**Homology modelling:**

Due to unavailability of solved X-ray crystallographic structure of Leishmanial DHFR-TS, homology modelling was performed for the generation of protein structure of DHFR-TS. Homology modelling was carried out using Modeller 9.19 for Windows (64-bit) obtained from (<https://salilab.org/modeller/>), with the help of ‘Advanced Modeling’ tutorial of Modeller 9.19 which consisted of modeling of protein-ligand complex based on multiple alignment, loop refinement. Three types of input files are required by the Modeller in order to generate a protein structure viz., the query sequence in PIR format, structure of template proteins in PDB format, python command files (scripts) in plain text format. Firstly, the query/target sequenceof*Leishmaniadonovani*DHFR-TS (Accession: XP\_003858395.1)was obtained in the FASTA format from NCBI (**National Center for Biotechnology Information**) (<https://www.ncbi.nlm.nih.gov/>).

This query sequence is utilized for two purposes: a) to run BLAST in order to identify appropriate template proteins on the basis of sequence identity and E-value (expected value), b) FASTA format of query sequence was converted to PIR format and saved as ‘DHFR-TS.ali’ file readable by modeller. 3INVA and 2H2QA belonging to *Trypanosomacruzi*Dihydrofolate reductase-thymidylate synthase proteins having sequence identity of 67% and zero E-value (expected value) each were selected as templates. The solved X-ray crystal structures of both the template proteins were obtained from protein data bank (<https://www.rcsb.org/>). A number of python based scripts such as salign.py, align2d\_mult.py, model\_mult.py, evaluate\_model.py were executed using modeller in order to generate the protein structure. Each script has its own significance from sequence alignment to generation of probable protein models to evaluation of generated protein models. The model\_mult.log file enlists all the models generated **(Table S1).**

Each model was evaluated on the basis of their DOPE (Discrete Optimized Protein Energy) and GA341 Scores, where a model is considered ideal when it possesses least DOPE Score and GA341 value as 1. Therefore, on the basis of DOPE and GA341, DHFR-TS.B99990005.pdb was selected for further evaluation by running evaluate\_model.py script. As a result, a dope profile for the protein model was generated as ‘model.profile’. Gnuplot was utilized to compare the dope score profiles of the templates and model **(Figure S1)**.

The plot showed that the conformation of the loop around residues 200 to 217 had higher DOPE score as compared to the model based on single templates. Therefore, the selected model was subjected to loop refinement function of modeller. Various scripts were executed such as loop\_refine.py, model\_energies\_py, evaluate\_model\_1.py. As a result, a total of ten probable loop models were generated by modeller **(Table S2)**. The best model (DHFR-TS.BL00050001.pdb) was selected on the basis of DOPE score.

The evaluate\_model.py generated a dope score profile for the selected model which was utilized to compare profile of previously selected best protein model (DHFR-TS.B99990005.pdb) as shown in **Figure S2**. The DOPE score for residues around 200-217 was successfully decreased.

The binding site of the modeled protein was located with the help of Discovery studio. The validation of the generated model was done with the help of Ramachandran plot, and various web based programs such as Pro-SA (Protein structure analysis) and ProQ (Protein quality predictor).

Ramachandran plot predicted that more than 99% of the non-glycine residues were in the allowed region and additionally allowed region. Only three of the non-glycine residues were in disallowed region, but all of these were far from binding site therefore had less effect on the structure quality **(Figure S3)**. The ProSA analysis of *L. donovani*DHFR-TS predicted the Z score to be -10.55 is within the range of native conformation of crystal structure **(Figure S4, a)**. The ProSA analysis also showed the overall residue energies of *L. donovani* DHFR-TS model **(Figure S4, b)**. The ProQ web analysis also predicted the model to be extremely good with predicted LG Score 5.475 and predicted Max sub 0.420. The final refinement of the model was done with the help of molecular dynamics simulations which were carried out for a period of 50ns. The RMSD plot of the modelled protein is given in **Figure 5**.The homology modeled DHFR-TS protein was ready to be employed for further application in various CADD **(Computer-aided drug design**) approaches **(Figure 6)**. Subsequently, the X-ray crystallographic protein structure for PTR1 (1E7W) **(Figure S7)** was selected on the basis of resolution.

