com as instâncias do AutoModel Client e gerenciar o processo de modelagem. A partir do AutoModel 0.3, o servidor é responsável somente pelo processamento da etapa e por receber os arquivos necessários para a execução desta tarefa.

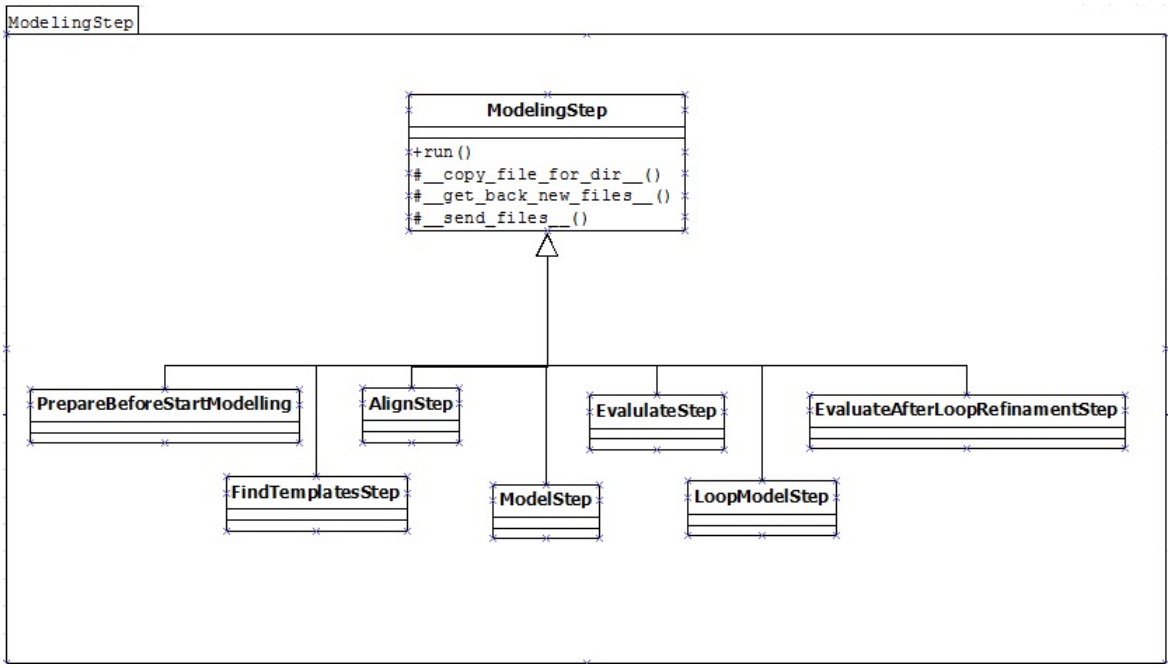


Figura 12 – Diagrama de classes do módulo ModellingStep

Como consequência da nova abordagem utilizada no AutoModel 0.3, ocorreu uma queda do volume de dados transferidos em cada etapa da modelagem (figura 13). Isto se deve a menor necessidade de transferência de arquivos para cada etapa, já que os únicos arquivos que vão para o servidor na nova versão do AutoModel são exatamente os necessários para cada etapa. Esta queda de volume de dados transferidos traz 2 características benéficas para o AutoModel, a primeira é que a modelagem dura menos tempo e a segunda é que não sobrecarrega o trafego dentro da rede da UENF

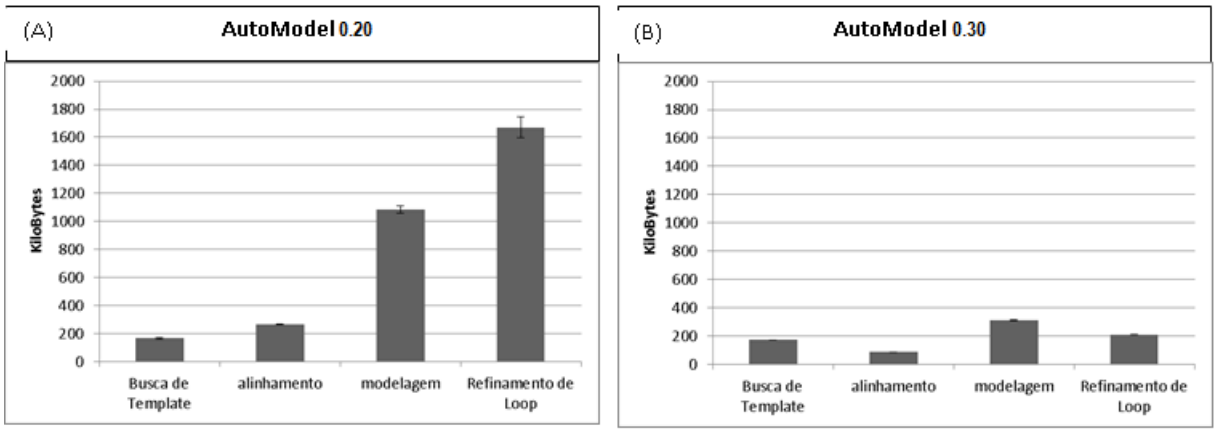


Figura 13 – Volume de dados que foi transferido na modelagem de uma proteína de 339 resíduos utilizando as 2 versões do AutoModel com o sistema de comunicação aperfeiçoado.

5.6 REGISTRO DO AUTOMODEL COMO UMA FERRAMENTA DE USO BIOTECNOLÓGICO

Antes de tornar o AutoModel disponível ao público, o AutoModel foi registrado ao INPI (Instituto Nacional da Propriedade Industrial) como uma ferramenta de uso biotecnológico. O INPI é o órgão responsável pelo registro de *softwares* e patentes. No caso de *softwares*, a lei brasileira entende que o código é uma obra de origem literária, não cabendo assim patentes. No entanto, a lei nº 9.609/98, conhecida como a lei do *software*, inclui o direito a paternidade e de modificação, direito este que, poderá ser exercido a qualquer momento. Esta lei também destaca que a propriedade de um *software* é exclusivamente do empregador, contratante de serviços ou órgãos públicos, o qual o desenvolvedor está vinculado. Além disso, o *software* poderá ser registrado junto a órgãos como o INPI a fim de proteger juridicamente a sua autoria (FILHO, 2003).

Para o registro do AutoModel, foi entregue a Assessoria de Patentes da UENF a documentação formal e técnica necessária para efetuá-la. A documentação formal possui informações do titular e a documentação técnica é composta pelos diagramas de classe do AutoModel Client e Server 0.4 e trechos do *software* que são responsáveis pela gestão de janelas do AutoModel Client e comunicação entre o AutoModel Server e AutoModel Client. Isto é, partes do AutoModel que comprovam sua originalidade.

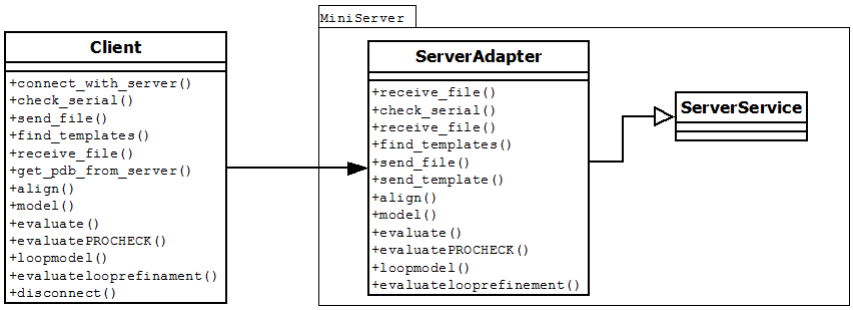


Figura 14 – Diagrama de Classes do módulo MiniServer e classe Client modificada para utilização do AutoModel off-line

5.7 MODO OFF-LINE

O modo off-line do AutoModel consiste em uma versão modificada do AutoModel Server implementada como um módulo no AutoModel Client 0.4. Este módulo com o nome de MiniServer não utiliza a biblioteca RPyC para a comunicação e sim uma versão modificada da classe Client do AutoModel Client e um adaptador para a classe ServerService. Este adaptador é necessário, pois, a classe ServerService fornece a interface de comunicação com o AutoModel Client. Como esta comunicação utiliza a biblioteca RPyC, é necessário que os métodos da classe ServerService comecem com a palavra exposed. Para não ter que reescrever todos métodos da classe Client modificada foi criada então a classe Adaptadora ServerAdapter. Foi implementada também a janela *Select AutoModel Mode*, que permite a escolha de qual modo o AutoModel vai iniciar. Dependendo da escolha do usuário, este módulo carrega a classe Client original que se conecta ao AutoModel Server ou a classe Client modificada que utiliza o módulo MiniServer (figura 14).

5.8 ALINHAMENTO COM O HMMER

Visando uma melhor qualidade dos modelos, o módulo de alinhamento do AutoModel 0.4 foi substituído. O módulo original utilizava o Modeller como software de alinhamento. O novo módulo, implementado no AutoModel 0.5, utiliza o pacote de softwares HMMER. O novo módulo Align também utiliza o banco de dados do Pfam para a busca de famílias de proteínas. Assim como o antigo módulo de alinhamento, o novo módulo recebe a sequência desalinhada da proteína alvo e proteína molde no formato PIR (arquivo seq.ali) e a estrutura da proteína molde (figura 15).

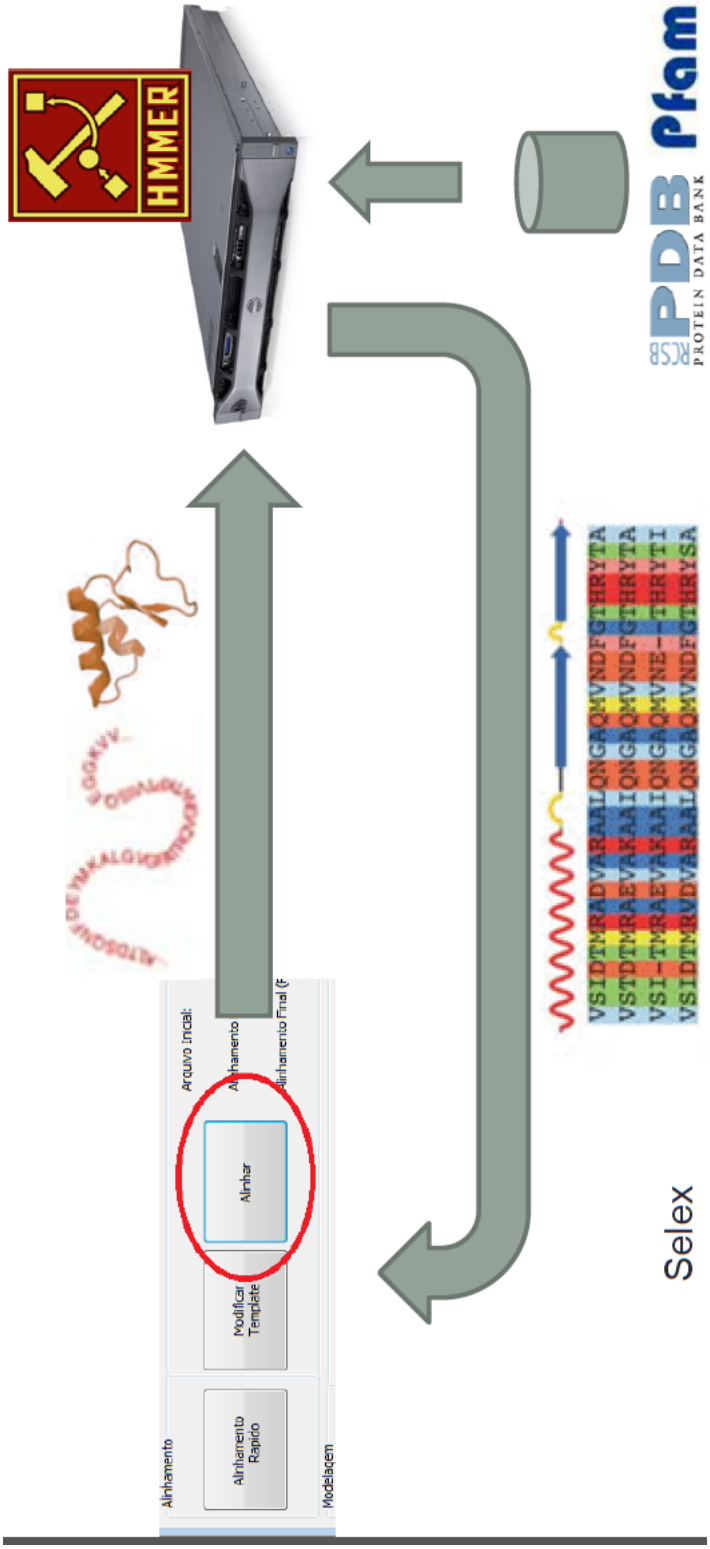


Figura 15 – Esquema do funcionamento do módulo de alinhamento com o HMMER: Ao pressionar o botão Alinhar, o AutoModel Client envia ao AutoModel Server as sequências da proteína-molde e proteína-alvo, estes serão alinhados pelo AutoModel Server utilizando os bancos de dados PDB e PFAM-A, após o processamento o AutoModel Server envia para o AutoModel Client o alinhamento das duas sequências.

Após o recebimento destes arquivos, o AutoModel Server converte o seq.ali no formato FASTA utilizando o método convert_seqali_pir_to_fasta_format da classe Align. Este método foi desenvolvido de forma a criar um script Python que utiliza o comando *alignment* do modeller para fazer a conversão. Este script é executado com o modeller e sua saída é um arquivo seq.fasta. Isto é necessário, pois o HMMER não lê arquivos no formato PIR.

No arquivo seq.fasta o AutoModel Server realiza então uma busca de perfis HMM, utilizando o banco de dados do PFAM. Para isto, o servidor do AutoModel utiliza o método search_an_hmm. Este método então executa o comando hmmsearch do HMMER e armazena o resultado no arquivo hmmsearch.txt.

Utilizando o arquivo hmmsearch.txt, é executado o método find_better_motif da classe Align. Este método, utiliza a biblioteca biopython para extrair os nomes dos perfis HMM que foram encontrados no arquivo seq.fasta. Com esta informação são extraídos os perfis HMM escolhidos do banco de dados PFAM e armazenado no arquivo PRF.hmm. Para isto, o AutoModel Server utiliza o método fetch_hmm que utiliza o software hmmfetch do HMMER.

O processo de alinhamento termina com o comando align_with_hmmer. Este comando utiliza os seguintes arquivos seq.fasta e PRF.hmm que foram criados previamente. Internamente, o comando align_with_hmmer executa o software hmalign com estes 2 arquivos. O resultado então é enviado para o AutoModel Client.

5.9 MÓDULO DE ALINHAMENTO COM O MUSCLE

Baseado no módulo de alinhamento utilizando o HMMER, o módulo de alinhamento do AutoModel 0.5M utiliza o software MUSCLE ao invés do HMMER. Internamente o funcionamento destes dois módulos é semelhante.

Na classe Align, o método align_with_hmmer, foi substituído por align_with_muscle. Este método quando acionado executa o muscle com o parâmetro –in arquivo com sequencias não alinhadas, –out seq.ali, onde seq.ali é o arquivo que será gerado após o alinhamento.

Com a utilização do Muscle não foi mais necessário a utilização da bibliotecas de fragmentos PFAM-A, diminuindo assim o espaço de armazenamento necessário para manter o AutoModel Server. Além disso, foi notada uma melhora significativa na velocidade de alinhamento quando se compara as versões 0.4 e 0.5 do AutoModel.

5.10 ANÁLISE DA QUALIDADE DOS MODELOS OBTIDOS DE DIFERENTES VERSÕES DO AUTOMODEL

Com o objetivo de melhorar a qualidade do modelo gerado, o Modeller, software de alinhamento utilizado pelo AutoModel 0.4, foi substituído pelo HMMER no AutoModel 0.5 e Muscle no AutoModel 0.5M. Neste contexto, para avaliar se este objetivo foi alcaçado foram realizados testes nos quais foram verificados a qualidade do modelo gerado e o tempo necessário de alinhamento na modelagem de diferentes proteínas. Estes dados foram então tabulados em uma planilha do Microsoft Excel.

Nestes testes mostraram que o AutoModel 0.5M que utiliza a ferramenta Muscle para fazer o alinhamento construiu modelos com qualidade igual ou superior ao AutoModel 0.4 que utiliza o Modeller. Além disso, com a versão 0.5M a modelagem se tornou muito mais rápida e também com um menor custo computacional. Em contrapartida o AutoMode 0.5 que utiliza o HMMER como ferramenta de alinhamento gerou modelos com qualidade inferior ao AutoModel 0.4 e AutoModel 0.5.

