Quick Guide to CNFpred

Jianzhu Ma, Sheng Wang and Jinbo Xu
April 9, 2013

CNFpred (ver 1.62) is a protein threading program and also a key component of the RaptorX server (http://raptorx.uchicago.edu) developed by Xu group, excelling at the alignment of distantly-related proteins with sparse sequence profile. It is freely available at http://raptorx.uchicago.edu/download/. Currently CNFpred can run on a X86_64 Linux system (optionally with OpenMPI installed). This package is provided **AS IS** without any warranty. You can use it for any purpose (e.g., CASP) except selling it, but please do not incorporate it into a protein structure prediction web server **WITHOUT** explicit permission from the authors since we are operating a structure prediction server based upon this program.

1. Overview of the package

The whole *CNFpred* package includes the following files:

1) CNFsearch1.62.release.tar.gz (~100M): the package contains all the executable programs.

2) TPL_BC40.tar.gz (~900M): a template database in which any two proteins share < 40% sequence identity.

3) TPL_Remain.tar.gz (~1G): this package plus TPL_BC40.tar.gz forms a complete template database in which any two proteins share up to 99% sequence identity. The template databases will be updated monthly.

4) nr70.tar.gz (~1.6G) and nr90.tar.gz (~2.3G): the NR70 and NR90 protein sequence databases, already formatted using formatdb in the BLAST package.

5) TemplateLists.tar.gz (~1M): the package contains some template lists.

You shall download the below PDB packages if you want to build 3D models from the CNFpred alignments using MODELLER. Please use our reformatted and improved PDB files to avoid inconsistency between the alignment files and the PDB files in the Protein Data Bank.

6) pdb_BC40.tar.gz (~700M): the reformatted PDB files for all the templates in TPL_BC40.tar.gz

7) pdb_Remain.tar.gz (~900M): the reformatted PDB files for all the templates in TPL_Remain.tar.gz

Note: whenever you download an updated version of our template database, you shall download TPL_BC40.tar.gz, TPL_Remain.tar.gz, TemplateLists.tar.gz and optionally pdb_BC40.tar.gz and pdb_Remain.tar.gz.

The CNFsearch1.62.release.tar.gz package consists of the following four main programs:

buildFeature:	It generate a feature file for a given target protein sequence.
CNFalign:	It aligns one target protein sequence to one template protein. There are three different implementations: *CNFalign_lite*, *CNFalign_fast* and *CNFalign_normal*. All the implementations have the same user interface. This package currently only contains *CNFalign_lite*.
CNFsearch:	It searches similar templates in a database for a query protein sequence. This program automatically calls *CNFalign_lite*, *CNFalign_fast* or *CNFalign_normal*, depending on availability and CPU types. *CNFalign_lite* is 5-6 times faster than *CNFalign_fast*, which in turn is 30% faster than *CNFalign_normal*. However, *CNFalign_lite* is 2% worse than the other two in terms of accuracy. For example, on all the CASP10 targets, *CNFalign_lite* can obtain an accumulative TMscore of 74.1 while *CNFalign_fast* and *CNFalign_normal* can obtain 75.5. There are two variants of this program: CNFsearch and CNFsearch_mpi that generate exactly the same results. Meanwhile, CNFsearch is a multi-thread program that can run on multiple CPUs of a single compute node and CNFsearch_mpi is a MPI-based program that can run on a Linux cluster of multiple compute nodes. As such, CNFsearch_mpi can make use of many more CPUs (if available) than CNFsearch. Roughly speaking, you can run CNFsearch on any 64-bit Linux system, but have to install OpenMPI (http://www.open-mpi.org/) to run CNFsearch_mpi.
build3Dmodel:	It builds a 3D model for the target protein sequence based on its alignment to one template structure.

Note: You can see the usage of one program by running it without arguments.