**Query sequence:**

*Leishmania donovani* DHFR-TS protein sequence (Accession: XP\_003858395.1)

>XP\_003858395.1 dihydrofolate reductase-thymidylate synthase [Leishmaniadonovani]

MSRAAARFKIPMPATKADFAFPSLRAFSIVVALDKQHGIGDGESIPWRVPEDMAFFKDQTTLLRNKKPPT

EKKRNAVVMGRKTWESVPVKFRPLKGRLNIVLSSKATVEELLAPLPEGKRAAAAQDVVVVNDGLAEALRL

LARPPYCSSIETAYCVGGAQVYADAMLSPCVEKLQEVYLTRIYTTAPACTRFFPFPPENTTTAWDLASSQ

GRRKSEADGLEFEICKYVPRNHEERQYLELIDRIMKTGIVKEDRTGVGTISLFGAQMRFSLRDNRLPLLT

TKRVFWRGVCEELLWFLRGETNAQLLADKDIHIWDGNGSREFLDSRGLTENKEMDLGPVYGFQWRHFGAD

YKGFEANYDGEGVDQIRSIVETIKANPNDRRLLFTAWNPCALQKMALPPCHLLAQFYVNTDTSELSCMLY

QRSCDMGLGVPFNIASYALLTILIAKATGLRPGELVHTLGDAHVYRNHVGALKSQLERVPHAFPTLVFKE

ERQFLEDYELTDMEVIDYVPHPPIKMEMAV

**PYTHON SCRIPTS**

**salign.py**

from modeller import \*

log.verbose()

env = environ()

env.io.atom\_files\_directory = './:../atom\_files/'

aln = alignment(env)

for (code, chain) in (('3inv','A'),('2h2q','A')):

 mdl = model(env, file=code, model\_segment=('FIRST:'+chain, 'LAST:'+chain))

aln.append\_model(mdl, atom\_files=code, align\_codes=code+chain)

for(weights, write\_fit, whole) in (((1., 0., 0., 0., 1., 0.), False, True),

 ((1., 0.5, 1., 1., 1., 0.), False, True),

 ((1., 1., 1., 1., 1., 0.), True, False)):

aln.salign(rms\_cutoff=3.5, normalize\_pp\_scores=False,

rr\_file='$(LIB)/as1.sim.mat', overhang=30,

 gap\_penalties\_1d=(-450, -50),

 gap\_penalties\_3d=(0, 3), gap\_gap\_score=0, gap\_residue\_score=0,

dendrogram\_file='3inv\_2h2q.tree',

alignment\_type='tree', # If 'progresive', the tree is not

 # computed and all structues will be

 # aligned sequentially to the first

feature\_weights=weights, # For a multiple sequence alignment only

 # the first feature needs to be non-zero

improve\_alignment=True, fit=True, write\_fit=write\_fit,

write\_whole\_pdb=whole, output='ALIGNMENT QUALITY')

aln.write(file='3inv\_2h2q.pap', alignment\_format='PAP')

aln.write(file='3inv\_2h2q.ali', alignment\_format='PIR')

aln.salign(rms\_cutoff=1.0, normalize\_pp\_scores=False,

rr\_file='$(LIB)/as1.sim.mat', overhang=30,

 gap\_penalties\_1d=(-450, -50), gap\_penalties\_3d=(0, 3),

gap\_gap\_score=0, gap\_residue\_score=0, dendrogram\_file='1is3A.tree',

alignment\_type='progressive', feature\_weights=[0]\*6,

improve\_alignment=False, fit=False, write\_fit=True,

write\_whole\_pdb=False, output='QUALITY')

 **align2d\_mult.py**

from modeller import \*

log.verbose()

env = environ()

env.libs.topology.read(file='$(LIB)/top\_heav.lib')

# Read aligned structure(s):

aln = alignment(env)

aln.append(file='3inv\_2h2q.ali', align\_codes='all')

aln\_block = len(aln)