Tabela 5 – Modelagem da gi|21913852| com o *template* 1uij

Versão do AutoModel	0.4 (9v9)	0.4 (9v4)	0.5 (HMMER)	0.5M (Muscle)
Qual. do modelo (*Z-score*) (fig. 16)	-4.65	-4.85	-1.21	-4.41
Tempo de alinhamento	2min7seg	2min18seg	1min27seg	1seg
Tempo da modelagem completa	8min10seg	7min44seg	7min	6min14seg

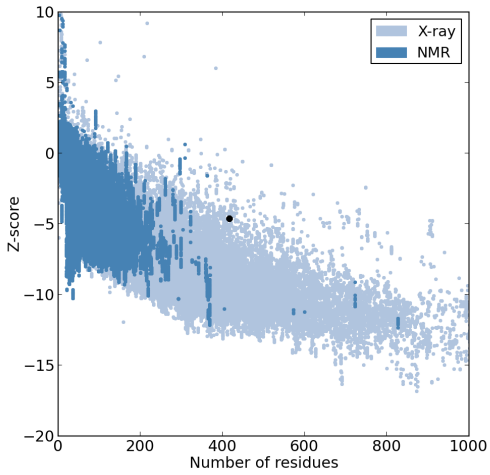
Tabela 6 – Modelagem da TvLDH com o *template* 1bdmA

Versão do AutoModel	0.4 (9v9)	0.4 (9v4)	0.5 (HMMER)	0.5M (Muscle)
Qual. do modelo (*Z-score*) (fig. 17)	-8.9	-8.43	-5.23	-8.18
Tempo de alinhamento	1min05seg	1min21seg	1min40seg	2seg
Tempo da modelagem completa	6min50seg	5min41seg	7min20seg	5min28seg

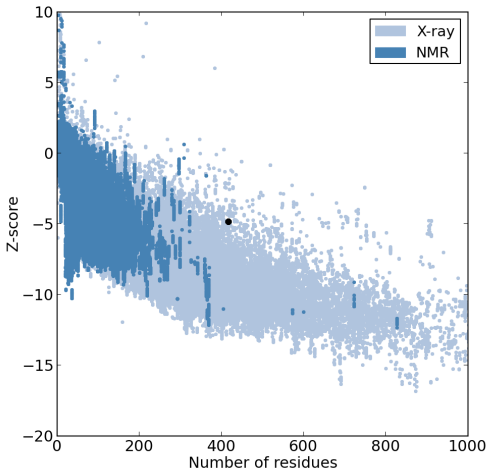
Tabela 7 – Modelagem da gi|157108525| com o *template* 1iknD

Versão do AutoModel	0.4 (9v9)	0.4 (9v4)	0.5 (HMMER)	0.5M (Muscle)
Qual. do modelo (*Z-score*) (fig. 18)	-1.4	5.26	-0.76	-2.66
Tempo de alinhamento	52seg	32seg	2min10seg	1seg
Tempo da modelagem completa	5min	4min22seg	6min	3min48seg

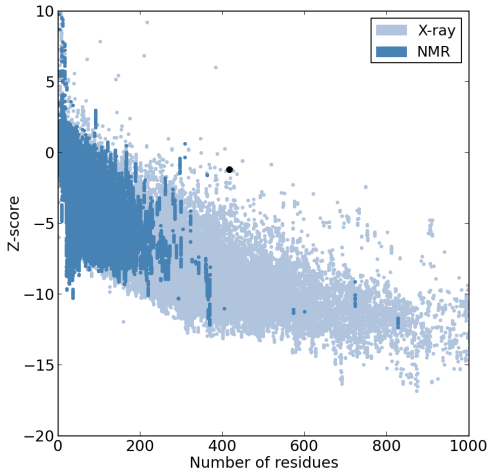
Gráficos *Z-Score* do modelo da sequência gi|21913852|



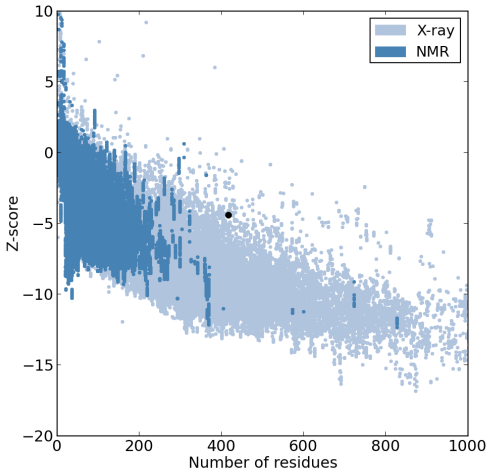
(a) AutoModel 0.4 (9v9)



(b) AutoModel 0.4 (9v4)



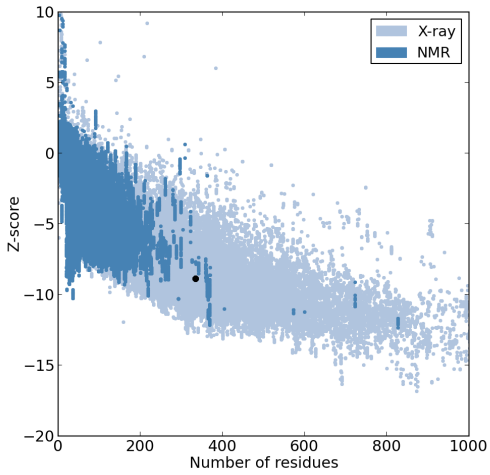
(c) AutoModel 0.5 (HMMER)



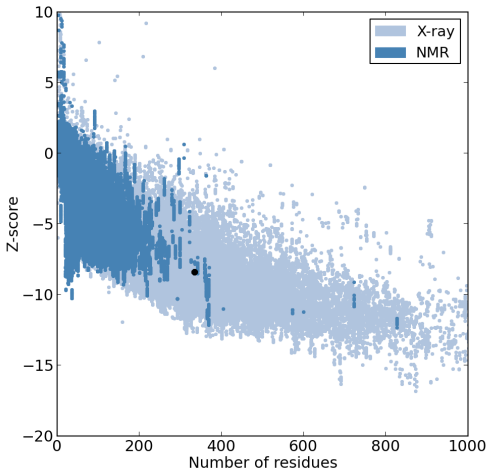
(d) AutoModel 0.5M (Muscle)

Figura 16 – Gráficos *Z-Score* do modelo da sequência gi|21913852| gerado por cada versão do AutoModel.

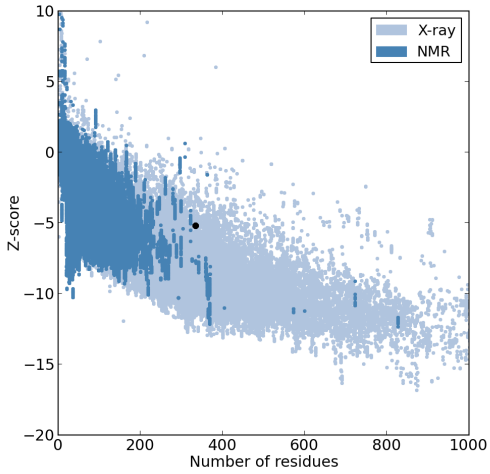
Gráficos *Z-Score* do modelo da sequência TvLDH



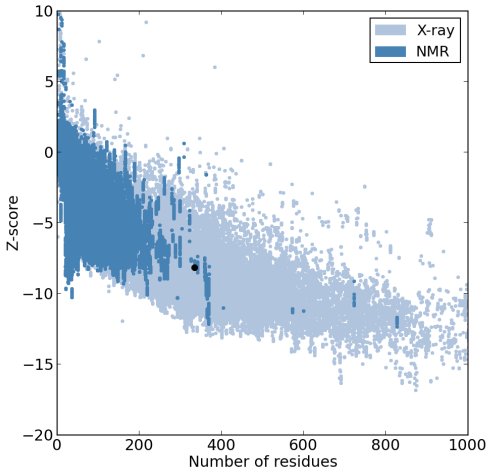
(a) AutoModel 0.4 (9v9)



(b) AutoModel 0.4 (9v4)



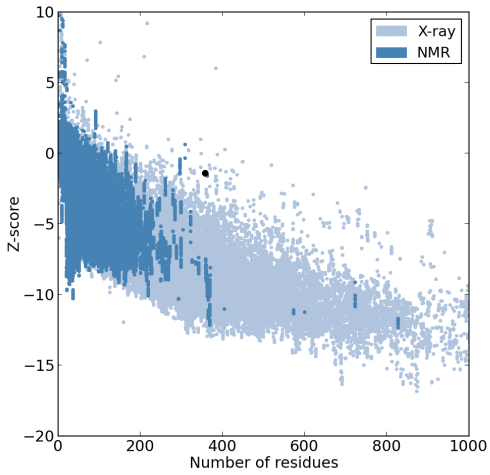
(c) AutoModel 0.5 (HMMER)



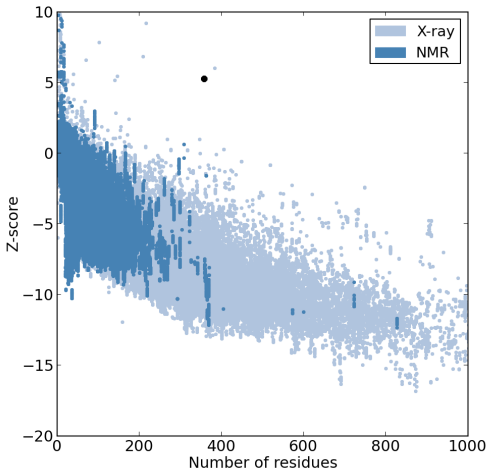
(d) AutoModel 0.5M (Muscle)

Figura 17 – Gráficos *Z-Score* do modelo da sequência TvLDH gerado por cada versão do AutoModel.

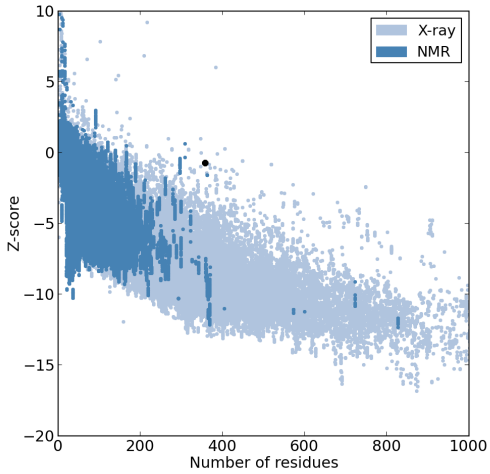
Gráficos *Z-Score* do modelo da sequência gi|157108525|



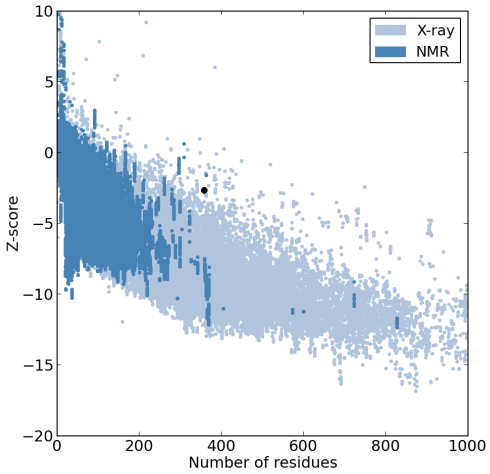
(a) AutoModel 0.4 (9v9)



(b) AutoModel 0.4 (9v4)



(c) AutoModel 0.5 (HMMER)



(d) AutoModel 0.5M (Muscle)

Figura 18 – Gráficos *Z-Score* do modelo da sequência gi|157108525| gerado por cada versão do AutoModel.

6 DISCUSSÃO

O AutoModel foi desenvolvido de forma a possuir uma interface gráfica de fácil uso e que guiasse o usuário na experimentação de modelagem de proteínas. Diferente da maioria das ferramentas de modelagem, o AutoModel possui um maior controle dos parâmetros de todas as suas etapas. A idéia é que com um maior controle da modelagem, os modelos *in silico* gerados pelo AutoModel sejam mais refinados do que nos sistemas automatizados e que incluam heteroatomos.

O AutoModel versão 0.1 foi a primeira versão deste *software*, ela possuía alguns recursos como seleção e alteração de heteroátomos para a geração do modelo e a geração do relatório estereoquímico com a utilização do *software* Procheck. No entanto, esta versão só chegou a ser utilizada dentro do laboratório, não sendo disponibilizada ao público. Portanto, era interessante oferecer esta ferramenta pela internet e oferecer novos parâmetros para a modelagem como o refinamento de *loops*.

O refinamento de *loops* é interessante, pois, o motor do AutoModel que é o *software* Modeller cria modelos utilizando a modelagem por homologia. Esta técnica utiliza um alinhamento entre a proteína-molde e a proteína-alvo, onde são localizadas as regiões conservadas entre as duas sequências e as não conservadas (KRIEGER; NABUURS; VRIEND, 2003). Normalmente os *loops* estão localizados nestas regiões não conservadas. Diferentemente das regiões conservadas, as regiões não conservadas não possuem informação estrutural disponível, levando o Modeller a utilizar algumas estratégias para resolver a estrutura destes locais (FISER; DO; ŠALI, 2000). No entanto, nem sempre o Modeller consegue utilizar a melhor estratégia para resolver esta parte da estrutura, requerendo assim uma intervenção do usuário. Deste modo, para predizer estruturas de proteínas com maior precisão o recurso de refinamento de *loops* se torna uma ferramenta essencial para o cumprimento do objetivo do AutoModel.

Além disto, no AutoModel foi resolvido um dos pontos fracos do Modeller que é o seu módulo de alinhamento de sequências. O módulo utilizado pelo Modeller é baseado em uma variação do algoritmo Smith-Waterman (SMITH; WATERMAN, 1981). No Modeller, este algoritmo acaba gerando vários erros durante o alinhamento que afeta negativamente a qualidade do modelo predito pelo Modeller. Para resolver este problema, o *software* desta etapa foi substituído pelo HMMER, que utiiza modelos probabilísticos conhecido como Cadeias Ocultas de Markov sendo muito mais sensível do que o algoritmo de Smith-Waterman. No entanto, os modelos gerados pelas versões do AutoModel que utilizava o HMMER (AutoModel 0.5) tiveram uma sensível queda sem sua qualidade quando comparado-se com a versão que utiliza o Modeller. Esta queda de qualidade pode está relacionada a qualidade de sequências que são alinhadas, o

HMMER foi desenvolvido para trabalhar com o alinhamento de multiplas sequências, enquanto o AutoModel só alinha duas sequências. Desta forma, na versão 0.5M o HMMER foi substituído pelo MUSCLE que obteve melhores resultados do que o HMMER e o Modeller, além disso, os resultados mostraram que o MUSCLE diminuiu bastante o custo computacional necessário para a modelagem, o que é interessante em um sistema que funciona em um servidor central como o AutoModel.

No desenvolvimento deste trabalho o AutoModel foi dividido em 4 versões que foram baseadas no AutoModel 0.1. Em cada versão foi adicionado um recurso ou resolvido um problema. Como foi o caso do novo módulo de chamadas de procedimento remoto, que com a nova versão se tornou mais eficiente e mais rápido, tornando o processo de modelagem mais estável. Além disto, a última versão do AutoModel está totalmente funcional e disponível ao público através do site (https://sites.google.com/site/uenfautomodel/

Como perspectiva de desenvolvimento do AutoModel estão:

i. Desenvolvimento da modelagem de proteínas com resíduos que sofreram modificações pós-traducionais;

ii. Aumentar o número de usuários, especialmente na américa do sul, através do curso CBAB/CABBIO 2015:tópicos em Biologia Computacional que será realizado em julho de 2015 na Universidade Tecnológica Federal do Paraná (UTFPR);

iii. Desenvolvimento de modelagem por *structural profile*;

APÊNDICE A – ESTUDO DE CASO DO AUTOMODEL

As lactato desidrogenases (LDH) são enzimas que estão presentes em várias vias metabólicas e estão envolvidas na manutenção do equilíbrio entre o catabolismo e anabolismo de carboidratos. A LDH catalisa a reação de transformação do piruvato em lactato. Este lactato pode ser usado para produzir NADH e ATP em tecidos aeróbicos com a utilização de ácido cítrico. Além disso, a lactato desidrogenase pode catalisar a reação de conversão de lactato em glicogênio (GARVIE, 1980).

O *Trichomonas vaginalis* é um parasita humano que é o causador da tricominíase. No genoma dele foi encontrado o gene de uma proteína LDH (TvLDH) que não possui estrutura tridimensional conhecida. Estudos de bioinformática revelaram que esta LDH possui uma maior similaridade com a malato desidrogenase (MDH) de algumas espécies de organismos do que com outras LDH caracterizando uma possível evolução convergente (WU et al., 1999).

Com intuito de elucidar a estrutura tridimensional da TvLDH realizamos uma modelagem por homologia da TvLDH utilizando como proteína-molde uma malato-desidrogenase pertencente ao organismo Thermus thermophilus.

A.1 MATERIAIS E MÉTODOS

Para a modelagem da TvLDH foi necessário utilizar a abordagem de modelagem por homologia. Esta abordagem permite a criação de modelos teóricos tridimensionais de proteínas com a utilização de “proteínas-moldes”. Estas “proteínas-moldes” são proteínas que possuem sua estrutura tridimensional conhecida e normalmente estão em bancos de dados como, por exemplo, o PDB (Proteín Data Bank). Para a realização da modelagem por homologia da TvLDH foi utilizado o software AutoModel versão 0.3. Além disto, foi utilizado o PROCHECK para avaliar a qualidade estereoquímica do modelo gerado e se necessário realizar ajustes do modelo.

A sequência da TvLDH foi carregada no AutoModel (figura 19) e com ela o software realizou uma busca em seu banco de dados e encontrou 10 proteínas que possuía similaridade com a TvLDH. Destas, a proteína-molde que possui uma maior cobertura foi a 1bdm. 1bdm é a estrutura tridimensional de uma malato-desidrogenase pertencente do organismo Thermus thermophilus (tMDH) (KELLY et al., 1993).

Na sequência da TvLDH presente no alinhamento (Figura 20) com a 1bdmA foram importados o ligante (NAD) e as moléculas de água para que ficasse mais claro

```
>P1;TvLDH
sequence:TvLDH:::::::0.00: 0.00
MSEAAHVLITGAAGQIGYILSHWIASGELYGDRQVYLHLLDIPPAMNRLTALTMELEDCAFPHLAGFVATTDPKA
AFKDIDCAFLVASMPLKPGQVRADLISSNSVIFKNTGEYLSKWAKPSVKVLVIGNPDNTNCEIAMLHAKNLKPEN
FSSLSMLDQNRAYYEVASKLGVDVKDVHDIIVWGNHGESMVADLTQATFTKEGKTQKVVDVLDHDYVFDTFFKKI
GHRAWDILEHRGFTSAASPTKAAIQHMKAWLFGTAPGEVLSMGIPVPEGNPYGIKPGVVFSFPCNVDKEGKIHVV
EGFKVNDWLREKLDFTEKDLFHEKEIALNHLAQGG*
```

Figura 19 – Sequência de resíduos em formato FASTA da TvLDH Segundo o AutoModel a cadeia A da proteína que possuí o nome-código, no PDB, 1bdm é a que possui maior similaridade com a TvLDH (45%), portanto a cadeia desta proteína foi utilizada como o template (proteína-molde) da modelagem.

em qual região da proteína fica o sítio ativo.

