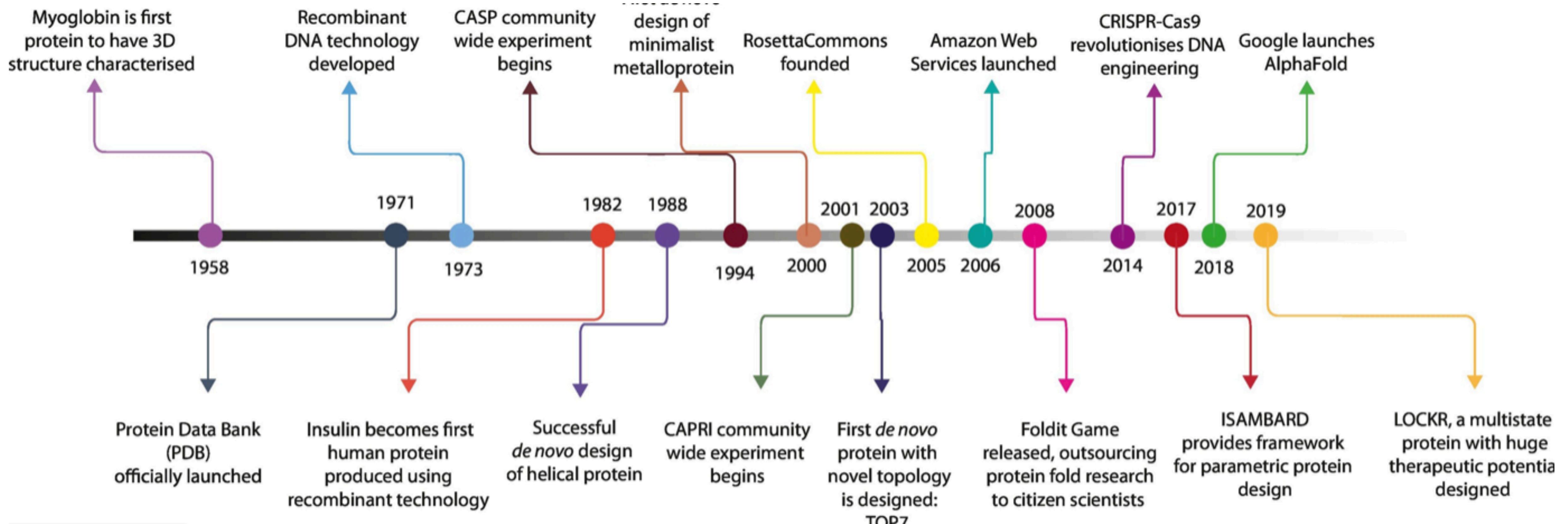


AlphaFold and Protein Structure Prediction

CCATS Group

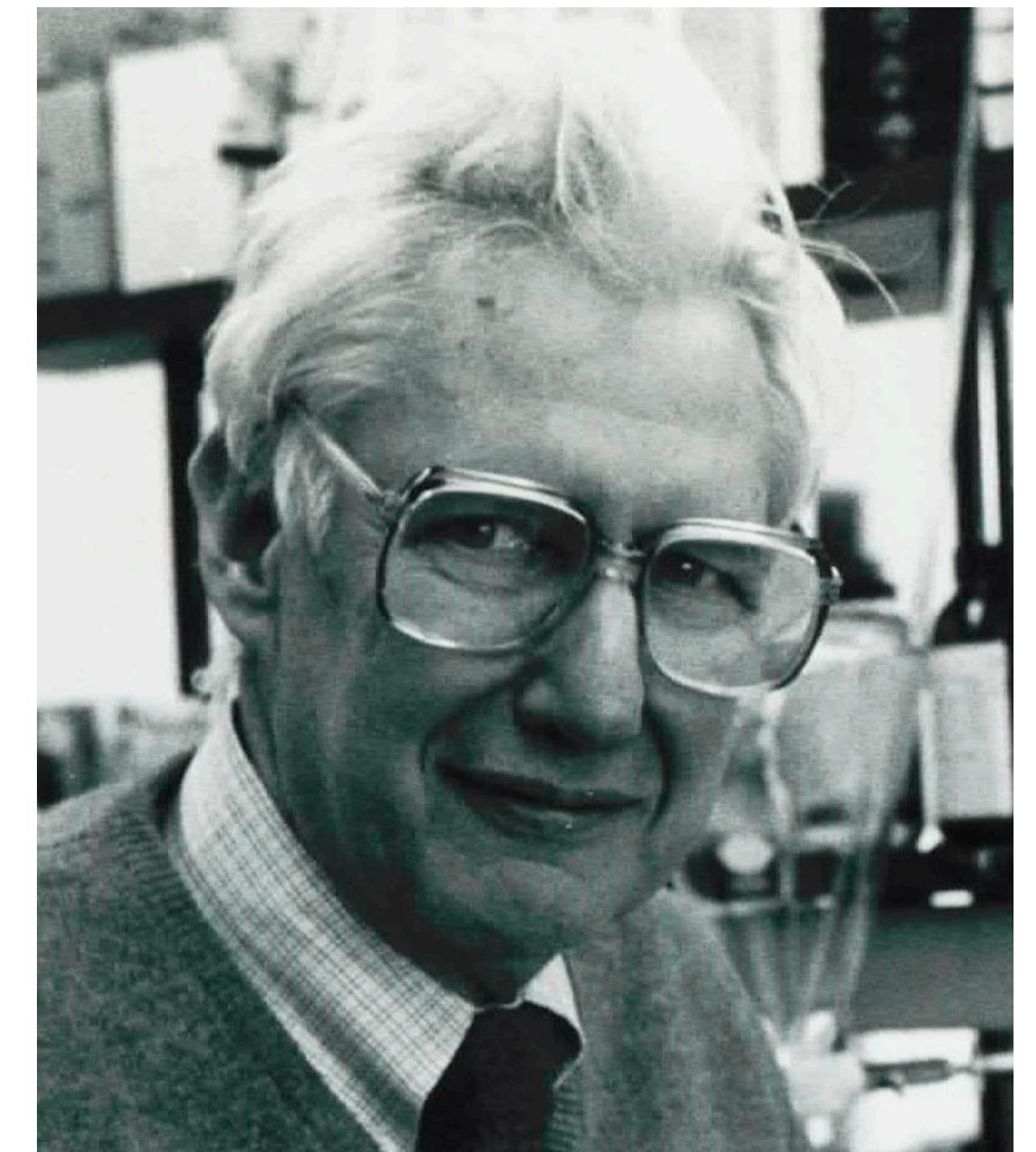


Timeline

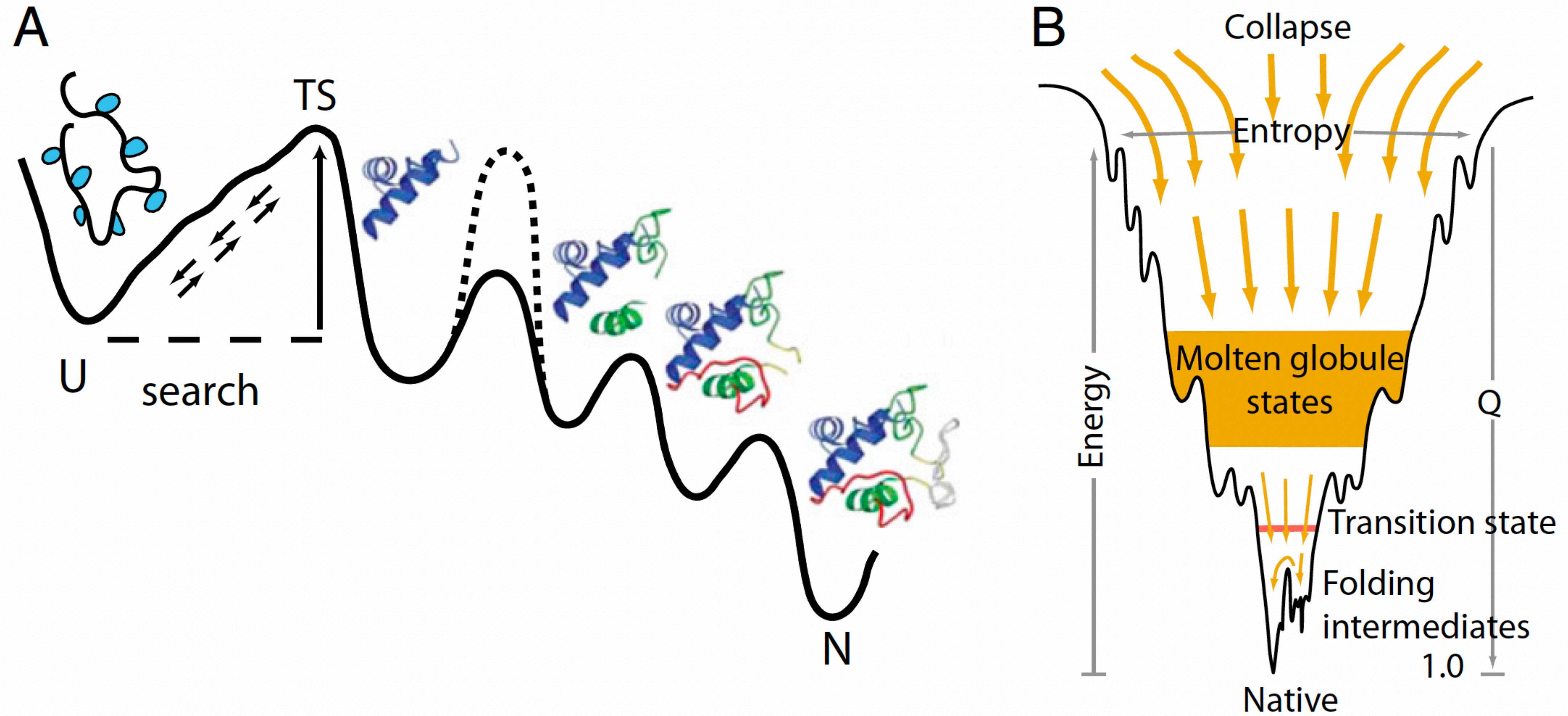


Protein Structure Prediction vs Protein Folding

- It is a much harder task to identify the protein folding pathway
- Cyrus Levinthal's Paradox
 - A protein with 150 amino acids, 298 dihedral angles
 - If each angle can adopt one of three stable conformations, there are $3^{298} \sim 10^{149}$ possible configurations
 - If the protein randomly samples the configurations at a rate of 10^{12} configurations per second, it would take the age of the universe for this protein to fold
 - God does not play dice with the universe