# Read aligned sequence(s):

aln.append(file='DHFR-TS.ali', align\_codes='DHFR-TS')

# Structure sensitive variable gap penalty sequence-sequence alignment:

aln.salign(output='', max\_gap\_length=20,

gap\_function=True, # to use structure-dependent gap penalty

alignment\_type='PAIRWISE', align\_block=aln\_block,

feature\_weights=(1., 0., 0., 0., 0., 0.), overhang=0,

 gap\_penalties\_1d=(-450, 0),

 gap\_penalties\_2d=(0.35, 1.2, 0.9, 1.2, 0.6, 8.6, 1.2, 0., 0.),

similarity\_flag=True)

aln.write(file='DHFR-TS-mult.ali', alignment\_format='PIR')

aln.write(file='DHFR-TS-mult.pap', alignment\_format='PAP')

**model\_mult.py**

from modeller import \*

from modeller.automodel import \*

env = environ()

a = automodel(env, alnfile='DHFR-TS-mult.ali',

 knowns=('3invA','2h2qA'), sequence='DHFR-TS',

assess\_methods=(assess.DOPE,assess.GA341))

a.starting\_model = 1

a.ending\_model = 5

a.make()

 **evaluate\_model.py**

from modeller import \*

from modeller.scripts import complete\_pdb

log.verbose() # request verbose output

env = environ()

env.libs.topology.read(file='$(LIB)/top\_heav.lib') # read topology

env.libs.parameters.read(file='$(LIB)/par.lib') # read parameters

# read model file

mdl = complete\_pdb(env, 'DHFR-TS.B99990005.pdb')

# Assess all atoms with DOPE:

s = selection(mdl)

s.assess\_dope(output='ENERGY\_PROFILE NO\_REPORT', file='DHFR-TS3.profile',

normalize\_profile=True, smoothing\_window=15)

**loop\_refine.py**

# 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

defselect\_loop\_atoms(self):

 # 10 residue insertion

 return selection(self.residue\_range('200', '217'))

m = MyLoop(env,

inimodel='DHFR-TS-mult.pdb', # initial model of the target

 sequence='DHFR-TS') # 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()

**model\_energies\_py**

from modeller import \*

from modeller.scripts import complete\_pdb

log.verbose() # request verbose output

env = environ()

env.libs.topology.read(file='$(LIB)/top\_heav.lib') # read topology

env.libs.parameters.read(file='$(LIB)/par.lib') # read parameters

for i in range(1, 11):

 # read model file

 code = "DHFR-TS.BL%04d0001.pdb" % i

 mdl = complete\_pdb(env, code)

 s = selection(mdl)

s.assess\_dope(output='ENERGY\_PROFILE NO\_REPORT', file='DHFR-TS4.profile',

normalize\_profile=True, smoothing\_window=15)

 **evaluate\_model\_1.py**

from modeller import \*

from modeller.scripts import complete\_pdb

log.verbose() # request verbose output

env = environ()

env.libs.topology.read(file='$(LIB)/top\_heav.lib') # read topology

env.libs.parameters.read(file='$(LIB)/par.lib') # read parameters

# directories for input atom files

env.io.atom\_files\_directory = './:../atom\_files'

# read model file

mdl = complete\_pdb(env, 'DHFR-TS.BL00050001.pdb')

s = selection(mdl)

s.assess\_dope(output='ENERGY\_PROFILE NO\_REPORT', file='DHFR-TS5.profile',

normalize\_profile=True, smoothing\_window=15)

**2-(4-((4-nitrobenzyl)oxy)phenyl)-1H-benzo[d]imidazole (C5)** creamish white solid, yield: 65% mp: 190-192℃, Rf0.68, IR (ν cm-1) 1693 (C=N), 1248 (-C-O), 2981 (=C-H), 1516, 1364 (-NO2); 1H NMR (400 MHz, CDCl3) δ (ppm): 8.26-8.24 (2H, d, J=8Hz), 8.14-8.12 (2H, d, J=8Hz), 7.75-7.73 (2H, d, J= 8Hz), 7.55-7.53 (2H, m), 7.22-7.15 (4H,  m); 13C NMR (100 MHz, CDCl3) 159.18 (C-13), 147.01 (C-21), 144.67 (C-18), 130.58 (11,15), 128.03 (C-19,23), 128.93 (C-20,22), 129.58 (C-10), 123.22 (C-1,2), 115.18 (C-3,6), 114.6 (C-12,14,) 68.12(C-17)