2. Search through a template database using *CNFsearch*

2.1 Configuration

Uncompress CNFsearch1.62.release.tar.gz by "tar –xzvf CNFsearch1.6.release.tar.gz" and you shall see a folder CNFsearch1.62/. Change the current work directory to "CNFsearch1.62/", you shall see the following files and folders:

1) executable files: *buildFeature*, *CNFalign_lite*, *CNFsearch_mpi*, *build3Dmodel* and *setup.pl*

2) folders: databases/, TGT/, util/ and tmp/

Note: each user has to install one copy of the *CNFsearch* package in his/her own account in order to run *CNFsearch* correctly, although the databases can be shared among multiple users.

In "CNFsearch1.62/", run the following command to do an initial setup.

```
./setup.pl
```

Note: whenever you upgrade the *CNFsearch* package or move *CNFsearch* to another destination, you shall run this setup script to make sure all the paths are configured correctly.

2.2 NR Database

If the directory "databases/NR_new/" does not exist, create one. Move the two NR packages (nr70.tar.gz and nr90.tar.gz) to "databases/NR_new/" and then uncompress them using the command "tar".

2.3 Template Databases

If the directory "databases/TPL_BC100/" does not exist, create one. Uncompress TPL_BC40.tar.gz and you shall see one folder "TPL_BC40/". Uncompress TPL_Remain.tar.gz and you shall see one folder "TPL_Remain/". Move all the .tpl files in "TPL_BC40/" and "TPL_Remain/" to "databases/TPL_BC100" and then delete two empty folders "TPL_BC40/" and "TPL_Remain/". In "databases/" you shall see some template list files, which are also included in the package TemplateLists.tar.gz. You can create your own template lists and save them in "databases/".

2.4 PDB files [optional]

If "databases/pdb_BC100" does not exist, create one. Uncompress pdb_BC40.tar.gz and you shall see one folder "pdb_BC40/". Uncompress pdb_Remain.tar.gz and you shall see one folder "pdb_Remain/". Move all the .pdb files in "pdb_BC40/" and "pdb_Remain/" to "databases/pdb_BC100" and then delete the two empty folders "pdb_BC40/" and "pdb_Remain/".

2.5 Build a feature file for a query protein sequence

To run CNFalign or CNFsearch on a query sequence, you shall first build a feature file (.tgt) for this. Such a feature file contains a variety of sequence and structural information for the query protein sequence. Given a query protein sequence file in the FASTA format (e.g. query.seq), you may run the following command to generate a feature file.

```
./buildFeature  -i  query.seq  -o  output_file  -c  cpu_num
```

where cpu_num specifies the number of CPUs to be used by BLAST. If everything works OK you will obtain a protein feature file (If not specified, the default file is query.tgt). With this feature file you can search the template database as follows.

```
./CNFsearch –a NP  -q  query  -l  databases/bc100_tpl_list  -d  databases/TPL_BC100/  -g TGT/ -n  200  -p 0.05  -o  query.rank
```

or

```
mpirun  -np  NP  ./CNFsearch_mpi  -q  query  -l  databases/bc100_tpl_list  -d databases/TPL_BC100/  -g  TGT/ -n  200  -p 0.05  -o  query.rank
```

The options are explained below.

-a (or –np)	NP:	The number of processors to be used by CNFsearch or CNFsearch_mpi.
-q	query:	Query protein name. query.tgt should be already available in the folder specified by tgt_root/.
-l	list:	The list of templates [default = databases/bc40_tpl_list].

-d tpl_root: The folder containing the template files (i.e., .tpl files) [default = databases/TPL_BC100/].

-g tgt_root: The folder containing the target protein feature files (i.e., the .tgt files) [default = TGT/].

-o out_file: The file containing a brief summary of the threading results [default = query.rank in current work directory].

-n topN: Only keep the results for the top topN templates [default = 100].

-f : If specified, *CNFsearch* always uses *CNFalign_lite* to speed up at the slight cost of accuracy.

-p pval: Only keep the results for the templates with p-value less than the given cutoff pval (default 0.05).