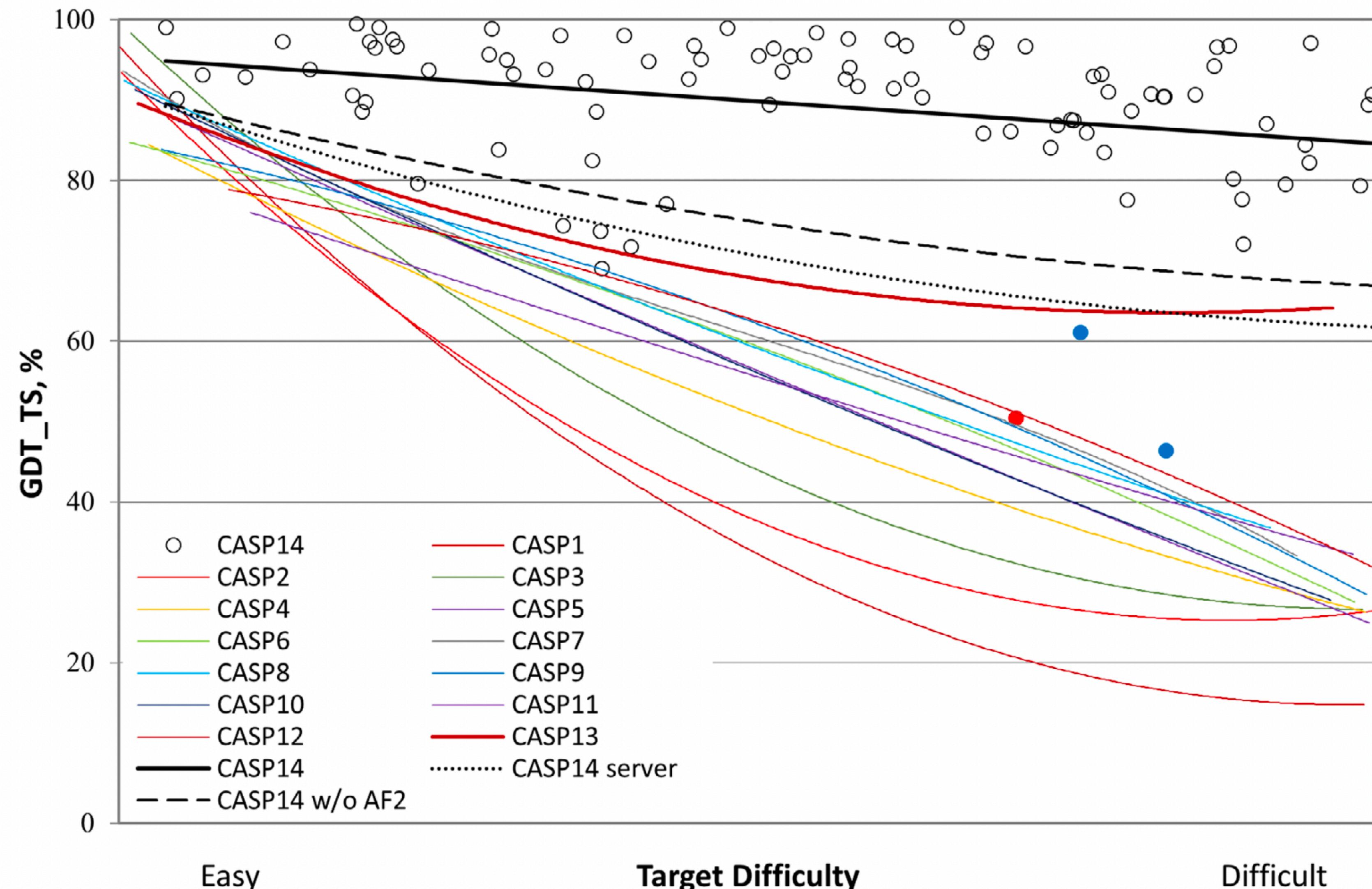
Protein Folding Pathway



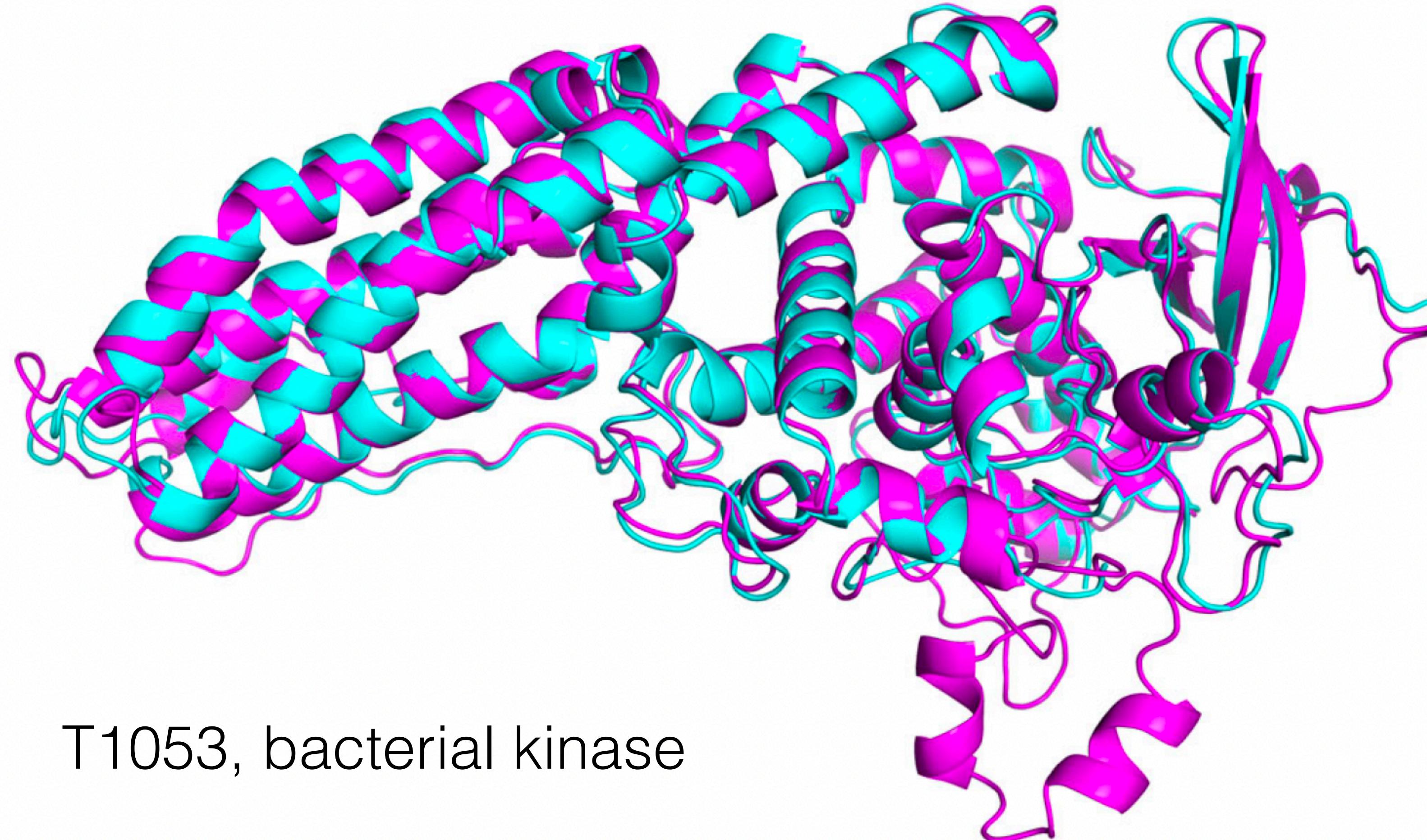
CASP14

- Critical Assessment of Structure Prediction (CASP)
- Test systems: 52 proteins or protein complexes. 42 (x-ray), 7 (cryo-EM), 3 (NMR)
- Participants: 97 research groups (19 countries), 215 modeling methods
- Global Distance Test Total Score (GDT_TS) through Local-Global Alignment
 - Zelma, *Nuc. Acids Res.* 2003, 31, 3370
 - Measures the similarity between structures of the same protein