**2-(4-((2,4-dichlorobenzyl)oxy)phenyl)-1H-benzo[d]oxazole (C8)** Light brownish solid, yield: 65%mp: 105-107℃, Rf0.70, IR (ν cm-1) 1633 (C=N), 1248 (-C-O), 916 (-C-Cl), 3070 (=C-H) 1H NMR (400 MHz, CDCl3) δ (ppm): 7.87-7.85 (2H, d, J=8Hz), 7.48-7.47 (1H, m), 7.30-7.28 (2H, m), 7.09-7.07 (2H, d, J= 8Hz), 6.85-6.80 (2Hz, m), 6.77-6.74 (2H, dd, J= 40.6, 7.68Hz), 6.72-6.68 (1H, m), 5.21 (2H, s); 13C NMR (100 MHz, CDCl3) 163.14 (C-8), 133.52 (C-23), 132.07 (C-18), 129.80 (C-19), 129.31 (C-20), 127.46 (C-1), 127.31 (C-2), 118.53 (C-6), 115.48 (C-11,15), 115.11 (C-12,14), 112.34 (C-3), 67.14 (C-17)

**SPECTRAL DATA COMPOUND 5 (1H NMR)**



**SPECTRAL DATA COMPOUND 5 13C NMR**



**SPECTRAL DATA COMPOUND 8 (1HNMR)**

**SPECTRAL DATA COMPOUND 8 (13CNMR)**



**Table S1: List of generated protein models via model\_mult.py script.**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **DHFR-TS Model** | **DOPE Score** | **GA341** |
|  | DHFR-TS.B99990001.pdb | -58972.06641 | 1.00000 |
|  | DHFR-TS.B99990002.pdb | -59282.17969 | 1.00000 |
|  | DHFR-TS.B99990003.pdb | -59327.55859 | 1.00000 |
|  | DHFR-TS.B99990004.pdb | -59120.92578 | 1.00000 |
|  | DHFR-TS.B99990005.pdb | -59355.75391 | 1.00000 |

**Table S2: Summary of successfully generated loop models.**

|  |  |  |
| --- | --- | --- |
| **S.No.**  | **DHFR-TS Model** | **DOPE Score** |
|  | DHFR-TS.BL00010001.pdb | -57552.054688 |
|  | DHFR-TS.BL00020001.pdb | -57439.125000 |
|  | DHFR-TS.BL00030001.pdb | -58450.937500 |
|  | DHFR-TS.BL00040001.pdb | -58514.281250 |
|  | DHFR-TS.BL00050001.pdb | -58533.960938 |
|  | DHFR-TS.BL00060001.pdb | -58320.531250 |
|  | DHFR-TS.BL00070001.pdb | -57791.804688 |
|  | DHFR-TS.BL00080001.pdb | -58450.85156 |
|  | DHFR-TS.BL00090001.pdb | -58181.117188 |
|  | DHFR-TS.BL00010001.pdb | -58327.539063 |



**Figure S1: Dope score profile for protein model DHFR-TS.B99990005.pdb.**



**Figure S2: Dope score profile for protein model after loop refinement.**



**Figure S3: The ramachandran plot of modeled *L.donovani* DHFR-TS.**

****

**Figure S4: Z-score (a) and energy profile diagrams (b) of homology modeled DHFR-TS obtained from ProSA.**

****

**Figure S5: The RMSD plot of modeled *L.donovani* DHFR-TS protein.**



**Figure S6: The *L.donovani* DHFR-TS homology modeled structure.**



**Figure S7: Crystal structure of PTR1 (1E7W)**