Suppose that we want to do threading on T0607.seq and the file TGT/T0607.tgt already exists. To search the BC40 template database using 10 CPUs and also to keep the templates with p-value <0.05, run the following command:

```
./CNFsearch –a 10 –q T0607 –p 0.05
```

or

```
mpirun -np 10 ./CNFsearch_mpi -q T0607 -p 0.05
```

To search the BC70 template database using 10 CPUs, keep the top 200 templates or those with p-value <0.001, run the following command:

```
./CNFsearch –a 10 -q T0607 -l databases/bc70_tpl_list -n 200 -p 0.001
```

or

```
mpirun -np 10 ./CNFsearch_mpi -q T0607 -l databases/bc70_tpl_list -n 200 -p 0.001
```

The ranking result and the alignments to the top templates will be saved in T0607.rank. The alignment of the query protein to each individual top template is saved in "tmp/T0607/FASTA/". In the next section we will further explain the ranking file.

2.6 CNFsearch ranking file

The ranking file summarizes the result of searching through the template database. As shown in Fig. 1, the first line contains the query protein name. The second line shows the query protein sequence. The third line is the query protein length. The NEFF (number of effective families) in the fourth line is the average Shannon "Sequence Entropy" for a PSI-BLAST sequence profile. NEFF is the average number of amino acid (AA) substitutions across all residues of a protein, ranging from 1 to 20 (i.e., the number of AA types). NEFF at one residue is calculated by $\exp(-\sum_k p_k \ln p_k)$ where p_k is the probability for the k^{th} AA type), and NEFF for the whole protein is the average across all residues. Generally speaking, NEFF is used to quantify the homologous information content available for a given protein. The larger the NEFF value, the more homologous information its profile contains. More detail information between NEFF and homologous information can be found in the paper (Peng and Xu, ISMB 2010). The fifth line contains the number of templates searched by CNFsearch.

```
Query Name = T0607

Query Sequence = MFKPQGLYDYICQQWQEEILPSLCDYIKIPNKSPHFDAKWEEHGYMEQAVNHIANWCKSHAPKGMTLEIVRLKNRTPLLFMEIPGQIDDTVLLYGHLDKQPEMSGWSD
DLHPWKPVLKNGLLYGRGGADDGYSAYASLTAIRALEQQGLPYPRCILIIEACEESGSYDLPFYIELLKERIGKPSLVICLDSGAGNYEQLWMTTSLRGNLVGKLTVELINEGVHSGSASGIVAD
SFRVARQLISRIEDENTGEIKLPQLYCDIPDERIKQAKQCAEILGEQVYSEFPWIDSAKPVIQDKQQLILNRTWRPALTVTGADGFPAIADAGNVMRPVTSLKLSMRLPPLVDPEAASVAMEKAL
TQNPPYNAKVDFKIQNGGSKGWNAPLLSDWLAKAASEASMTYYDKPAAYMGEGGTIPFMSMLGEQFPKAQFMITGVLGPHSNAHGPNEFLHLDMVKKLTSCVSYVLYSFSQKK

Query Length = 471

Query NEFF = 10.7

Searched Templates = 22031
-----------------------------------------------------------

Date Tue Feb 12 13:50:45 2013
```

No	Template	Pvalue	Score	qRange	tRange	tLength	Cols	#tGaps	#qGaps	#seqID
1	3pfeA	2.898e-19	496.8	2-470	3-471	472	469	0	0	469
2	3dljA	3.933e-15	390.7	3-469	12-483	485	452	15	20	96
3	2pokA	1.472e-14	376	3-468	30-481	481	448	18	4	78
4	1cg2A	3.25e-12	315.9	3-470	6-392	393	381	87	6	61
5	3pfoA	6.48e-12	308.2	3-471	13-431	433	391	78	28	56
6	1vgyA	9.325e-12	304.1	16-468	5-378	393	372	81	2	69
7	1lfwA	3.559e-11	289.2	5-471	3-468	470	410	57	56	63
8	3rzaA	5.66e-11	284	10-470	20-394	396	362	99	13	58

Figure. 1. An example of the ranking file generated by *CNFsearch*.