CASP14



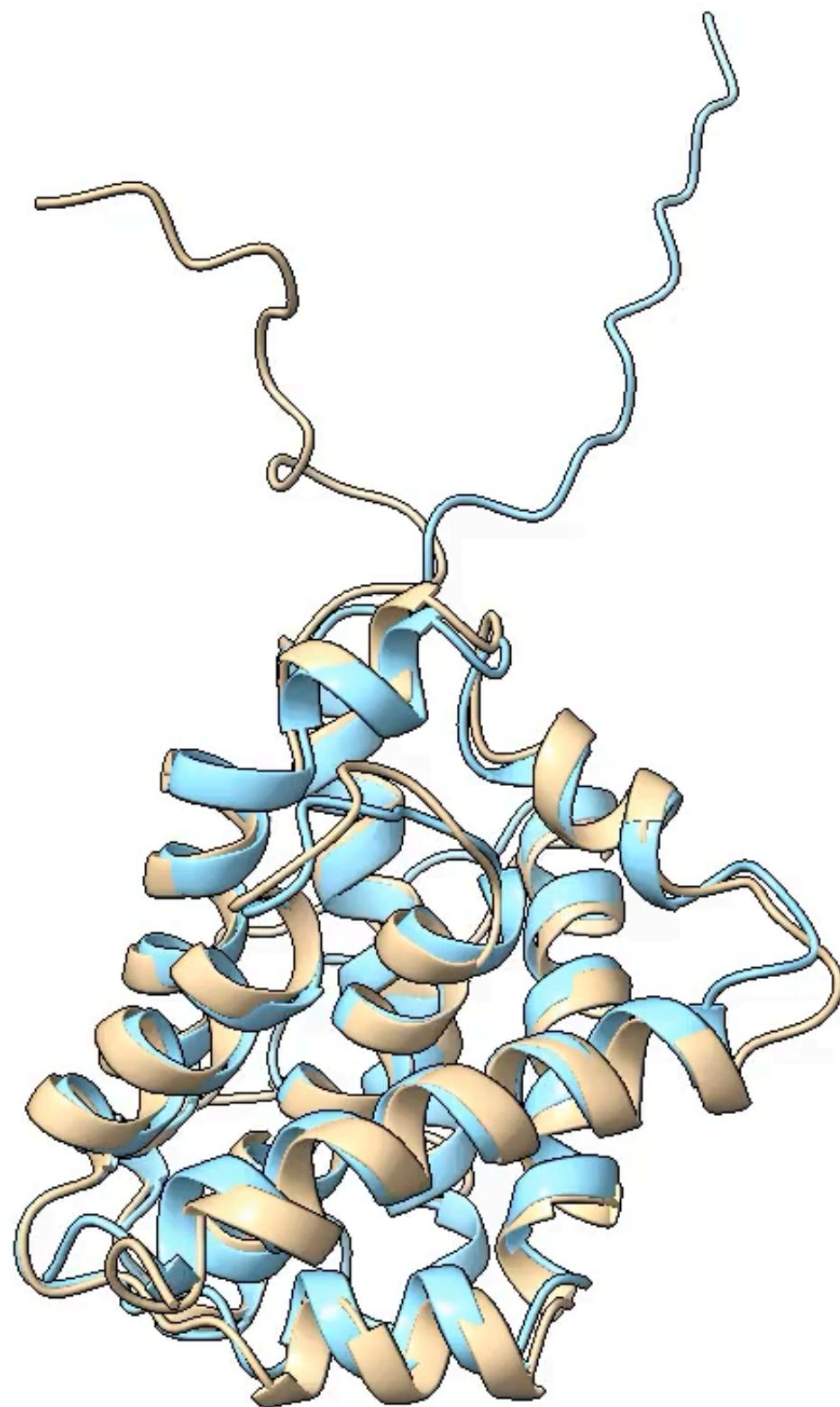
AlphaFold2 Results From CASP14



T1053, bacterial kinase

AlphaFold Tutorial

- **Goals:**
 - A. Predict the structure of Bax
 - B. Predict the complex of PaParDE (Bourne Lab)
- The working directory I prepared is organized as follow:
 - ✓ /home/van/MacroMol_Modelling_Tutorial/02_alphafold/
 - > Results/
 - ✓ Tutorial/
 - ✓ 00_monomer_pred/
 - > monomer.slurm
 - ✓ 01_complex_pred/
 - > complex.slurm



NMR Structure (PDB 1F16)
AlphaFold Prediction

AlphaFold Workflow

1. Input

- Amino acid Sequence (FASTA format)

2. AlphaFold Program

- Conda environment (contains dependencies; ~ 2.2 Gb)
- AlphaFold Software (~14 Mb)

3. AlphaFold Databases (~2.2 Tb)

- BDF, MGnify, PDB70, PDB MMCIF, PDB SEQRES, UniClust30, Uniprot, Uniref90

4. GPU (not required but strongly encouraged)

5. Set a max template date

If all goes well, you get 5 to 25 predicted structures!