Utilizando este alinhamento o AutoModel foi instruído a criar o modelo tridimensional da TvLDH. O AutoModel criou 5 modelos dos quais o software escolheu aquele que possuiam menor energia (figura 21).

```
>P1;1BDM
structureX:1BDM.pdb:   0 :A:+409 :A:MOL_ID  1; MOLECULE  MALATE DEHYDROGENASE;
CHAIN  A; EC  1.1.1.37; ENGINEERED  YES:MOL_ID  1; ORGANISM_SCIENTIFIC
THERMUS THERMOPHILUS; ORGANISM_TAXID  274; EXPRESSION_SYSTEM_VECTOR_TYPE
BACTERIUM: 1.80: 0.17
MKAPVRVAVTGAAGQIGYSLLFRIAAGEMLGKDQPVILQLLEIPQAMKALEGVVMELEDCAFPLLAGLEATDDPD
VAFKDADYALLVGAAPRL--QV-------NGKIFTEQGRALAEVAKKDVKVLVVGNPANTNALIAYKNAPGLNPR
NFTAMTRLDHNRAKAQLAKKTGTGVDRIRRMTVWGNHSSIMFPDLFHAEV--DGR--PALELVDMEWYEKVFIPT
VAQRGAAIIQARGASSAASAANAAIEHIRDWALGTPEGDWVSMAVPS-QGE-YGIPEGIVYSFPVTAK-DGAYRV
VEGLEINEFARKRMEITAQELLDEME-QVKALGLI-.wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww*

>P1;seq.ali
sequence:seq.ali:     : :     : :::-1.00:-1.00
MSEAAHVLITGAAGQIGYILSHWIASGELYG-DRQVYLHLLDIPPAMNRLTALTMELEDCAFPHLAGFVATTDPK
AAFKDIDCAFLVASMPLKPGQVRADLISSNSVIFKNTGEYLSKWAKPSVKVLVIGNPDNTNCEIAMLHAKNLKPE
NFSSLSMLDQNRAYYEVASKLGVDVKDVHDIIVWGNHGESMVADLTQATFTKEGKTQKVVDVLDHDYVFDTFFKK
IGHRAWDILEHRGFTSAASPTKAAIQHMKAWLFGTAPGEVLSMGIPVPEGNPYGIKPGVVFSFPCNVDKEGKIHV
VEGFKVNDWLREKLDFTEKDLFHEKEIALNHLAQGG.wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww*
```

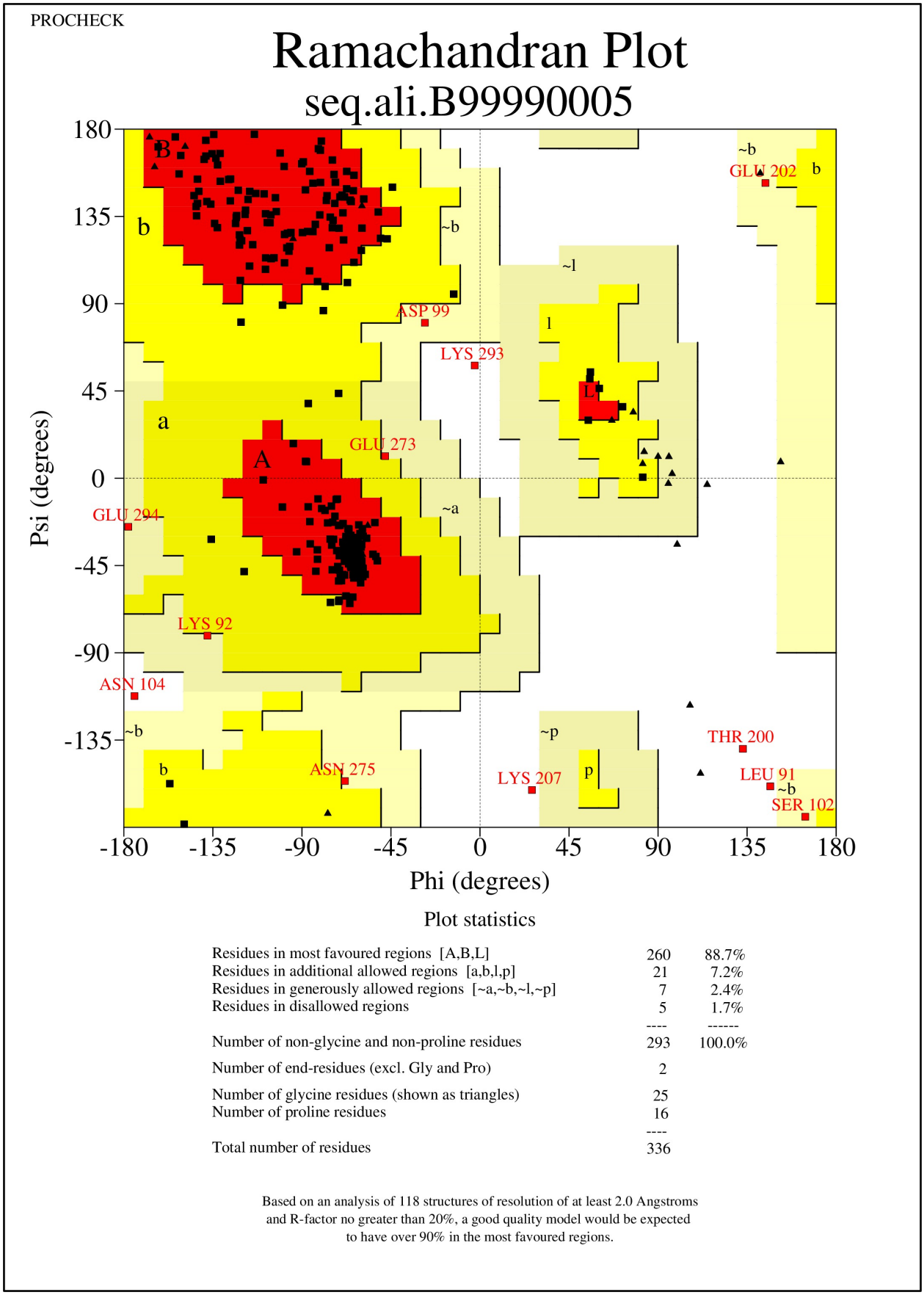
Figura 20 – Alinhamento entre a cadeia A da 1bdm com a TvLDH que está com o nome seq.ali. Note que foi importado o NAD (representado por um ponto) e as moléculas de água (representada por w).

Após a criação do modelo foi feita uma análise estereoquímica deste utilizando o software PROCHECK que é incluso no próprio AutoModel (Figuras 22, 23, 24). Nesta análise, observando o gráfico Ramachandran foi constatado que o loop que continha os resíduos ARG97, ALA98, ILE101, SER102, SER103 precisava ser remodelado. Para isto foi utilizado a opção “refinar loops” do AutoModel, Nesta opção foi ordenado que o software remodelasse o loop do resíduo 93 até o 103. O AutoModel gerou assim um novo modelo e dele foi utilizado o software PROCHECK novamente para gerar uma

Figura 21 – Modelo gerado pelo AutoModel para TvLDH com o ligante NAD

nova análise estereoquímica (Figuras 25, 26, 27). Desta nova análise percebe-se que houve uma melhora na região que continham aqueles resíduos (Figura 28).

Modelo TvLDH



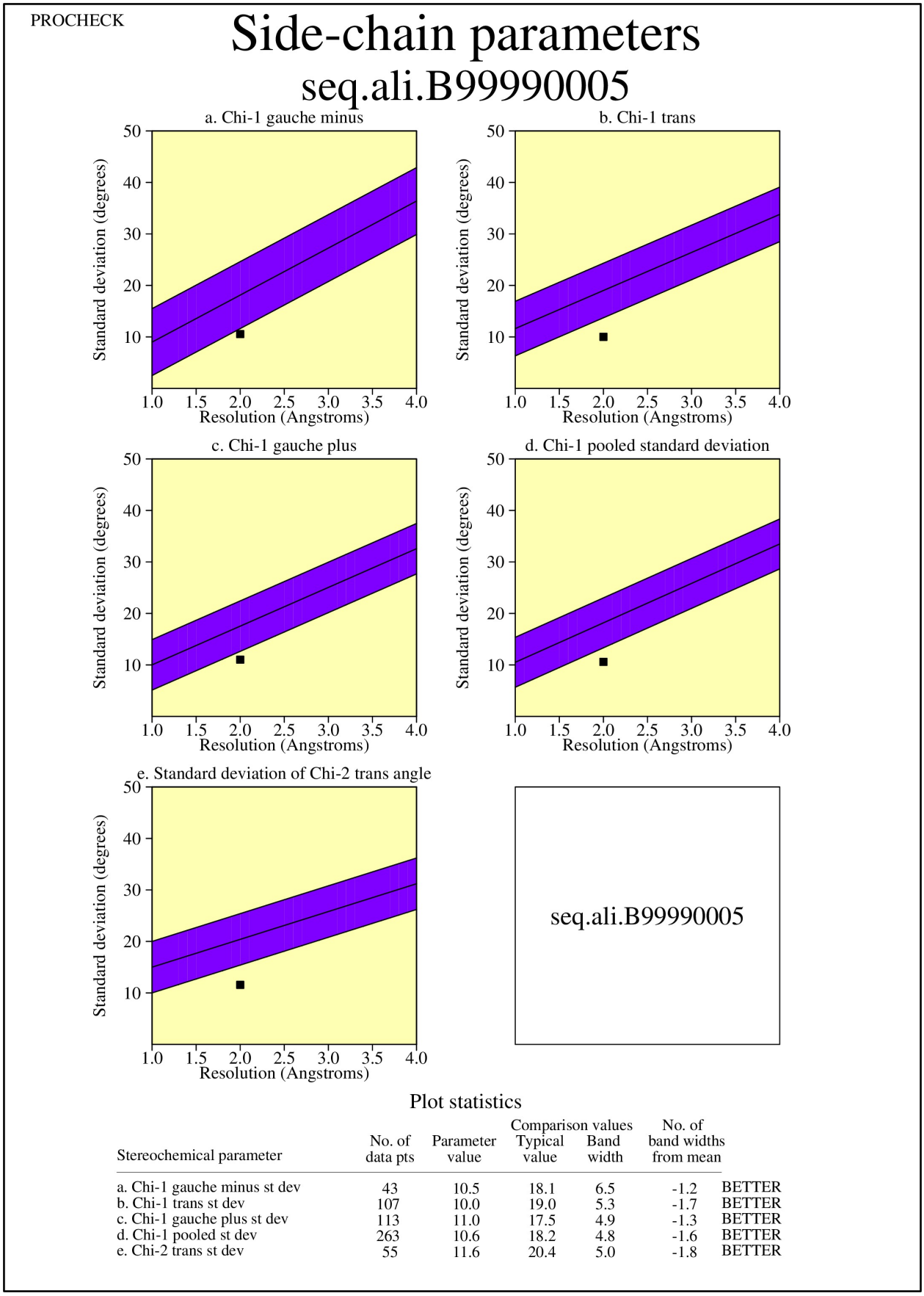
Plot statistics

Residues in most favoured regions [A,B,L]	260	88.7%
Residues in additional allowed regions [a,b,l,p]	21	7.2%
Residues in generously allowed regions [~a,~b,~l,~p]	7	2.4%
Residues in disallowed regions	5	1.7%
Number of non-glycine and non-proline residues	293	100.0%
Number of end-residues (excl. Gly and Pro)	2	
Number of glycine residues (shown as triangles)	25	
Number of proline residues	16	
Total number of residues	336	

Based on an analysis of 118 structures of resolution of at least 2.0 Angstroms and R-factor no greater than 20%, a good quality model would be expected to have over 90% in the most favoured regions.

Figura 22 – Gráfico Ramachandran da TvLDH antes de sofrer o refinamento de loops.

Modelo TvLDH

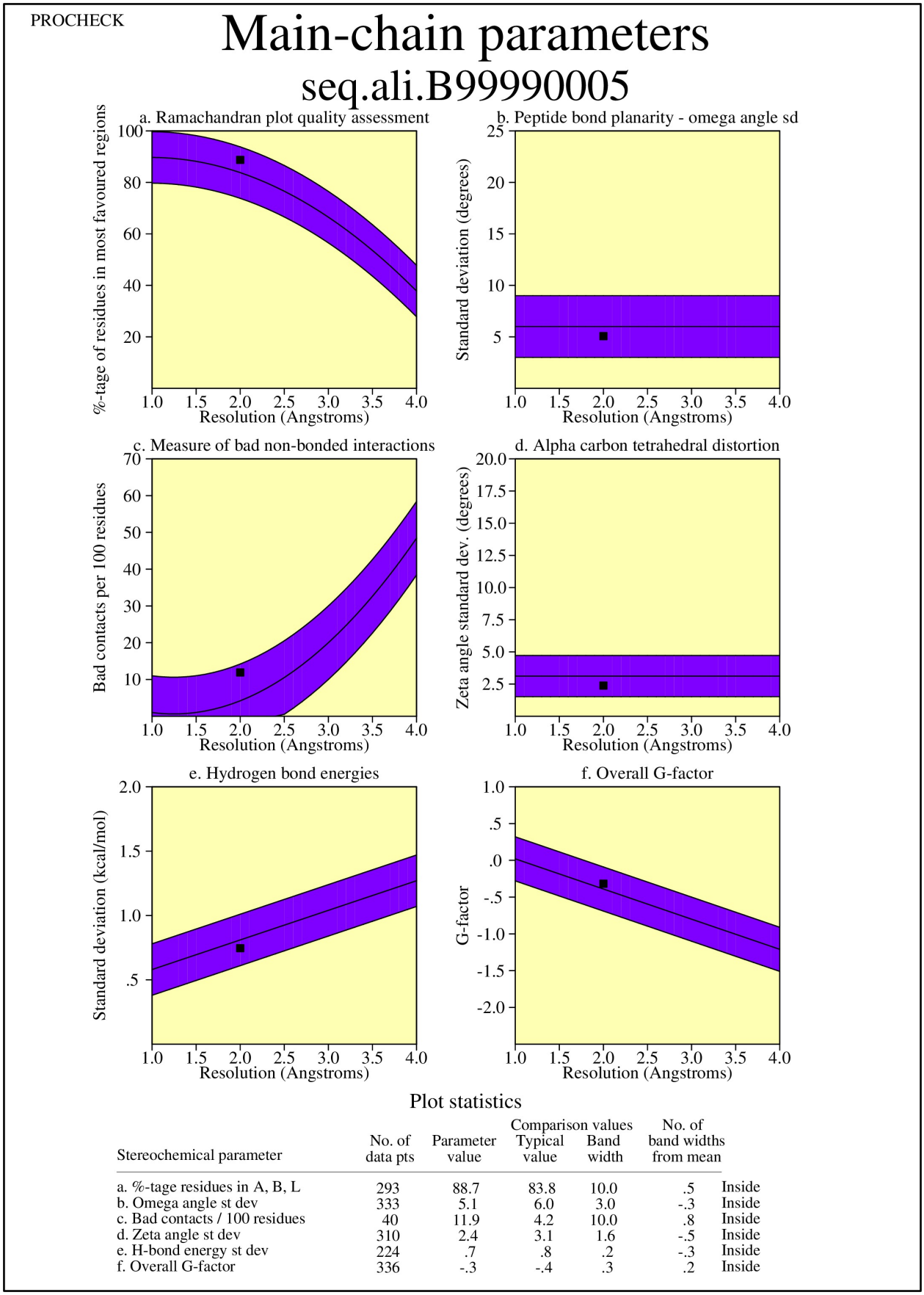


Plot statistics

Stereochemical parameter	No. of data pts	Parameter value	Comparison values Typical value	Band width	No. of band widths from mean	
a. Chi-1 gauche minus st dev	43	10.5	18.1	6.5	-1.2	BETTER
b. Chi-1 trans st dev	107	10.0	19.0	5.3	-1.7	BETTER
c. Chi-1 gauche plus st dev	113	11.0	17.5	4.9	-1.3	BETTER
d. Chi-1 pooled st dev	263	10.6	18.2	4.8	-1.6	BETTER
e. Chi-2 trans st dev	55	11.6	20.4	5.0	-1.8	BETTER

Figura 23 – Análise estereoquímica das cadeias laterais do modelo da TvLDH antes de sofrer o refinamento de loop.