The columns are explained as follows.

Column 1	'**No**'	Ranking of the template
Column 2	'**Template**'	Name of the template protein (PDB ID or SCOP protein name)
Column 3	'**P-value**'	The P-value of the alignment. The smaller, the better.
Column 4	'**Score**'	The alignment raw score between the template protein and query protein.
Column 5	'**qRange**'	Range of aligned region on the query protein.
Column 6	'**tRange**'	Range of aligned region on the template protein.
Column 7	'**tLength**'	The template protein length.
Column 8	'**Cols**'	The number of aligned positions in the pairwise alignment.
Column 9	'**#tGaps**'	The number of gaps in the template protein (Insertions).
Column 10	'**#qGaps**'	The number of gaps in the query protein (Deletions).
Column 11	'**#seqID**'	The number of identical residues in the alignment.

2.7 Reliability of P-value

In the ranking file, P-value can be understood as a confidence score indicating the relative quality of the top-ranked templates and (corresponding) alignments. To calculate the P-value, we employs a set of reference templates (in "databases/CAL_TPL)"), which consists of ~1800 single-domain templates with different SCOP folds. Given a query protein, we first thread it to this reference template database and then estimate an extreme value distribution from the ~1800 alignment scores (i.e., alignment potentials). Based upon this distribution, we calculate the P-value of each alignment when threading the query protein to the real template database. The P-value actually measures the quality of the template (and the alignment) by comparing it to the reference templates.

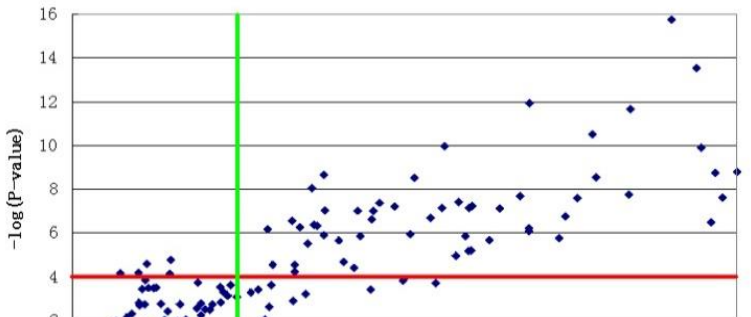

Figure. 2. The relationship between P-value and the model quality on the 123 CASP10 targets. The x-axis is the model quality measured by $max(GDT, uGDT)$ and the y-axis is *–log(P-value)*.

To study the relationship between the P-value and the model quality we conduct experiments on the 123 CASP10 target proteins. We use both GDT and uGDT (i.e., un-normalized GDT) to measure the real model quality. GDT has been employed as an official measure by CASP for many years. It measures the quality of a model by comparing it with the native and outputs a value from 0 to 100, indicating the worst and the best quality, respectively. uGDT is equal to GDT times the target length divided by 100. uGDT is more suitable when the target protein is relatively large or multi-domain and there are only good templates covering a segment of the target (e.g., one of the domains). We say one alignment is reasonable when its resultant model has uGDT or GDT larger than 50. We use 50 as a cutoff because that many proteins similar at only the fold level have GDT or uGDT around 50. We say one predicted model is reasonable if $max(uGDT, GDT)$ is larger than 50 because the alignment of many proteins similar at only the fold level has GDT or uGDT around 50. Fig. 2 shows the relationship between P-value and $max(uGDT, GDT)$ on the 123 CASP10 targets. When P-value is small (i.e. $<10^{-5}$), the models have uGDT or GDT greater than or equal to 50. Even if P-value is less than 10^{-4}, there are very few models with both uGDT and GDT less than 50. That is, the template is a reasonable one when its corresponding P-value is less than 10^{-4}.