FASTA Files

- FASTA means “FAST All,” which is an extension from the previous “FAST-N” or “FAST-P”
 - Format originated from FASTA Program for sequence alignments (1985)
 - Now the universal standard for bioinformatic tools (BLAST, CLUSTAL, etc.)
1. First line contains “>” which denotes a comment
 - Formally, this line also contain descriptions such as:
 - UNIPROT entry, Gene Identity, Organism, Accession Numbers, etc.
 2. Next line contains one letter code of nucleotide or amino acid sequence
 - **AA sequence for Alphafold!**
 - Usually 80 characters/line
 3. Followed by empty line for multiple sequences.

FASTA Files

Example Bax UNIPROT Data:

- <https://www.uniprot.org/uniprotkb/Q07812>
- <https://rest.uniprot.org/uniprotkb/Q07812.fasta>

UniProtKB/Swiss-Prot Entry Name organism name gene name sequence version
Unique Identifier Protein Name organism identifier protein existence

>sp|Q07812|BAX_HUMAN Apoptosis regulator BAX OS=Homo sapiens OX=9606 GN=BAX PE=1 SV=1
MDGSQEPRGGGPTSSSEQIMKTGALLQGFIQDRAGRMGEAPELALDPVPQDASTKKLS
ECLKRIGDELDNSMELQRMIAAVDTDSPREVFFRVAADMFSMDGNFNWGRVVALFYFASKL
VLKALCTKVPELIRTIMGWTDFLRERLLGWIQDQGGWDGLLSYFGTPTWQTVTIFVAGV
LTASLTIWKKG

amino acid sequence
80 characters/line

AlphaFold Program on OSCER

- <https://github.com/deepmind/alphafold>
- Installation: 1) NVIDIA Container Toolkit, and 2) Docker file (*cannot do this as normal user*)
- My workaround:
 1. Conda environment (Contains NVIDIA Toolkit, Tensorflow, OpenMM, HHsuite, etc.)
 - Anaconda/Miniconda for package management (**conda activate af2.2.0**)
 2. Download my own copy of AlphaFold (**/home/van/Programs/alphafold**)
- Database for AlphaFold is stored on OSCER
 - Half of the database is stored in: **/opt/oscer/software/alphafold/2.0.0**
 - Other half requires periodic updates which is automated in: **/data/alphafold** (done at the beginning of every month)

AlphaFold Databases

- BDF
 - Clustered protein sequences (MSA)
- MGnify
 - Genetic microbiome data (MSA)
- PDB70
 - Sequence for HHsuite (MSA)
- PDB MMCIF
 - MMCIF files from RCSB
- PDB SEQRES
 - RCSB data for complex prediction
- Uniclust30
 - Clustered sequences based on 30% pairwise identity
- Uniprot
 - Gene data for complex prediction
- Uniref90
 - Clustered sequences from Uniref100 database (sequences shorter than 11 have 1 representative member here)

Differences for Monomer v. Complex Prediction

1. Input File

(Amino Acid Sequence(s) in FASTA format)

```
$ vi monomer.fasta
```

```
1 > name_1
```

```
2 [amino acid sequence 1]
```

```
$ vi complex.fasta
```

```
1 > name_1
```

```
2 [amino acid sequence 1]
```

```
3 > name_2
```

```
4 [amino acid sequence 2]
```

```
...
```

Example:

Monomer FASTA

```
1 >bax
2 MDGSGEQPRGGGPTSSEQIMKTGALLQGFIQDRAGRMGEAPELA
LDPVPQDASTKKLSECLKRIGDELDNMELQRMIAAVDTDSPREVF
FRVAADMFSDGNFNWGRVVALFYFASKLVLKALCTKVPELIRTIMG
WTLDFLRERLLGWIQDQGGWDGLLSYFGPTWQTVTIFVAGVLTAS
LTIWKKMG
```

Complex FASTA (PaParDE - Bourne Lab)

```
1 > ParD
2 GSHMMSLKWTRKAAADLDIAIYDHVVLIQPEKALKAVQDIVEQVKP
LQQVANQGAGRSEVPGVRTLTLERWPFSAPFRVKKGKEIQILRIDR
VEITP
3 >ParE
4 GSHMMSTVVSFRADDALVAALDELARATHRDRPYHLRQALAQYLER
QQWQVAAIDEGLADANAGRLLEHIEIEKRWGLQ
```

Differences for Monomer v. Complex Prediction

2. AlphaFold Databases

Monomer Prediction Database

- A. BFD
- B. MGnify
- C. PDB MMCIF
- D. PDB70
- E. UniClust30
- F. Uniref90
- G. –
- H. –

Complex Prediction Database

- A. BFD
- B. MGnify
- C. PDB MMCIF
- D. –
- E. UniClust30
- F. Uniref90
- G. PDB SEQRES
- H. UNIPROT