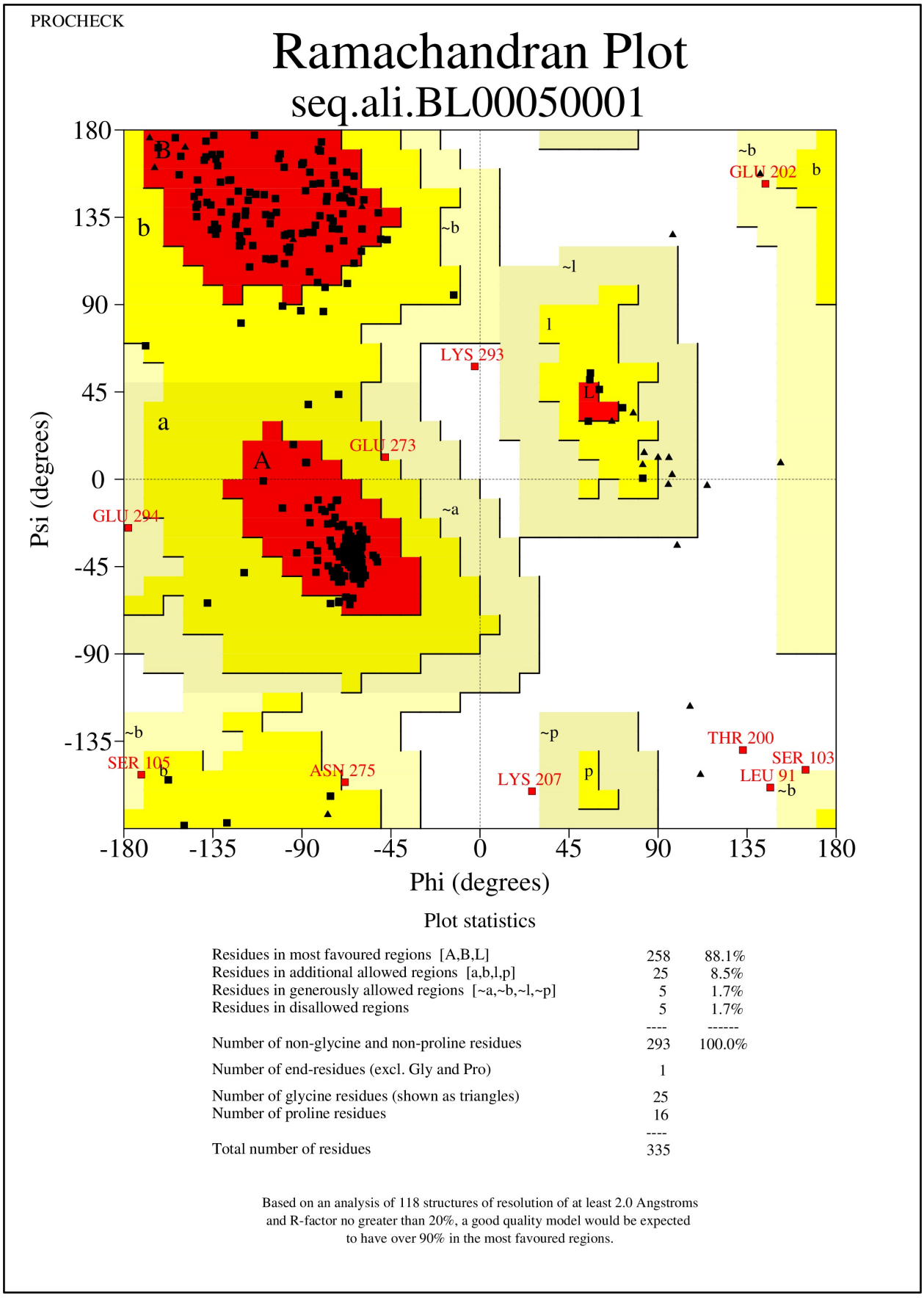
Modelo TvLDH



Stereochemical parameter	No. of data pts	Parameter value	Comparison values Typical value	Band width	No. of band widths from mean	
a. %-tage residues in A, B, L	293	88.7	83.8	10.0	.5	Inside
b. Omega angle st dev	333	5.1	6.0	3.0	-.3	Inside
c. Bad contacts / 100 residues	40	11.9	4.2	10.0	.8	Inside
d. Zeta angle st dev	310	2.4	3.1	1.6	-.5	Inside
e. H-bond energy st dev	224	.7	.8	.2	-.3	Inside
f. Overall G-factor	336	-.3	-.4	.3	.2	Inside

Figura 24 – Análise estereoquímica da cadeia principal do modelo da TvLDH antes de sofrer o refinamento de loop.

Modelo TvLDH com Loop Refinado



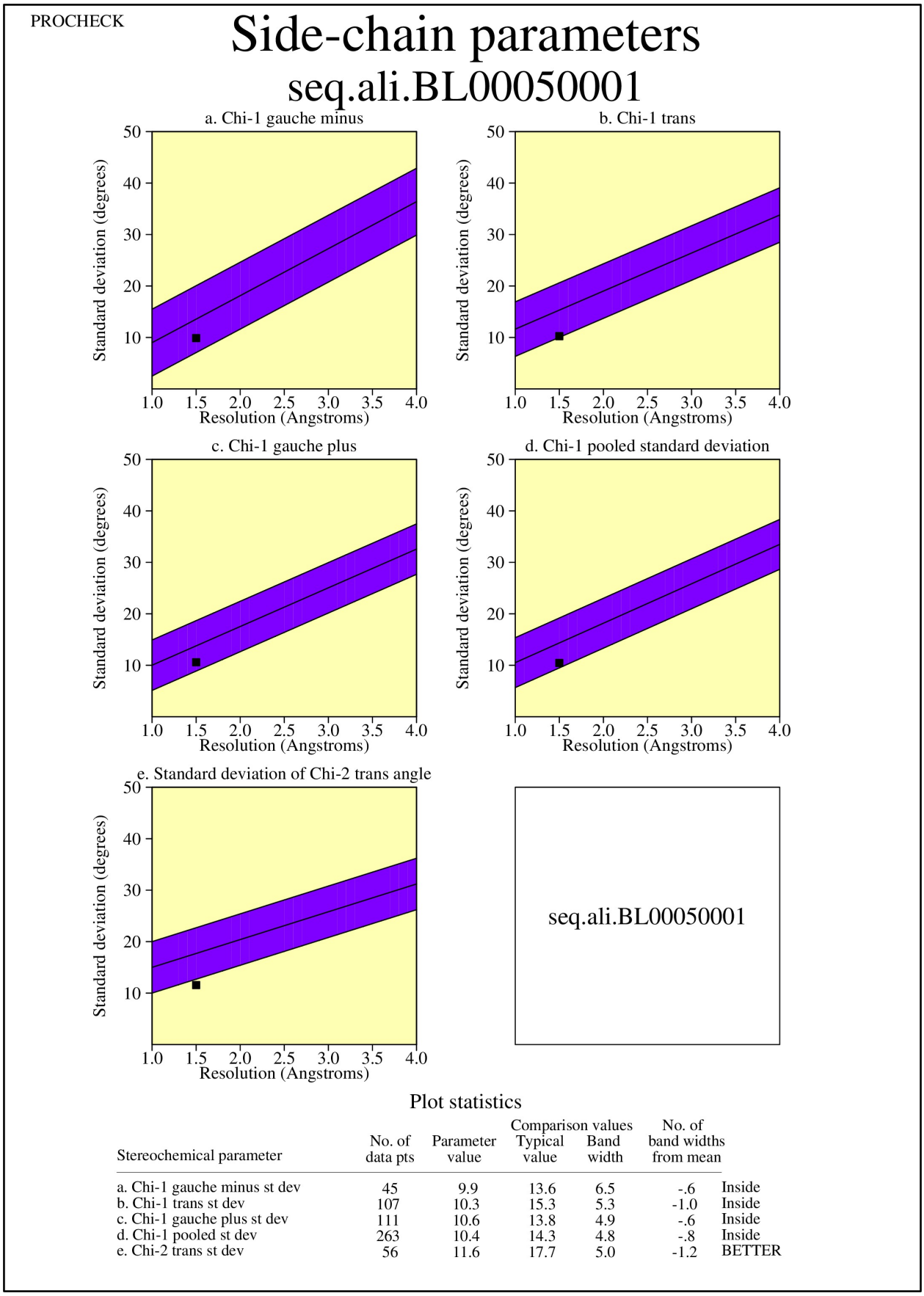
Plot statistics

Residues in most favoured regions [A,B,L]	258	88.1%
Residues in additional allowed regions [a,b,l,p]	25	8.5%
Residues in generously allowed regions [~a,~b,~l,~p]	5	1.7%
Residues in disallowed regions	5	1.7%
Number of non-glycine and non-proline residues	293	100.0%
Number of end-residues (excl. Gly and Pro)	1	
Number of glycine residues (shown as triangles)	25	
Number of proline residues	16	
Total number of residues	335	

Based on an analysis of 118 structures of resolution of at least 2.0 Angstroms and R-factor no greater than 20%, a good quality model would be expected to have over 90% in the most favoured regions.

Figura 25 – Gráfico Ramachandran da TvLDH depois de sofrer o refinamento de loops.

Modelo TvLDH com Loop Refinado

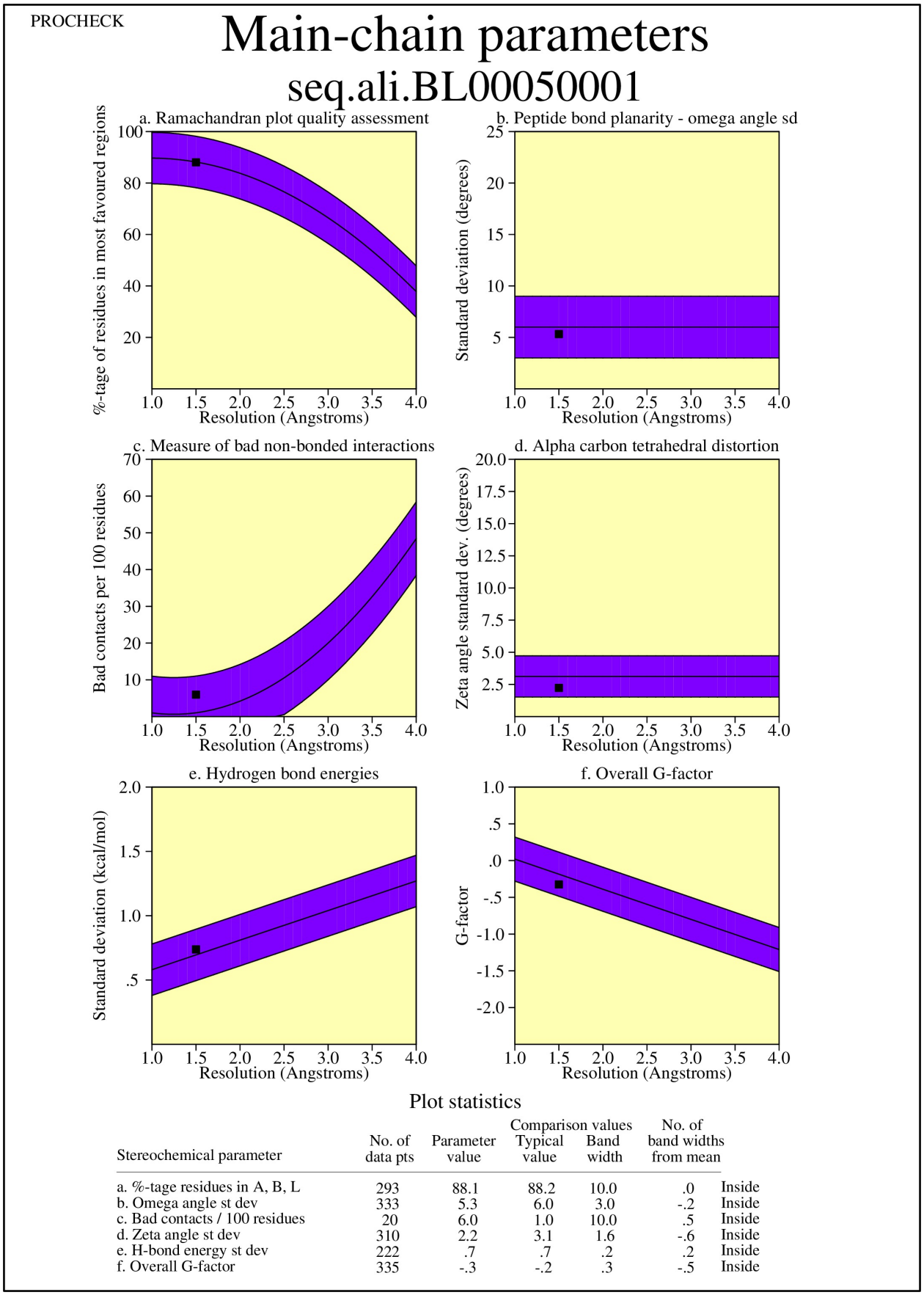


Plot statistics

Stereochemical parameter	No. of data pts	Parameter value	Comparison values Typical value	Band width	No. of band widths from mean	
a. Chi-1 gauche minus st dev	45	9.9	13.6	6.5	-.6	Inside
b. Chi-1 trans st dev	107	10.3	15.3	5.3	-1.0	Inside
c. Chi-1 gauche plus st dev	111	10.6	13.8	4.9	-.6	Inside
d. Chi-1 pooled st dev	263	10.4	14.3	4.8	-.8	Inside
e. Chi-2 trans st dev	56	11.6	17.7	5.0	-1.2	BETTER

Figura 26 – Análise estereoquímica das cadeias laterais do modelo da TvLDH após sofrer o refinamento de loop.

Modelo TvLDH com Loop Refinado



Stereochemical parameter	No. of data pts	Parameter value	Comparison values Typical value	Band width	No. of band widths from mean	
a. %-tage residues in A, B, L	293	88.1	88.2	10.0	.0	Inside
b. Omega angle st dev	333	5.3	6.0	3.0	-.2	Inside
c. Bad contacts / 100 residues	20	6.0	1.0	10.0	.5	Inside
d. Zeta angle st dev	310	2.2	3.1	1.6	-.6	Inside
e. H-bond energy st dev	222	.7	.7	.2	.2	Inside
f. Overall G-factor	335	-.3	-.2	.3	-.5	Inside

Figura 27 – Análise estereoquímica da cadeia principal do modelo da TvLDH após sofrer o refinamento de loop.

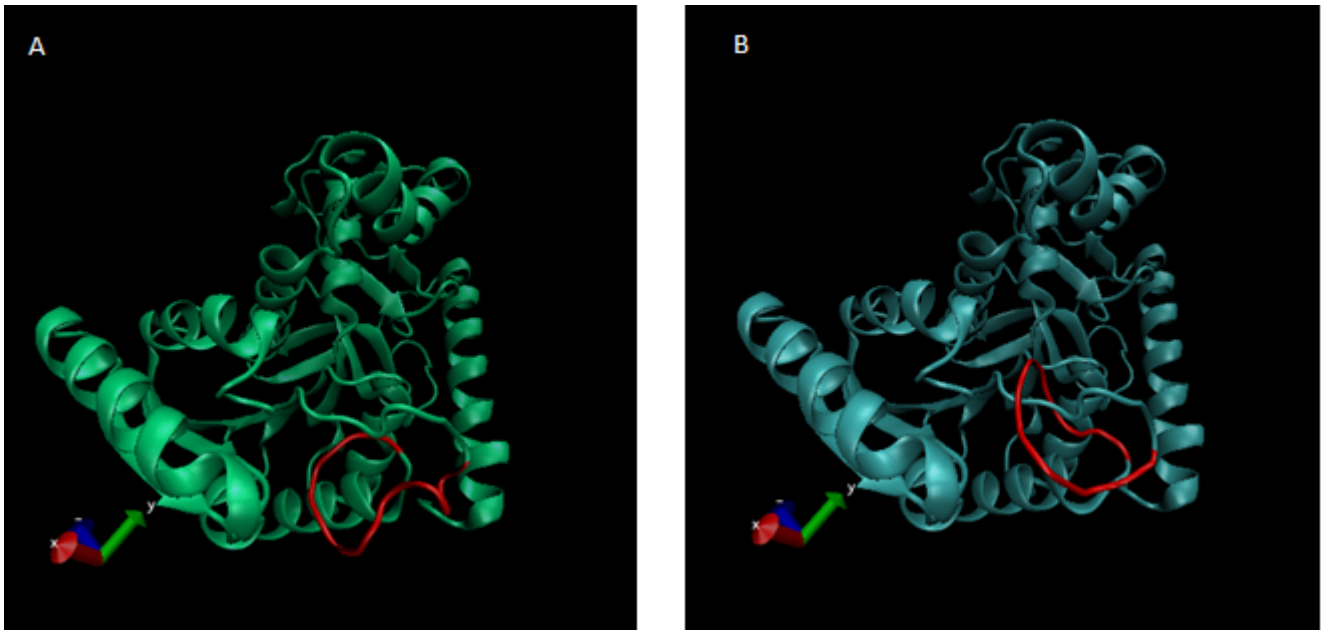


Figura 28 – (A) Modelo do TvLDH gerado pelo AutoModel exibindo a região do loop que será refinado (em vermelho) (B) Modelo com o loop refinado do TvLDH (vermelho)

ANEXO A – MANUAL DO AUTOMODEL

A.1 O QUE É MODELAGEM MOLECULAR POR HOMOLOGIA?

É um método de modelagem de proteína que basea-se no fato que a evolução criou famílias proteicas em que seus integrantes compartilham de características em comum. Utilizado a sequencia de aminoácidos que compõe uma proteína e sabendo de qual família ela pertence é possível predizer com razoável precisão como será a estrutura tridimensional dessa proteína.

A Modelagem de proteínas por homologia é divida em 4 etapas: Na primeira etapa busca-se uma ou mais proteínas homólogas a proteína que se deseja modelar que tenha uma estrutura tridimensional conhecida, Depois dessa etapa é feito o alinhamento entre essas estruturas para a determinação de regiões que foram conservadas. Esse alinhamento é utilizado na terceira etapa que é a montagem do modelo tridimensional. Após a montagem é verificado se o modelo criado é bom ou ruim através de métodos de validação.

A.2 O QUE É O AUTOMODEL?

O AutoModel é uma aplicação desenvolvida para auxiliar os usuários a criar modelos tridimensionais de proteínas através da internet. Para criar o modelo de uma seqüência o usuário necessita apenas de um arquivo com a seqüência no formato FASTA, um formato de arquivo baseado em texto onde cada aminoácido ou nucleotídeo é representado por uma letra, e o programa Cliente do AutoModel. A aplicação é capaz de fazer todo o processo semi-automaticamente, isto é, requerendo poucas interações com o usuário. Além de criar modelos o AutoModel é capaz de fazer, ao término da modelagem, uma análise dos modelos gerados.

A Modelagem utilizando o AutoModel é divido em 4 etapas:

i. Busca e Escolha de proteínas-molde;

ii. Alinhamento das sequencias;

iii. Construção do modelo tridimensional;

iv. Validação.

A.3 INICIANDO O PROGRAMA AUTOMODEL:

O AutoModel foi desenvolvido de modo que funcione nos sistemas operacionais mais comuns: Microsoft Windows, Linux e Mac OS;

Dependendo do Sistema operacional existem duas formas de iniciar o AutoModel:

i. Utilizando o ambiente gráfico do sistema operacional.

ii. Utilizando o terminal do sistema operacional.

A.3.1 Abrindo o AutoModel utilizando o Ambiente gráfico do sistema operacional Windows:

Utilizando o Windows Explorer abra a pasta do AutoModel e clique duas vezes em cima do arquivo AutoModel.bat (figura 29).