2.8 Interpreting a pairwise alignment

```
No 1
>3pfeA

T sse_real            CCHHHHHHHHHHHHHHCHHHHHHHHHCCCCCCCCCCCCCHHHHCHHHHHHHHHHHHHHHCCCCCCCEEEEECCCCCCCCEEE
T acc_real            MEMEMBMEMBMMBBEEMBBEMBMEBBEBMBMBBEMMEEMEEMMMBMEBBMMBBEBBEEEBMBEEMEMEMBEMEEMBBBBB
T 3pfeA             3 FKPQGLYDYICQQWQEEILPSLCDYIKIPNKSPHFDAKWEEHGYMEQAVNHIANWCKSHAPKGMTLEIVRLKNRTPLLF   81 (472)
                      ++++++|||||+++|+++|||||+||||||+++++++++++++***~*+++|||+|||||||+**~*+++|||+*~*+||||
S T0607             2 FKPQGLYDYICQQWQEEILPSLCDYIKIPNKSPHFDAKWEEHGYMEQAVNHIANWCKSHAPKGMTLEIVRLKNRTPLLF   80 (471)
S acc_pred            EEEEMBMEBBMEEMMEEBBEMBMEBBEBMMEMEEEEEEMEEMEMBMEBBEMBMEMBEEMMEEEBEBMMBEMEEEBBBBB
S acc_conf            56754966596764677876695578674454585667465575656986695659875369656645456864488 98
S sse_pred            CCHHHHHHHHHHHHHHHHHHHHHHHHHCCCCCCCCCCCCCHHHHHHHHHHHHHHHHHHHHHCCCCCCCEEEEECCCCCCCEEE
S sse_conf            99678999999996765999999877677999999887567789999999999999985999996899978999969 99
```

Figure. 3. An example alignment in the ranking file.

The last section in the ranking file contains the alignments of the query protein to the top templates. Fig.3 shows one example alignment between the template and the query protein. The alignment consists of one or more blocks with the following lines:

Row 1: 'T sse_real' The real 3-class secondary structure determined by DSSP.

Row 2: 'T acc_real'	The real 3-state solvent accessibility determined by DSSP.
Row 3: 'T 3pfeA'	The template protein name.
Row 5: 'S T0607'	The query protein name.
Row 6: 'S acc_pred'	The predicted 3-state solvent accessibility by our in-house software.
Row 7: 'S acc_conf'	The confidence score of the predicted solvent accessibility.
Row 8: 'S sse_pred'	The secondary structure predicted by PSIPRED.
Row 9: 'S sse_conf'	The confidence score of the predicted secondary structure.

The line in the middle shows the alignment score at each aligned positions, which indicates the relative alignment quality. The symbols used here have the following meanings.

'|' : Alignment score above 13.
'+' : Alignment score between 10 and 12.
'*' : Alignment score between 6 and 9.
'~' : Alignment score between 2 and 5.
'.' : Alignment score between -1 and 1.
'-' : Alignment score between -4 and -2.
'=' : Alignment score below -5.

3. Pairwise alignment using *CNFalign*

CNFalign_lite (or CNFalign_fast and CNFalign_normal) is the program to align a protein sequence to a template. Suppose that the feature file of the query protein is already generated in the desired directory. You can run the following command to generate a pairwise alignment between the template and the query protein.

./CNFalign_lite -t template_name -q target_name [-l tpl_root] [-g tgt_root] [-d output_root]

The options are explained below.

-t template_name:	Template protein name. template_name.tpl file should be ready in the folder specified by tpl_root.
-q target_name:	Target protein name. target_name.tgt file should already be available in the folder specified by tgt_root.
-l tpl_root:	The folder containing the template feature files (i.e., the .tpl files) [default = databases/TPL_BC100/].
-g tgt_root:	The folder containing the target feature files (i.e., the .tgt files) [default=TGT/].
-d output_root:	The folder where to save the output file. If not specified, no output will be generated.

The output file is named as template_name-target_name.fasta and template_name-target_name.cnfpred. Meanwhile, the .cnfpred file has the same format as an alignment in the ranking file.