Tutorial/00_monomer_pred/monomer.slurm (First 35 Lines)

```
1 #!/bin/bash
2 #SBATCH --partition=gpu
3 #SBATCH --exclusive
4 #SBATCH --ntasks=20
5 #SBATCH --ntasks-per-node=20
6 #SBATCH --mem=0
7 #SBATCH --output=%j.out
8 #SBATCH --error=%j.err
9 #SBATCH --time=48:00:00
10 #SBATCH --job-name=INPUT
11
12 # THIS SCRIPT IS FOR MONOMER PREDICTION
13
14 #Loading module for GPU
15 module load CUDA/11.3.1
16
17 # Activates Richard's Conda environment prepared for AlphaFold (Tensorflow, CuDNN, etc.)
18 eval "$(curl -s https://raw.githubusercontent.com/RichardErickson/AlphaFold/master/miniconda3/bin/conda shell.bash hook)"
19 conda activate af2.2.0
20
21 # Use Richard's AlphaFold program
22 AF="/home/van/Programs/alphafold/run_alphaFold.py"
23
24 # Sets output directory to the current folder
25 OUT_DIR=`pwd`
26
27 # Databases for AlphaFold
28 #     DATA_DIR contains the sequence database for MSAs
29 #     DATA_DIR2 contains mmCIF and other files that require periodic updating
30 DATA_DIR="/opt/oscer/software/alphaFold/2.0.0/data"
31 DATA_DIR2="/data/alphaFold"
32
33 # Input name of sequence file. CHANGE THIS BETWEEN DIFFERENT RUNS!
34 input="INPUT"
35
```

SLURM Allocation requests

- Rename INPUT to job name

Load CUDA library (for GPU)

Variable “AF” points to alphaFold program

Variable “OUT_DIR” points to current working directory

Variable “DATA_DIR” points to AF databases

Variable “DATA_DIR2” points to AF databases that update the beginning of every month

Variable “input” is the name of your pasta file (rename INPUT)

Tutorial/00_monomer_pred/monomer.slurm (Lines 35 to EOF)

Max template year for PDBs

```
36 # Set year for the maximum template year (oldest PDB you want included to the prediction).
37 year="2000"
38
39 python ${AF} \
40   --fasta_paths=${input}.fasta \
41   --output_dir=${OUT_DIR} \
42   --max_template_date=${year}-01-01 \
43   --use_gpu_relax=True \
44   --jackhmmer_binary_path=/opt/oscer/software/HMMER/3.2.1-GCC-8.2.0-2.31.1/bin/jackhmmer \
45   --data_dir=${DATA_DIR2} \
46   --bfd_database_path=${DATA_DIR}/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt \
47   --uniref90_database_path=${DATA_DIR}/uniref90/uniref90.fasta \
48   --mgnify_database_path=${DATA_DIR}/mgnify/mgy_clusters.fa \
49   --pdb70_database_path=${DATA_DIR}/pdb70/pdb70 \
50   --uniclust30_database_path=${DATA_DIR}/uniclust30/uniclust30_2018_08/uniclust30_2018_08 \
51   --template_mmcif_dir=${DATA_DIR2}/pdb_mmcif/mmcif_files \
52   --obsolete_pdbs_path=${DATA_DIR2}/pdb_mmcif/obsolete.dat
53
```

- *fasta_paths*
Input sequence (e.g. bax.fasta)
- *output_dir*
Predictions are saved to output directory in this directory
- *max_template_date*
For PDB search
- *use_gpu_relax*
Use GPU accelerated minimization
- *jackhmmer_binary_path*
Points to jackhammer program (probably not needed anymore)
- *The rest are flags pointing to the appropriate databases.*

* The symbol “\” means continue reading next line (ignores line breaks)

Monomer Tutorial (~2.75 hours, node: c301)

1. Make input file:

```
$ vi bax.fasta  
>bax  
MDGSQEPRGGGPTSSEQIMKTGALLQGFIQDRAGRMGGEAPELALDPVPQDASTKKLSECLKRIGDEL  
DSNMELQRMIAAVDTDSPREVFFRVAADMFSDDGNFNWGRVVALFYFASKLVLKALCTKVPELIRTIMGWTLD  
FLRERLLGWIQDQGGWDGLLSYFGTPTWQTIFVAGVLTASLTIWKKMG
```