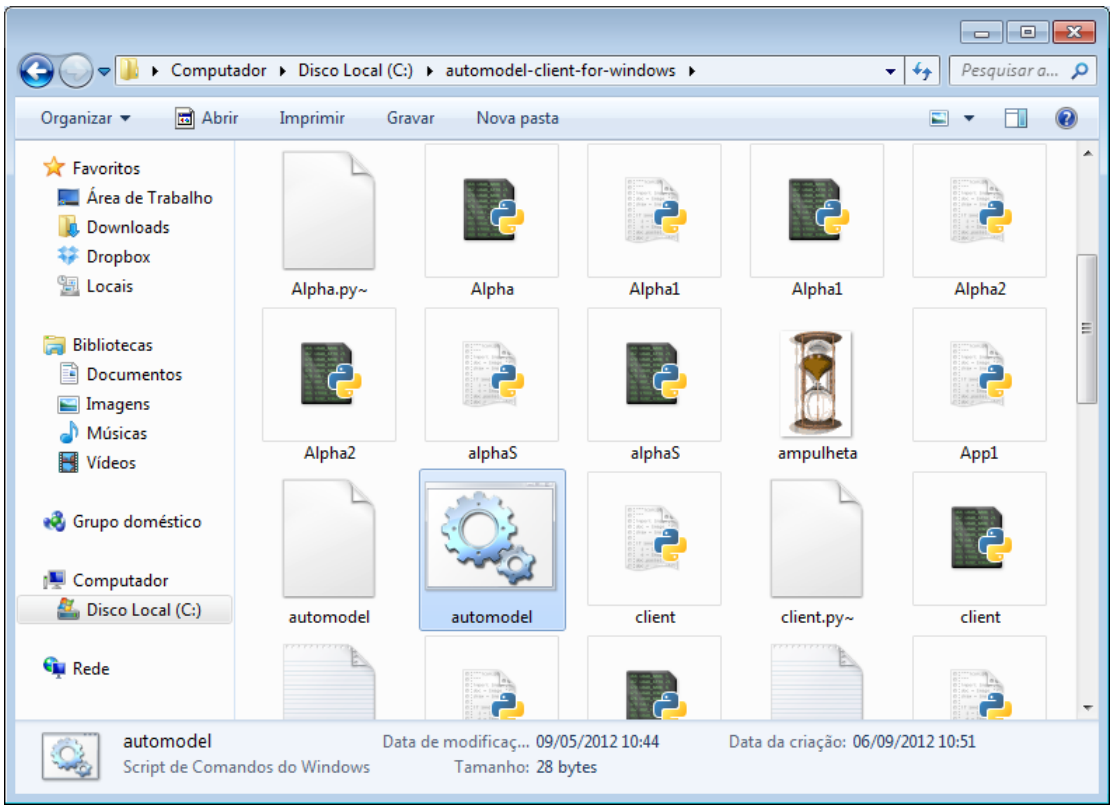


Figura 29 – Para acessar o AutoModel no Windows, basta executar o arquivo automodel.bat

A.3.2 Abrindo o AutoModel utilizando o Ambiente gráfico do sistema operacional Linux e Mac OS:

Utilizando o gerenciador de arquivos do seu operacional (Nautilus, Dolphin, Konqueror, Finder, etc.) abra a pasta do AutoModel e clique duas vezes em cima do arquivo AutoModel (figura 30).

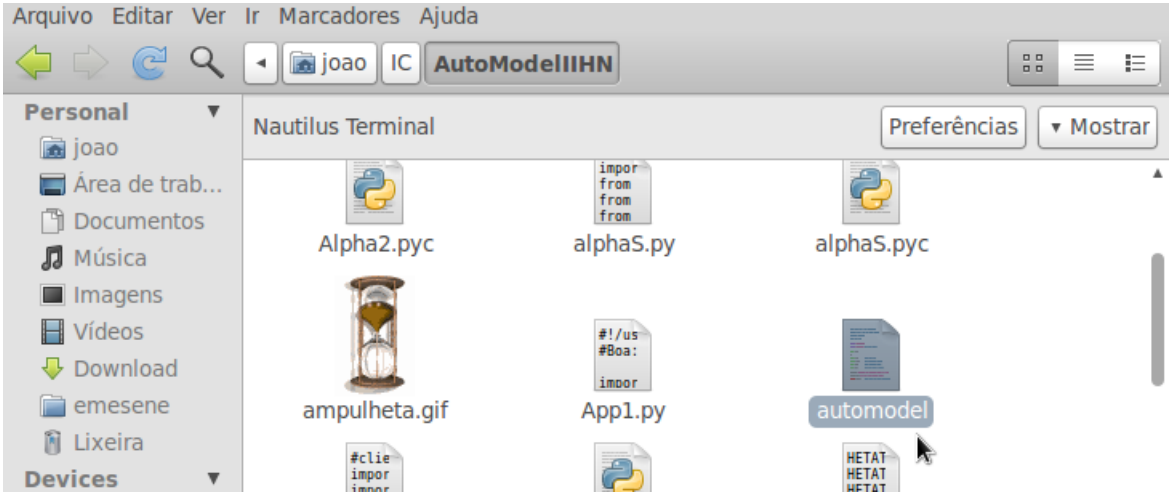


Figura 30 – Exemplo do arquivo automodel.sh sendo executado no Nautilus

A.4 CARREGANDO A PROTEÍNA DO USUÁRIO NO AUTOMODEL:

O usuário pode carregar sua sequência no AutoModel de duas formas: Um arquivo no formato FASTA ou escrevendo/colando sua sequencia na janela do AutoModel.

A.4.1 Carregando a sequencia utilizando um arquivo no formato Fasta:

Para carregar o arquivo no AutoModel basta clicar no botão *Open* no campo *Select Target Sequence* (figura 31).

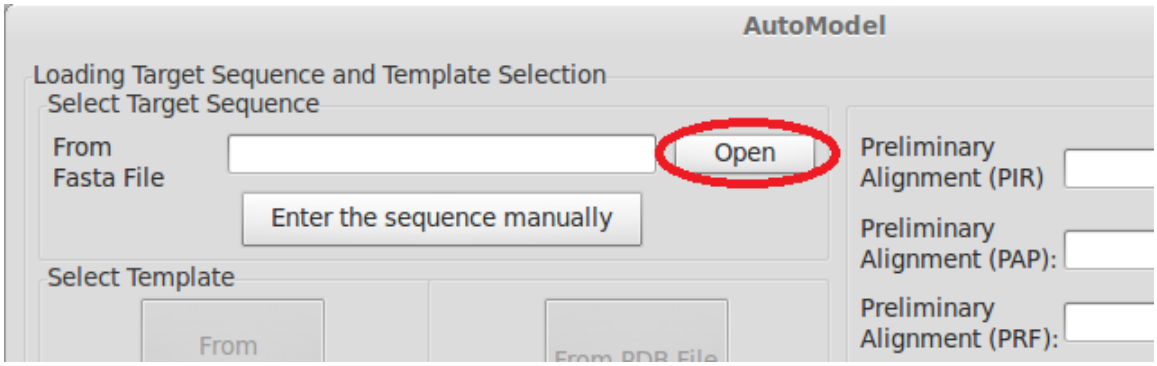


Figura 31 – Botão *Open* do campo *Select Target Sequence*

Na janela *Escolha um arquivo no formato FASTA* selecionamos o arquivo com a seqüência que desejamos carregar e depois clicamos em *Abrir* (figura 32):

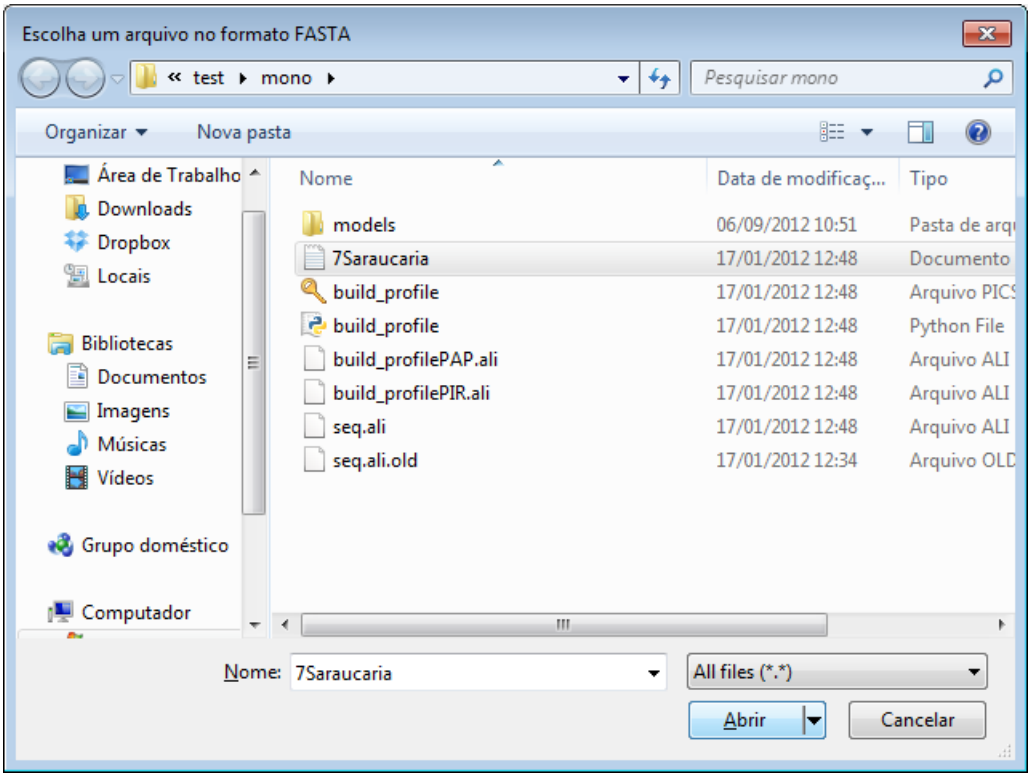


Figura 32 – Janela para a carga do arquivo fasta que possui a sequência alvo.

A.4.2 Carregando a sequencia do usuário Escrevendo/Colando na Janela do AutoModel:

A sequencia do usuário pode ser carregada no AutoModel sem a necessidade dela estar em um arquivo. Para isso, basta pressionar na janela principal do AutoModel o botão *Enter the sequence manually* (figura 33).

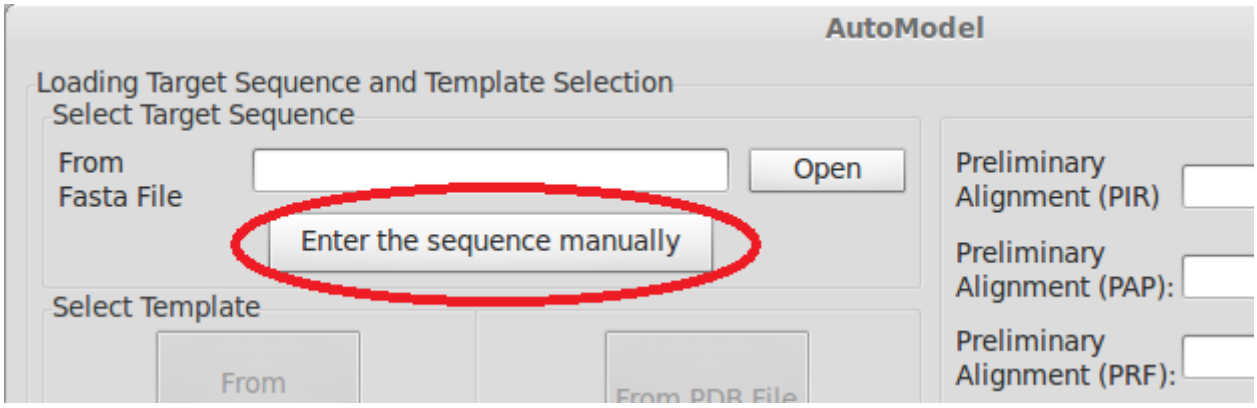


Figura 33 – Botão *Enter the sequence manually* para acesso a janela *Load Target Sequence* que permite carregar manualmente a sequencia do usuário.

E na janela *Load Target Sequence* escrever ou colar a seqüência e pressionar o botão *Save* (figura 34).

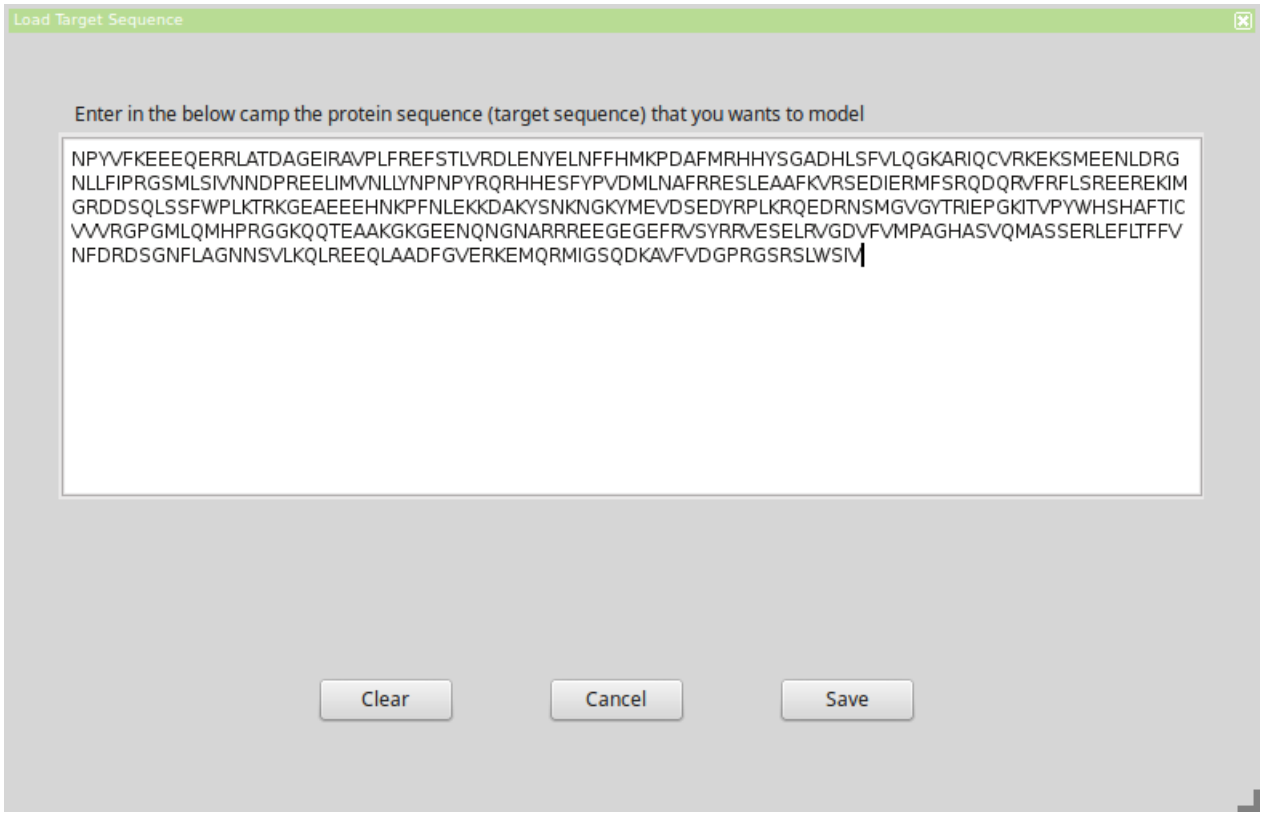


Figura 34 – Janela *Load Target Sequence* para o usuário entrar com a sequência alvo

A.5 BUSCA E ESCOLHA DE PROTEÍNAS-MOLDE (TEMPLATE):

A primeira etapa da modelagem molecular é procurar uma ou mais proteínas conhecidas que são homologas à proteína carregada pelo usuário. Isso pode ser feito de duas maneiras: Buscando em um banco de dados de estruturas conhecidas ou Carregando manualmente a proteína no formato PDB manualmente.

A.5.1 Buscando a proteína-molde em um banco de dados:

Esse método geralmente é utilizado quando não se sabe qual a família de proteínas a proteína carregada pelo usuário pertence.

No AutoModel para procurar, candidatos a proteína molde, utilizando o banco de dados basta pressionar o botão *From Database* da seção *Select Template* (figura 35).

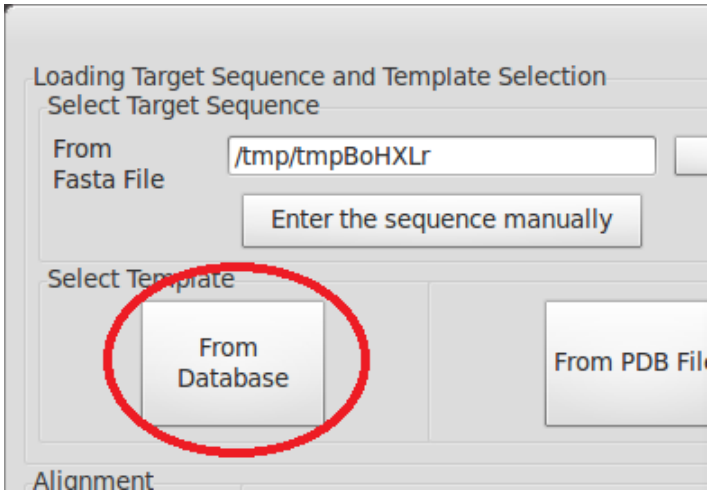


Figura 35 – Botão *From Database* da janela principal do AutoModel. Este botão permite o AutoModel busque por candidatos a proteínas-molde. Estes candidatos serão exibidos na janela *Selecting a template protein*.

A janela *Selecting a template protein* irá aparecer na tela (figura 36):

Nesta janela ao topo estará a sequencia carregada pelo usuário e abaixo da sequencia existirá uma lista de até 10 *templates* onde o usuário deverá escolher uma, utilizando como base as informações contidas do lado esquerdo da janela e através do botão [?]. Abaixo da lista de *templates* é exibida a estrutura primária do *template* selecionado.

Ao término da escolha de template pressionasse o botão *OK* para continuar a modelagem.