4. Attention for using *CNFsearch* and *CNFalign*

1) When you use a job scheduler to submit the MPI job to a cluster or supercomputer, please do not pass the TMP environment variable to the compute nodes which are scheduled to run this

MPI job. We only tested *CNFsearch* with OpenMPI but not MPICH. If you do not have MPI, you can still run *CNFsearch* on a single CPU, but much more slowly.

2) The maximum allowable query protein sequence length is 2000 residues.

3) If you want to rerun a protein sequence, it is better to clean up its corresponding temporary files in "tmp/" to avoid mixing of old and new results.

4) To speedup, you may search BC40 first to select the top 100 templates. Then you can search all the templates in BC100 which are similar to the top 100 templates. You may use the databases/bc40_map_* file to find all the similar proteins of a given template.

5) If you run a sequence with >1500 residues, please make sure that your machine has enough memory.

6) Enclosed in the package there is an example output file T0607-lite.rank, which can be used to test if you install the package correctly.

5. Build 3D models from an alignment

You can build a 3D model from a CNFsearch alignment using our script. Suppose that you have an alignment file and the template PDB file, you can run the following command to build 3D models.

./build3Dmodel -i align_file -q query_name -d pdb_root [-m mod_bin] [-n mod_num]

The options are explained below.

-i align_file	The input alignment file in the FASTA format.
-q query_name	The name of query protein for which you will build 3D models.
-d pdb_root	The directory containing the PDB file of the template protein (e.g., databases/pdb_BC100/).
-m mod_bin	The *MODELLER* executable file (default = ~/bin/modeller9v8/bin/mod9v8).
-n mod_num	The number of 3D models to be generated from the alignment (default=1).

Before running Build3DModel, please make sure that MODELLER is installed. It is available at http://www.salilab.org/modeller/9v8/modeller-9v8.tar.gz.

6. What are missing in this package?

1) A template re-rank module using context-specific pairwise distance-dependent potential EPAD (available at http://ttic.uchicago.edu/~jinbo/software.htm). This re-ranking algorithm can help improve threading accuracy by about 2.0 TMscore units on all the CASP10 targets compared to CNFalign_fast.

2) A multi-template threading module that can help improve the modeling accuracy by about 1.8 TMscore units on all the CASP10 targets compared to 1). See http://onlinelibrary.wiley.com/doi/10.1002/prot.23016/abstract for technical details.

3) A module for both local and global alignment quality assessment.

All these modules are implemented in RaptorX Server (http://raptorx.uchicago.edu).

7. Coverage vs. quality

Currently the alignment score-based ranking algorithm is biased towards templates with larger coverage on the query sequence. Given a multi-domain query protein sequence S and its two templates T1 and T2, if T1 is a close homolog but can only cover a short domain of S_i, and T2 is a remote homolog but can cover all the domains, then it is very likely that T2 will be ranked before T1. To rank T1 before T2, you may cut S into domains and do search again. In the future release, we will provide one ranking algorithm that can rank short but close templates higher.
For a multi-domain protein sequence, it is better to cut it into domains using tools such as Pfam or CDD. Nevertheless you can also cut a long protein sequence into domains based upon the results generated by CNFpred.

8. Running time

The running time is linear to the query sequence length and template database size. It takes the lite mode <10 minutes to search T0607 (471 AAs) against the BC100 database using 12 cores (Intel Xeon processor E5-2600 family).

9. Questions

For bug reports, questions, or comments, please email to Dr. Jinbo Xu at jinboxu@gmail.com. If the program cannot run on your computing systems, we may recompile it for you.

10. References

Jianzhu Ma, Sheng Wang and Jinbo Xu. Protein Threading Using Context-Specific Alignment Potential. Bioinformatics (Proceedings of ISMB 2013), July 2013

Jianzhu Ma, Jian Peng, Sheng Wang and Jinbo Xu. A Conditional Neural Fields Model for Protein Threading. Bioinformatics (Proceedings of ISMB 2012), July 2012.

Jian Peng and JinboXu. Low-homology Protein Threading. Bioinformatics (Proceedings of ISMB 2010), July 2010.