2. Change INPUT to bax in monomer.slurm (lines 10 & 34)

```
sed -i 's/INPUT/bax/' monomer.slurm
```

3. Submit the SLURM script

```
sbatch monomer.slurm
```

Tutorial/01_complex_pred/complex.slurm (First 35 Lines)

```
1 #!/bin/bash
2 #SBATCH --partition=gpu
3 #SBATCH --exclusive
4 #SBATCH --ntasks=20
5 #SBATCH --ntasks-per-node=20
6 #SBATCH --mem=0
7 #SBATCH --output=%j.out
8 #SBATCH --error=%j.err
9 #SBATCH --time=48:00:00
10 #SBATCH --job-name=INPUT
11
12 # THIS SCRIPT IS ONLY FOR MULTIMER PREDICTION!!!!!!
13
14 # Loading module for GPU
15 module load CUDA/11.3.1
16
17 # Activates Richard's Conda environment prepared for AlphaFold (Tensorflow, CuDNN, etc.)
18 eval "$(curl -s https://raw.githubusercontent.com/RichardErickson/AlphaFold/master/miniconda3/bin/conda shell.bash hook)"
19 conda activate af2.2.0
20
21 # Use Richard's AlphaFold program
22 AF="/home/van/Programs/alphafold/run_alphaFold.py"
23
24 # Sets output directory to the current folder
25 OUT_DIR=`pwd`
26
27 # Databases for AlphaFold
28 #     DATA_DIR contains the sequence database for MSAs
29 #     DATA_DIR2 contains mmCIF and other files that require periodic updating
30 DATA_DIR="/opt/oscer/software/alphaFold/2.0.0/data"
31 DATA_DIR2="/data/alphaFold"
32
33 # Input name of sequence file. CHANGE THIS BETWEEN DIFFERENT RUNS!
34 input="INPUT"
35
```

SLURM Allocation requests

- Rename INPUT to job name

Load CUDA library (for GPU)

Variable “AF” points to alphafold program

Variable “OUT_DIR” points to current working directory

Variable “DATA_DIR” points to AF databases

Variable “DATA_DIR2” points to AF databases that update the beginning of every month

Variable “input” is the name of your pasta file (rename INPUT)

Same as monomer.sh so far..
(Except for comment on line 12)

Tutorial/01_complex_pred/complex.slurm (Lines 35 to EOF)

Max template year for PDBs

```
36 # Set year for the maximum template year (oldest PDB you want included to the prediction).
37 year="2000"
38
39 python ${AF} \
40     --fasta_paths=${input}.fasta \
41     --output_dir=${OUT_DIR} \
42     --max_template_date=${year}-01-01 \
43     --use_gpu_relax=True \
44     --model_preset=multimer \
45     --jackhmmer_binary_path=/opt/oscer/software/HMMER/3.2.1-GCC-8.2.0-2.31.1/bin/jackhmmer \
46     --data_dir=${DATA_DIR2} \
47     --bfd_database_path=${DATA_DIR}/bfd/bfd_metaclust_clu_complete_id30_c90_final_seq.sorted_opt \
48     --uniref90_database_path=${DATA_DIR}/uniref90/uniref90.fasta \
49     --mgnify_database_path=${DATA_DIR}/mgnify/mgy_clusters.fa \
50     --uniclust30_database_path=${DATA_DIR}/uniclust30/uniclust30_2018_08/uniclust30_2018_08 \
51     --template_mmcif_dir=${DATA_DIR2}/pdb_mmcif/mmcif_files \
52     --obsolete_pdbs_path=${DATA_DIR2}/pdb_mmcif/obsolete.dat \
53     --pdb_seqres_database_path=${DATA_DIR2}/pdb_seqres/pdb_seqres.txt \
54     --uniprot_database_path=${DATA_DIR2}/uniprot/uniprot.fasta
```

- All flags are the same as monomer.slurm
- Differences are in lines 53, 54, and removed the PDB70 database

* The symbol “\” means continue reading next line (ignores line breaks)

Complex Tutorial

1. Make input file:

```
$ vi paparde.fasta  
> ParD  
GSHMMSLKWTRKAAADLDIAIYDHVVIGPEKALKAVQDIVEQVKPLQQVANQGAGRSEVPGVRTLTERWPFSAPF  
RVKGKEIQILRIDRVEITP
```

```
>ParE  
GSHMMSTVVSFRADDALVAALDELARATHRDRPYHLRQALAQYLERQQWQVAAIDEGLADANAGRLLEHIEIEKRWGL  
Q
```

2. Change INPUT to paparde to complex.slurm (lines 10 & 34)

```
sed -i 's/INPUT/paparde/' complex.slurm
```

3. Submit the SLURM script

```
sbatch complex.slurm
```

SLURM Commands

- `sbatch [file.slurm]` ; Submits SLURM script to scheduler
- `squeue` ; Check the SLURM queue
 - `squeue -u [username]` ; Check your SLURM queue
 - `squeue -p [partition]` ; Check the queue for a specific partition
- `scancel [job number]` ; Cancel a submitted job

SLURM Commands

- `module load [name]` ; Loads the environment for specified program
- `module purge` ; Cleans loaded modules
- `module list` ; List all currently loaded modules
- `module avail` ; List all available modules
- `module spider [name]` ; Searches for program
- Alternatively, you can look for software in: **/opt/oscer/software**