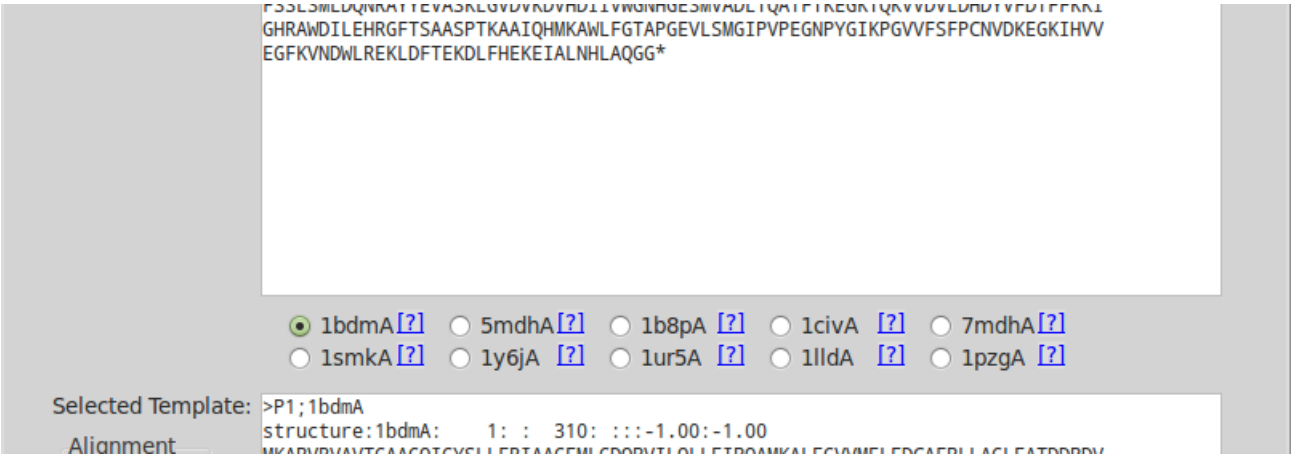


Figura 36 – Detalhe da janela *Selecting a template protein* exibindo os candidatos a proteínas molde.

A.5.2 Escolhendo a proteína-molde através de um arquivo:

Esse método é utilizado quando se conhece o *template* e sua estrutura está armazenada em um arquivo no formato PDB.

No AutoModel para carregar a proteína template devemos clicar no botão *From PDB File* da seção *Select Template* (figura 37).

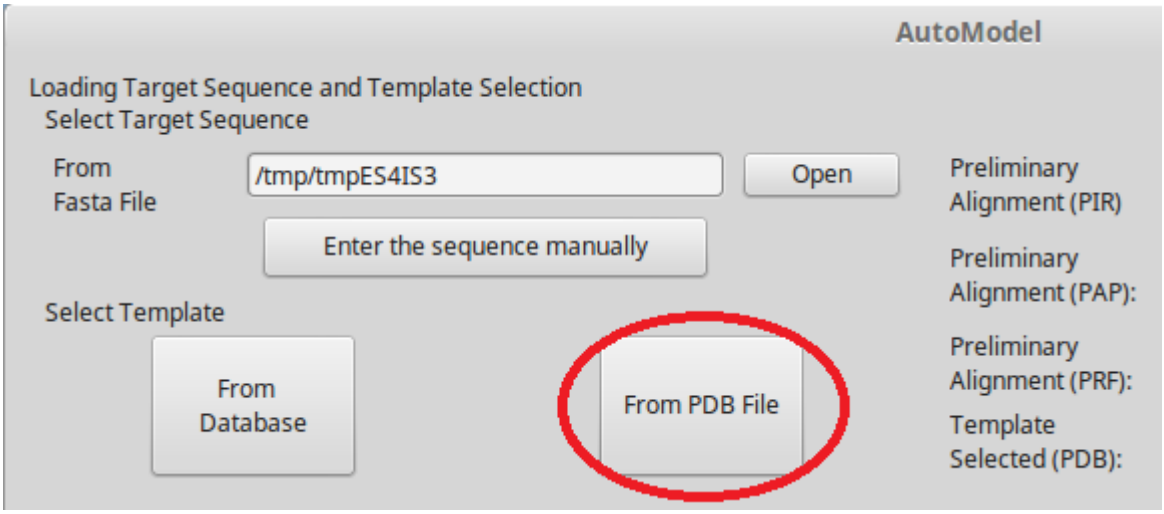


Figura 37 – Botão *From PDB File* que permite o usuário utilizar uma proteína-molde de um arquivo PDB

E na janela *Select a file in PDB format* (figura 38) seleciona-se o arquivo PDB que contém a proteína que será utilizada como molde na modelagem e pressiona-se *Open* .

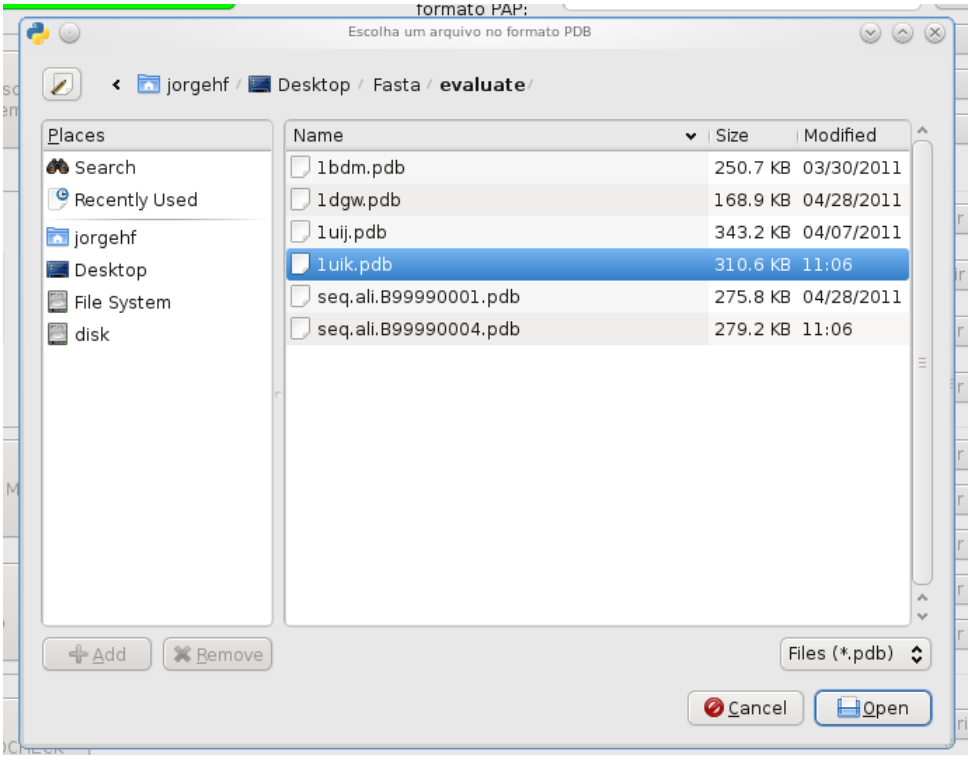


Figura 38 – Janela *Select a file in PDB format* que permite o usuário selecionar um arquivo PDB que contém uma proteína-molde.

A.6 MODIFICANDO A PROTEÍNA-MOLDE ESCOLHIDA:

Após a escolha de *Template* podemos fazer alterações nele antes de fazer o alinhamento.

Para isso, pressionamos o botão *Edit Template*, onde abrirá a janela *Selecting Chains and Heteroatoms* (figura 39).

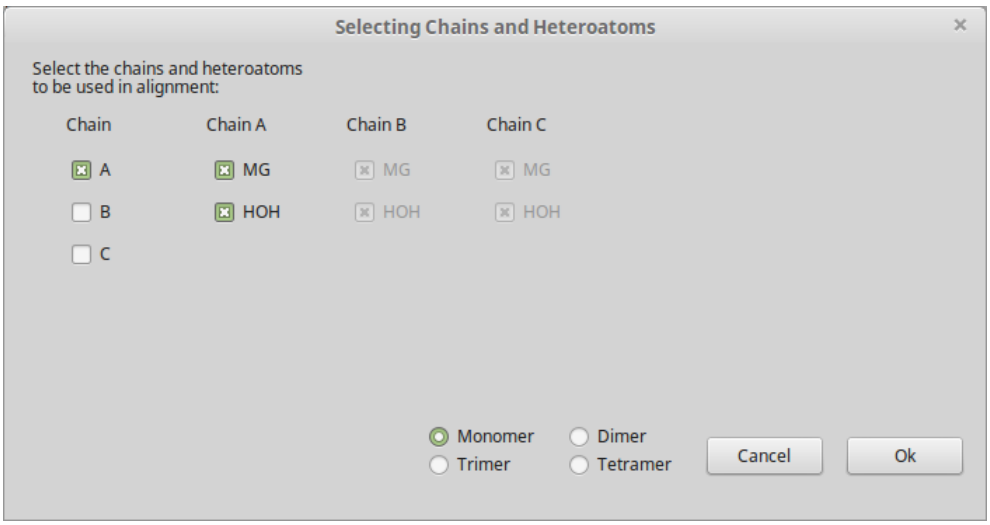


Figura 39 – Janela de edição de proteína-molde: *Edit Template*

Nessa janela o usuário pode escolher quais cadeias de proteínas irão se

utilizadas na modelagem, assim como seus respectivos heteroátomos. Também é possível escolher que a sequencia do usuário será modelada como monômero, dímero, trimero ou tetrâmero.

A.7 ALINHAMENTO DA SEQÜÊNCIA COM O TEMPLATE

Depois de escolhido a proteína que servirá como molde na modelagem, é necessário fazer o alinhamento entre a sequencia do usuário e a estrutura primária do *template*.

Esse alinhamento é necessário para que os métodos utilizados na modelagem sejam capazes de localizar a posição cartesiana da maior quantidade possíveis de átomos. Na modelagem a montagem da cadeia principal da proteína do usuário deve possuir o mesmo amoldamento da cadeia principal da proteína template.

No alinhamento é possível distinguir as regiões que estão estruturalmente conservadas das regiões variáveis. As regiões conservadas são as regiões que possuem alta similaridade entre as proteínas. As regiões variáveis foram as regiões onde não foi encontrado um correspondente estrutural com a sequencia-molde.

No AutoModel para fazer o alinhamento basta pressionar o botão *Align* na seção *Alignment* (figura 40).

Figura 40 – Botão *Align* na janela principal do AutoModel.

A.8 FAZENDO UM ALINHAMENTO RÁPIDO

Com o AutoModel você pode fazer um alinhamento rápido, sem se preocupar com os passos anteriores. Este tipo de alinhamento é útil quando o usuário tem uma sequência e a proteína que será usado como molde.

Para fazer o alinhamento rápido basta clicar em *Fast Alignment* (figura 41):

Na janela *Sequence Alignment* (figura 42) escolher a sequencia do usuário e a proteína-molde pressionando seus respectivos botões *Open*.

E no fim pressionar o botão *OK* (figura 43).

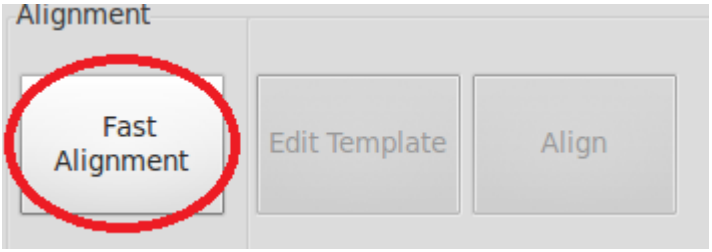


Figura 41 – Botão *Fast Alignment* na janela principal do AutoMOdel.

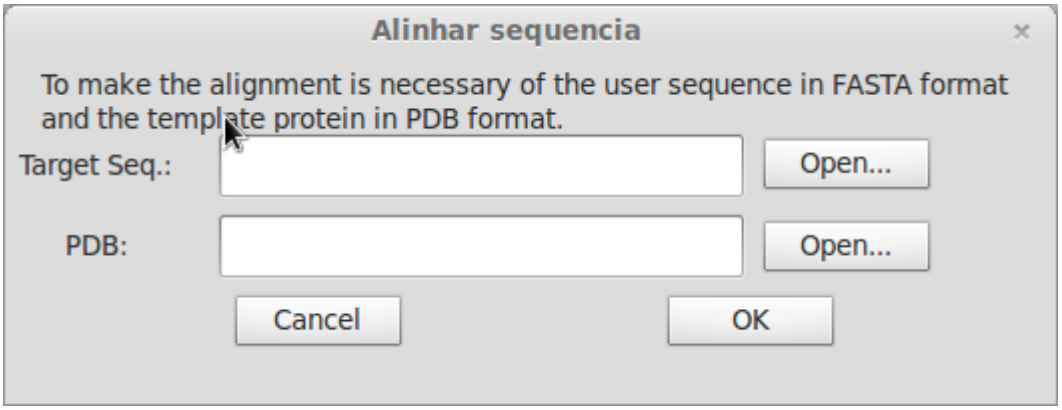


Figura 42 – Janela *Sequence Alignment* do AutoModel Client.

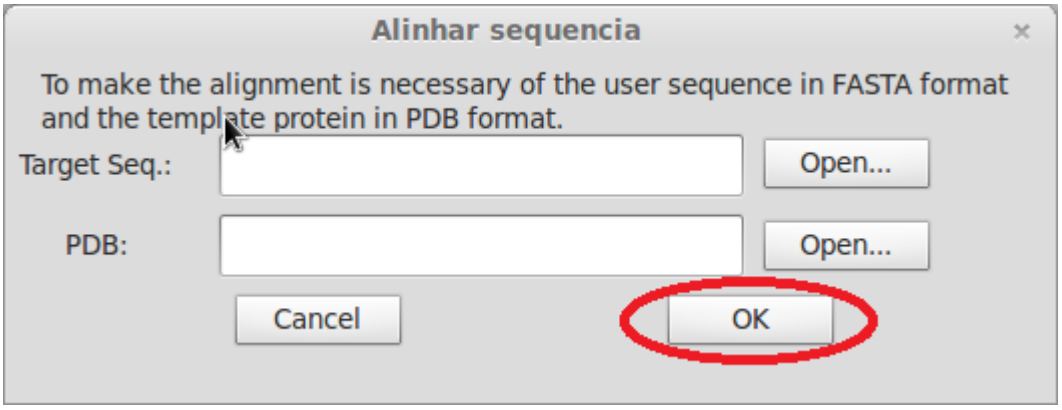


Figura 43 – Após selecionar a sequencia alvo e a proteína molde, clica-se no botão OK

A.9 MODIFICANDO OS HETEROÁTOMOS DO ALINHAMENTO

No AutoModel é possível modificar os Heteroátomos do alinhamento antes de criar os modelos tridimensionais. Para isso, basta-se clicar no botão *Change Heteroatoms* da seção *Modeling* (figura 44), onde será aberta a janela *Changing Heteroatom*.

Nessa Janela é possível alterar os heteroátomos de cada cadeia ou domínio por outro de mesma família através de menus suspensos. A mudança desses heteroátomos altera a forma que o AutoModel os tratam durante a modelagem.

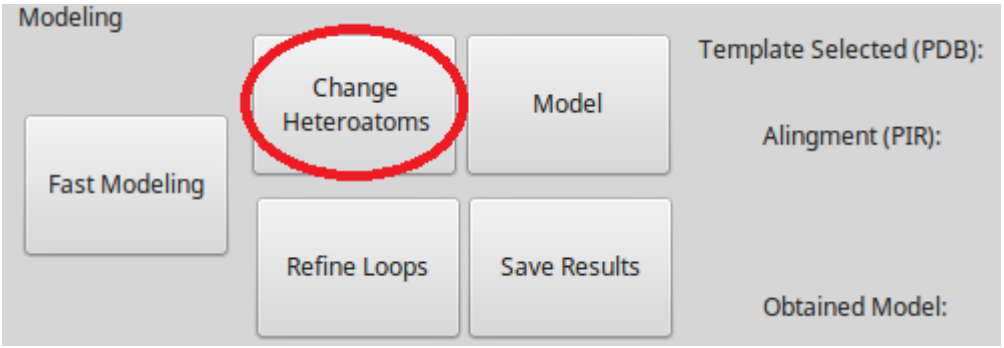


Figura 44 – Botão *Change Heteroatoms* na seção *Modeling* do AutoModel Client.

A.10 FAZENDO A MODELAGEM DO ALINHAMENTO

Após o alinhamento devemos instruir o AutoModel a fazer a modelagem dele. O que o AutoModel vai fazer nessa etapa é pegar os dados gerados pelo alinhamento e criar um arquivo onde contém a posição cartesiana de cada aminoácido ou heteroátomo.

Para fazer a modelagem basta clicar no botão *Model* (figura 45). Serão gerados 5 modelos e o melhor modelo será apresentado.

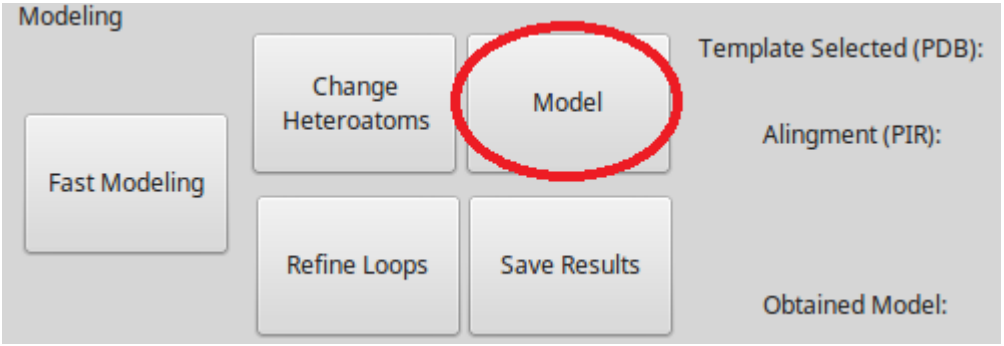


Figura 45 – Botão *Model* na seção *Modeling* do AutoModel Client.

A.11 FAZENDO UMA MODELAGEM RÁPIDA

Assim como o Alinhamento o AutoModel é capaz de fazer uma modelagem rápida, sem que o usuário tenha que se preocupar os passos anteriores. Para fazer essa modelagem é necessário que o usuário tenha o arquivo com a proteína-molde da proteína que será modelada e o arquivo do alinhamento da sequencia do usuário com a proteína-molde.

Para fazer a modelagem rápida, basta o usuário pressionar o botão de *Fast Modeling* na seção *Modeling* (figura 46).

Na janela *Template Modeling* escolher o arquivo PDB da proteína-molde e o alinhamento da sequencia do usuário com a proteína-molde pressionando seus

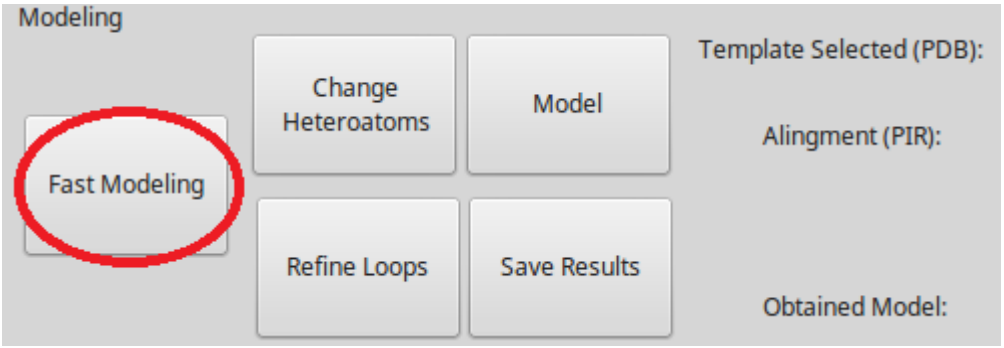


Figura 46 – Botão de *Fast Modeling* da seção *Modeling*.

respectivos botões *Open* (figura 47).

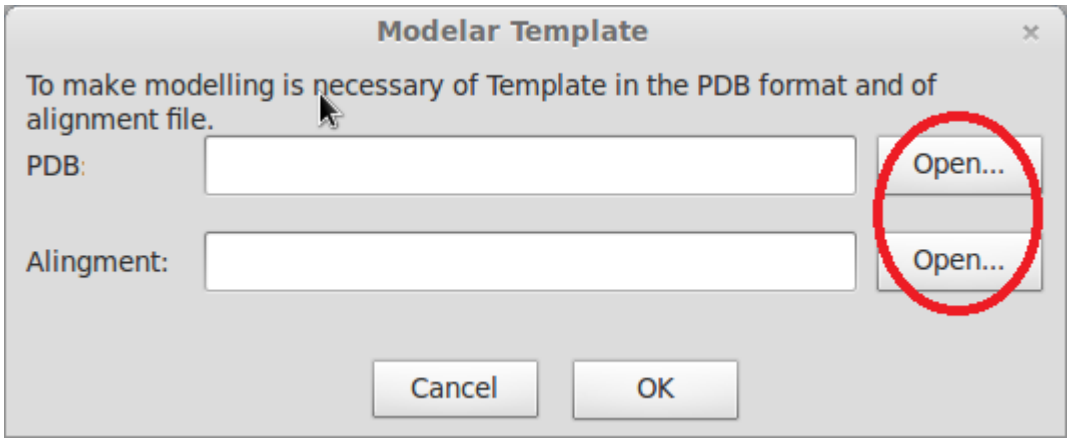


Figura 47 – Janela *Template Modeling* do AutoModel Client. Utilizando os botões *Open* o usuário pode selecionar a proteína molde (PDB) e o alinhamento para realizar uma modelagem rápida.

E no fim pressionar o botão *OK*.

A.12 VISUALIZANDO E SALVANDO OS RESULTADOS

Ao término da modelagem, o usuário deverá clicar no botão *open* de *Obtaneid Model* para poder visualizar o modelo gerado (figura 48).

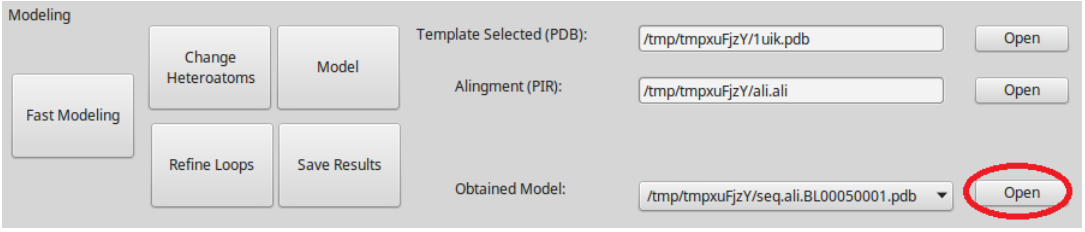


Figura 48 – Botão utilizado para visualizar o modelo gerado pelo AutoModel.

O modelo será apresentado usando o aplicativo VMD (figura 49).

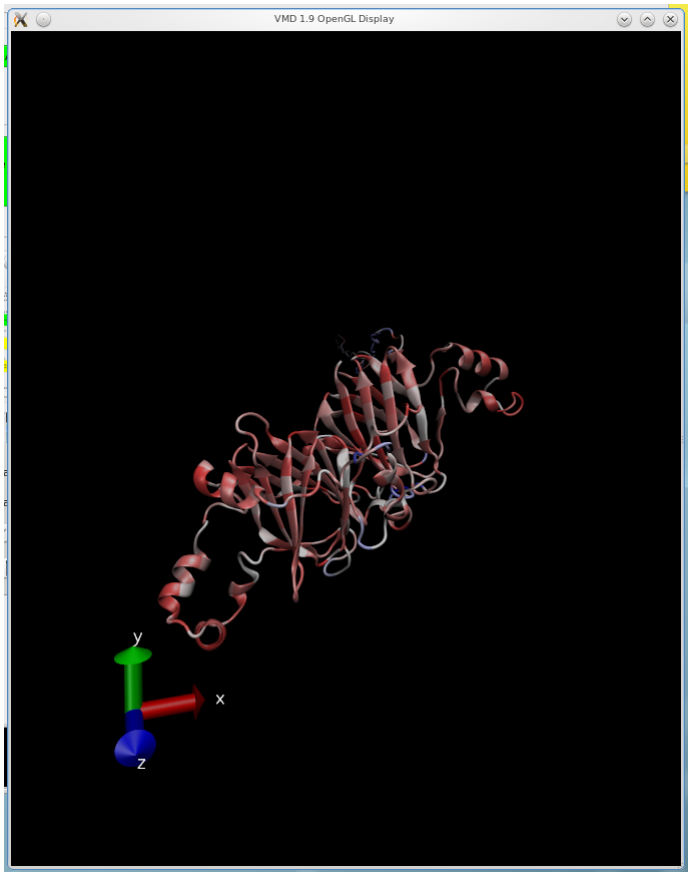


Figura 49 – Software VMD exibindo um modelo gerado pelo AutoModel.

A.13 VALIDANDO O MODELO GERADO

Validar o modelo gerado é uma etapa muito importante que pode ser executado em diferentes níveis estruturais do modelo. A qualidade do empacotamento global da proteína, erros estruturais e parâmetros estereoquímicos devem ser avaliados.

O modelo de uma proteína deve ter uma boa qualidade em sua estrutura terciária (tridimensional). Isso depende de vários fatores como a proteína escolhida como molde o alinhamento gerado e etc. Também deve-se verificar se existem regiões com diferenças conformacionais não explicadas entre os elementos de estrutura secundária (regiões conservadas) das estruturas-molde e da estrutura–modelada. No AutoModel pode-se validar o modelo de duas maneiras:

i. Através do score de energia DOPE;

ii. Através da análise dos relatórios gerados pelo programa PROCHECK;

A.13.1 Validando através do score de energia DOPE

No AutoModel 0.5 a validação é feita automaticamente após a modelagem, sendo exibida assim o gráfico de escore de energia DOPE. No entanto, é possível

visualizar este gráfico, também, clicando no botão *Open* do campo *DOPE Score* (figuras 50 e 51).

Figura 50 – Botão *Open* do campo *DOPE Score*, utilizado para visualizar o gráfico de energia DOPE.

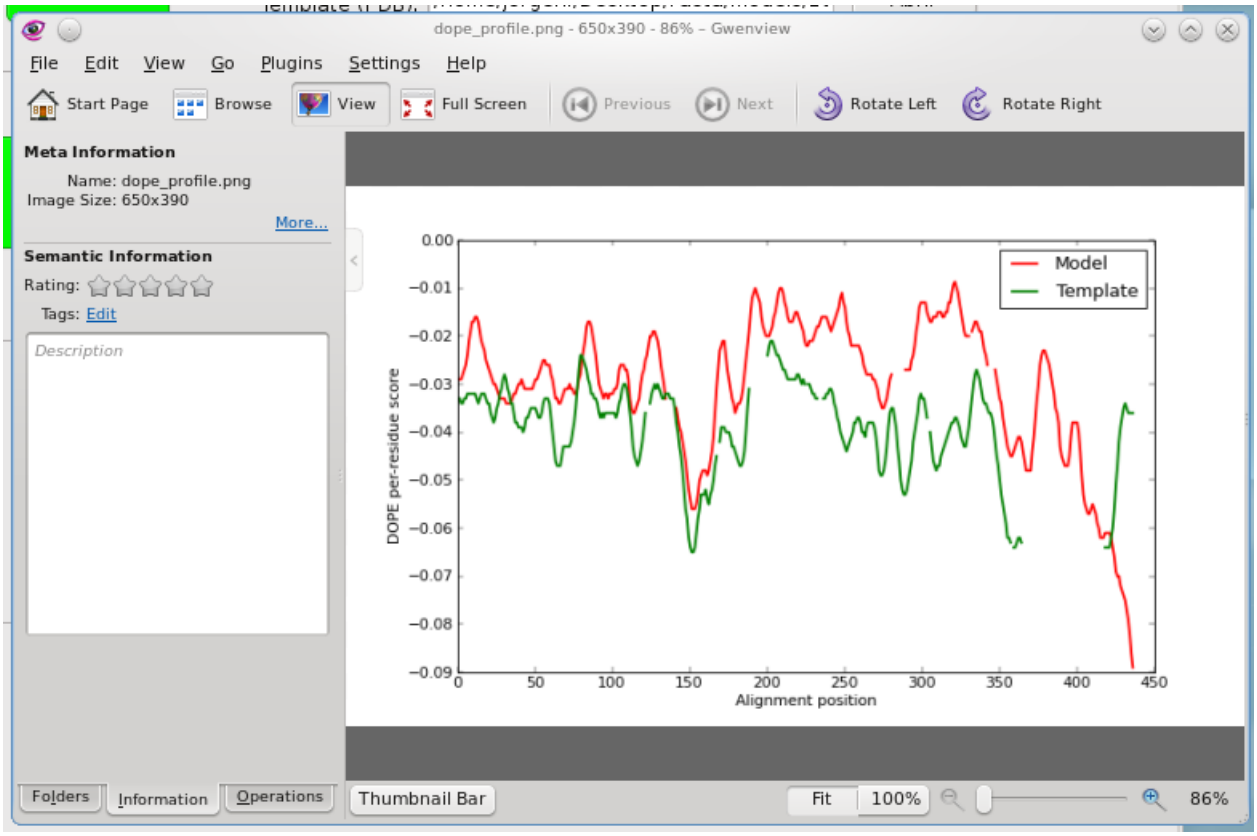


Figura 51 – Gráfico de escore DOPE gerado pelo AutoModel.

A.13.2 Validando através do programa PROCHECK.

O PROCHECK® é um programa que gera várias informações estatísticas do modelo gerado.

Para acessar os relatórios gerados pelo PROCHECK basta clicar no botão *Evaluate* na seção *Evaluate* (figura 52).

Isso criará uma pasta com vários relatórios sobre a modelagem (figura 53).

Figura 52 – Botão *Evaluate* na seção *Evaluate*

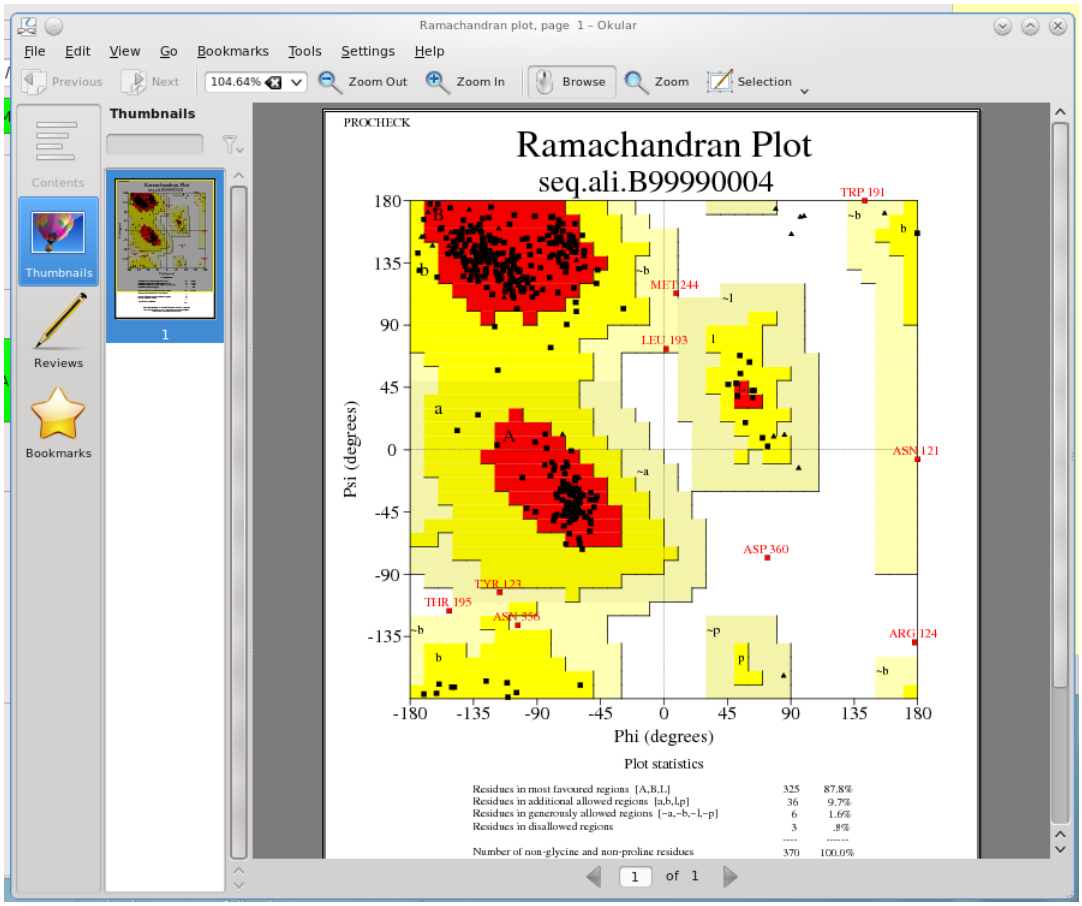


Figura 53 – *Ramachandran Plot*, um dos relatórios gerado pelo AutoModel, utilizando o Procheck.

A.14 REFINANDO O MODELO GERADO

Muitas vezes na etapa de validação é verificado que em algumas regiões do modelo gerado a qualidade está muito ruim. Essas regiões mal modeladas normalmente compreendem as regiões de loops. O AutoModel possui uma opção que permite remodelar estas regiões. Para aciona-la basta clicar no botão *Refine Loops* da seção *Modeling* (figura 54):

Após clicar neste botão será aberta a seguinte janela (figura 55):

Nesta janela é possível delimitar a região (*loop*) na qual se deseja refinar (figura 56).

Ao final basta pressionar o botão *Ok* para que se inicie o refinamento. Ao termino do refinamento será gerado um novo modelo que ficará acessível no campo

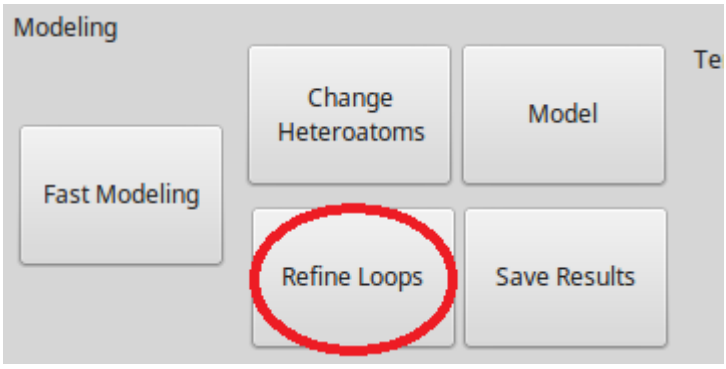


Figura 54 – Botão *Refine Loops* da seção *Modeling*

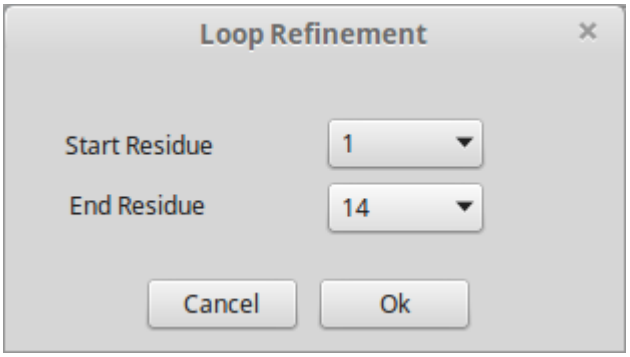


Figura 55 – Janela *Loop Refinement* do AutoModel Client.

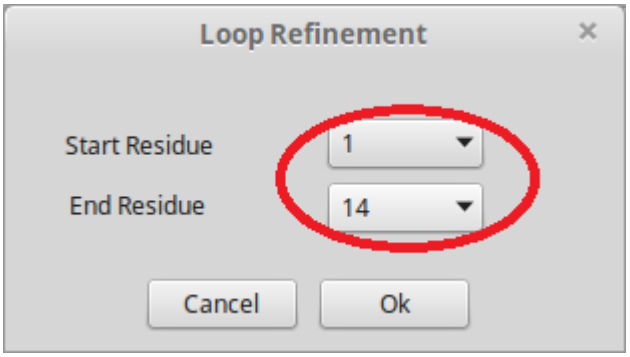


Figura 56 – Os campos *Start Residue* e *End Residue* devem ser utilizados para delimitar a região de *loop* que será remodelado

Obtained Model (figura 57).

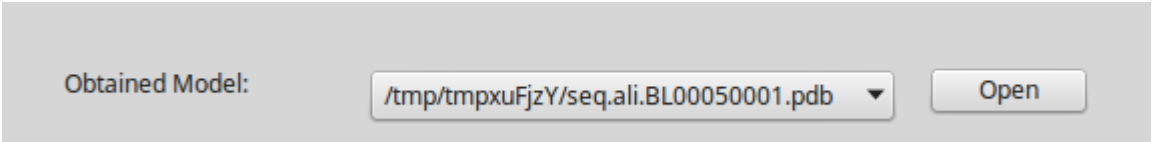


Figura 57 – Campo *Obtained Model* para acesso do modelo gerado após o refinamento de *loops*.

Após o refinamento da região de *loop* é possível visualizar a qualidade do novo modelo gerado através do botão *Compare Models* na janela principal (figura 58).

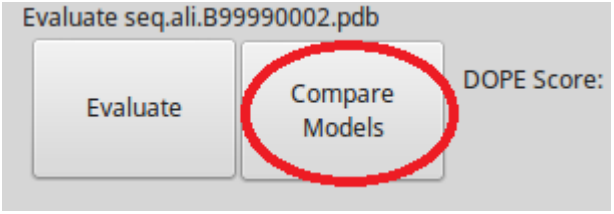


Figura 58 – O botão *Compare Models* exibe um gráfico para a comparação entre os modelos gerados e a proteína molde utilizada.

Após uns instantes será exibido na tela um gráfico com o calculo do escore DOPE para cada resíduo similar ao da etapa de validação (figura 59):

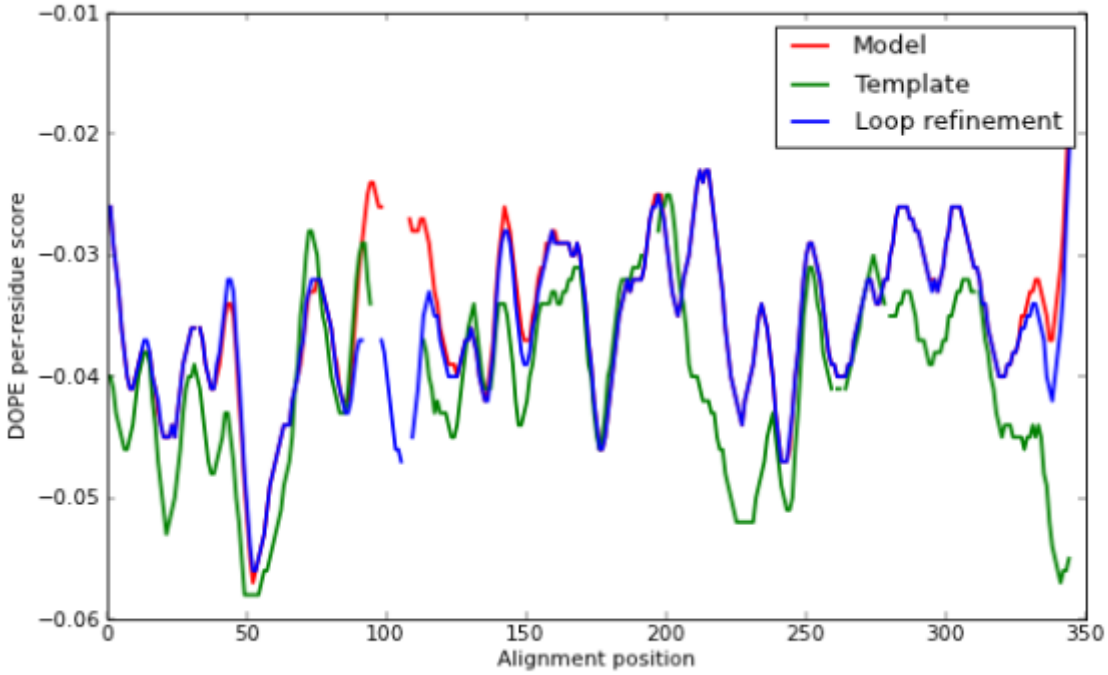


Figura 59 – Gráfico com o escore DOPE por resíduo da proteína-molde, modelo gerado e modelo com *loop* refinado

A.15 REINICIANDO A MODELAGEM

A qualquer momento a modelagem pode ser reiniciada com a utilização do botão *Reset* (figura 60).

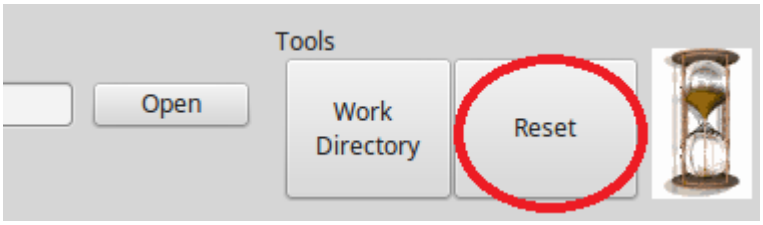


Figura 60 – Botão *Reset* na janela principal do AutoModel Client.

A.16 ABRINDO A PASTA DE MODELAGEM

Para acessar os arquivos gerados durante a modelagem, basta clicar no botão *Work Directory* da seção *Tools* (figura 61).

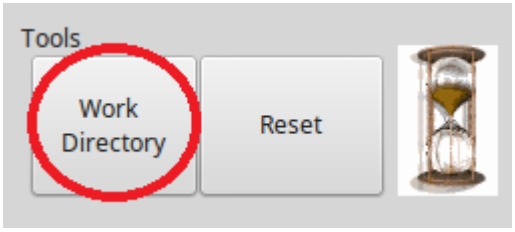


Figura 61 – Botão *Work Directory* da seção *Tools* na janela principal do AutoModel Client.

REFERÊNCIAS

ALTSCHUL, S. F. et al. Basic local alignment search tool. *Journal of molecular biology*, Elsevier, v. 215, n. 3, p. 403–410, 1990. 4

ATTWOOD, T. et al. *Concepts, historical milestones and the central place of bioinformatics in modern biology: a European perspective*. [S.l.]: INTECH Open Access Publisher, 2011. 1

BATEMAN, A. et al. The pfam protein families database. *Nucleic acids research*, Oxford Univ Press, v. 32, n. suppl 1, p. D138–D141, 2004. 10

BERMAN, H.; HENRICK, K.; NAKAMURA, H. Announcing the worldwide protein data bank. *Nature Structural & Molecular Biology*, Nature Publishing Group, v. 10, n. 12, p. 980–980, 2003. 4

CHANG, J.-M.; TOMMASO, P. D.; NOTREDAME, C. Tcs: a new multiple sequence alignment reliability measure to estimate alignment accuracy and improve phylogenetic tree reconstruction. *Molecular biology and evolution*, SMBE, p. msu117, 2014. 1

CHEN, V. B. et al. Molprobity: all-atom structure validation for macromolecular crystallography. *Acta Crystallographica Section D: Biological Crystallography*, International Union of Crystallography, v. 66, n. 1, p. 12–21, 2009. 6

COCK, P. J. et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, Oxford Univ Press, v. 25, n. 11, p. 1422–1423, 2009. 17

CORPET, F. Multiple sequence alignment with hierarchical clustering. *Nucleic acids research*, Oxford Univ Press, v. 16, n. 22, p. 10881–10890, 1988. 6

DALL'AGNO, K. C.; SOUZA, O. N. de. An expert protein loop refinement protocol by molecular dynamics simulations with restraints. *Expert Systems with Applications*, Elsevier, v. 40, n. 7, p. 2568–2574, 2013. 11

DAVID, W. M. *Bioinformatics: sequence and genome analysis*. [S.l.]: Cold spring harbor laboratory press New York, 2001. 2

DELANO, W. L. The pymol molecular graphics system. 2002. 6

DU, Q.; XIE, N.; HUANG, R. Recent development of peptide drugs and advance on theory and methodology of peptide inhibitor design. *Medicinal chemistry (Shariqah (United Arab Emirates))*, 2014. 1

EDDY, S. R. Accelerated profile hmm searches. *PLoS computational biology*, Public Library of Science, v. 7, n. 10, p. e1002195, 2011. 10

EDGAR, R. C. Muscle: a multiple sequence alignment method with reduced time and space complexity. *BMC bioinformatics*, BioMed Central Ltd, v. 5, n. 1, p. 113, 2004. 11

EDGAR, R. C. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, Oxford Univ Press, v. 32, n. 5, p. 1792–1797, 2004. 11

ESWAR, N. et al. Tools for comparative protein structure modeling and analysis. *Nucleic acids research*, Oxford Univ Press, v. 31, n. 13, p. 3375–3380, 2003. 6, 7, 12

ESWAR, N. et al. Comparative protein structure modeling using modeller. In: ____. *Current Protocols in Bioinformatics*. John Wiley & Sons, Inc., 2002. ISBN 9780471250951. Disponível em: <http://dx.doi.org/10.1002/0471250953.bi0506s15>. 13

FILHO, J. L. de A. *AutoModel: aplicação Web para modelagem de proteinas*. Monografia (Graduação), Campos dos Goytacazes - RJ, 2013. 8

FILHO, O. A. S.; ALENCASTRO, R. B. d. Modelagem de proteínas por homologia. *Química Nova*, SciELO Brasil, v. 26, n. 2, p. 253–259, 2003. 4, 6

FILHO, R. B. d. A. O. A. S. *Modelagem de proteínas por homologia*. [S.l.]: scielo, 2003. 253 - 259 p. 27

FINN, R. D. et al. Pfam: the protein families database. *Nucleic acids research*, Oxford Univ Press, p. gkt1223, 2013. 17

FINN, R. D.; CLEMENTS, J.; EDDY, S. R. Hmmer web server: interactive sequence similarity searching. *Nucleic acids research*, Oxford Univ Press, p. gkr367, 2011. 4, 10

FISER, A.; DO, R. K. G.; ŠALI, A. Modeling of loops in protein structures. *Protein science*, Wiley Online Library, v. 9, n. 9, p. 1753–1773, 2000. 12, 35

FISER, A.; ŠALI, A. Modeller: generation and refinement of homology-based protein structure models. *Methods in enzymology*, Elsevier, v. 374, p. 461–491, 2003. 12

FLOUDAS, C. Computational methods in protein structure prediction. *Biotechnology and bioengineering*, Wiley Online Library, v. 97, n. 2, p. 207–213, 2007. 4

GARVIE, E. I. Bacterial lactate dehydrogenases. *Microbiological reviews*, American Society for Microbiology (ASM), v. 44, n. 1, p. 106, 1980. 37

GERALD, K. *Cell and molecular biology: concepts and experiments*. [S.l.]: John Wiley and Sons, Hoboken, NJ, 2005. 2

GILISSEN, C. et al. Disease gene identification strategies for exome sequencing. *European Journal of Human Genetics*, Nature Publishing Group, v. 20, n. 5, p. 490–497, 2012. 1

GOUJON, M. et al. A new bioinformatics analysis tools framework at embl–ebi. *Nucleic acids research*, Oxford Univ Press, v. 38, n. suppl 2, p. W695–W699, 2010. 11

HILLISCH, A.; PINEDA, L. F.; HILGENFELD, R. Utility of homology models in the drug discovery process. *Drug discovery today*, Elsevier, v. 9, n. 15, p. 659–669, 2004. 4

HOGEWEG, P. The roots of bioinformatics in theoretical biology. *PLoS computational biology*, Public Library of Science, v. 7, n. 3, p. e1002021, 2011. 1

HUMPHREY, W.; DALKE, A.; SCHULTEN, K. Vmd: visual molecular dynamics. *Journal of molecular graphics*, Elsevier, v. 14, n. 1, p. 33–38, 1996. 6

HUNTER, J. D. Matplotlib: A 2d graphics environment. *Computing in science and engineering*, v. 9, n. 3, p. 90–95, 2007. 16

KELLEY, L. A.; STERNBERG, M. J. Protein structure prediction on the web: a case study using the phyre server. *Nature protocols*, Nature Publishing Group, v. 4, n. 3, p. 363–371, 2009. 6

KELLY, C. A. et al. Determinants of protein thermostability observed in the 1.9-. ang. crystal structure of malate dehydrogenase from the thermophilic bacterium thermus flavus. *Biochemistry*, ACS Publications, v. 32, n. 15, p. 3913–3922, 1993. 37

KIHARA, D. et al. Touchstone: an ab initio protein structure prediction method that uses threading-based tertiary restraints. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 98, n. 18, p. 10125–10130, 2001. 4

KRIEGER, E.; NABUURS, S.; VRIEND, G. Homology modeling. *Methods of biochemical analysis*, v. 44, p. 509, 2003. 3, 6, 12, 35

LAMBERT, C. et al. Esypred3d: Prediction of proteins 3d structures. *Bioinformatics*, Oxford Univ Press, v. 18, n. 9, p. 1250–1256, 2002. 6

LARKIN, M. A. et al. Clustal w and clustal x version 2.0. *Bioinformatics*, Oxford Univ Press, v. 23, n. 21, p. 2947–2948, 2007. 6

LASKOWSKI, R. A. et al. Procheck: a program to check the stereochemical quality of protein structures. *Journal of applied crystallography*, International Union of Crystallography, v. 26, n. 2, p. 283–291, 1993. 6

LEHNINGER, A.; NELSON, D.; COX, M. *Lehninger Principles of Biochemistry*. W. H. Freeman, 2005. ISBN 9780716743392. Disponível em: <https://books.google.com.br/books?id=7chAN0UY0LYC>. XI, 2

LIWO, A. et al. Protein structure prediction by global optimization of a potential energy function. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 96, n. 10, p. 5482–5485, 1999. 4

MAREUIL, F. et al. Grid computing for improving conformational sampling in nmr structure calculation. *Bioinformatics*, Oxford Univ Press, v. 27, n. 12, p. 1713–1714, 2011. 1

MOULT, J. et al. Critical assessment of methods of protein structure prediction (casp)—round x. *Proteins: Structure, Function, and Bioinformatics*, Wiley Online Library, v. 82, n. S2, p. 1–6, 2014. 1

MURRAY, R. et al. *Harper's illustrated biochemistry (LANGE basic science)*. [S.l.]: McGraw-Hill Medical, 2003. 2

OKONECHNIKOV, K. et al. Unipro ugene: a unified bioinformatics toolkit. *Bioinformatics*, Oxford Univ Press, v. 28, n. 8, p. 1166–1167, 2012. 11

OUZOUNIS, C. A. Rise and demise of bioinformatics? promise and progress. *PLoS computational biology*, Public Library of Science, v. 8, n. 4, p. e1002487, 2012. 1

PEARSON, W. R.; LIPMAN, D. J. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 85, n. 8, p. 2444–2448, 1988. 6, 10

PYTHON SOFTWARE FOUNDATION. *PEP 8*: Style guide for python code. [S.l.], 2013. 15

RAPPIN, N.; DUNN, R. *wxPython in Action*. [S.l.]: Manning, 2006. 15

ROY, A.; KUCUKURAL, A.; ZHANG, Y. I-tasser: a unified platform for automated protein structure and function prediction. *Nature protocols*, Nature Publishing Group, v. 5, n. 4, p. 725–738, 2010. 4

ŠALI, A.; OVERINGTON, J. P. Derivation of rules for comparative protein modeling from a database of protein structure alignments. *Protein Science*, Wiley Online Library, v. 3, n. 9, p. 1582–1596, 1994. 7

ŠALI, A. et al. Evaluation of comparative protein modeling by modeller. *Proteins: Structure, Function, and Bioinformatics*, Wiley Online Library, v. 23, n. 3, p. 318–326, 1995. 6

SCHWEDE, T. et al. Swiss-model: an automated protein homology-modeling server. *Nucleic acids research*, Oxford Univ Press, v. 31, n. 13, p. 3381–3385, 2003. 6

SHEN, M. *Statistical potential for assessment and prediction of protein structures*. [S.l.]: Wiley-Blackwell, 2006. 2507–2524 p. 8

SMITH, T. F.; WATERMAN, M. S. Identification of common molecular subsequences. *Journal of molecular biology*, Elsevier, v. 147, n. 1, p. 195–197, 1981. 10, 35

TAVARES, H. L. *shoulddsl documentation*. 2012. Disponível em: <http://www.should-dsl.info/>. 15

TOPF, M. et al. Refinement of protein structures by iterative comparative modeling and cryoem density fitting. *Journal of molecular biology*, Elsevier, v. 357, n. 5, p. 1655–1668, 2006. 3

TORVALDS, L.; HAMANO, J. Git: Fast version control system. *URL http://git-scm. com*, 2010. 15

VERLI, H. *Bioinformática: da Biologia à Flexibilidade Moleculares*. 3. ed. Porto Alegre: The name of the publisher, 2014. 2, 3

WEBB, B.; SALI, A. Comparative protein structure modeling using modeller. *Current protocols in bioinformatics*, Wiley Online Library, p. 5–6, 2014. 6, 7

WU, G. et al. Convergent evolution of trichomonas vaginalis lactate dehydrogenase from malate dehydrogenase. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 96, n. 11, p. 6285–6290, 1999. 37

XU, D. et al. Aida: ab initio domain assembly server. *Nucleic acids research*, Oxford Univ Press, v. 42, n. W1, p. W308–W313, 2014. 4

YANG, G. et al. Pboost: A gpu based tool for parallel permutation tests in genome-wide association studies. *Bioinformatics*, Oxford Univ Press, p. btu840, 2014. 1

ZHANG, Y. Interplay of i-tasser and quark for template-based and ab initio protein structure prediction in casp10. *Proteins: Structure, Function, and Bioinformatics*, Wiley Online Library, v. 82, n. S2, p. 175–187, 2014. 4