Supplementary Information for

Computational Modeling of Designed Ankyrin Repeat Protein Complexes with their Targets

Filip Radom[1], Emanuele Paci[2] and Andreas Plückthun[1]

[1]Department of Biochemistry, University of Zurich, Zurich, Switzerland

[2]Astbury Centre for Structural Molecular Biology, University of Leeds, Leeds, United Kingdom

Andreas Plückthun
Department of Biochemistry,
University of Zurich,
8057 Zurich, Switzerland
Tel: +41 44 635 5570
FAX: +41 44 635 5712
Email: plueckthun@bioc.uzh.ch

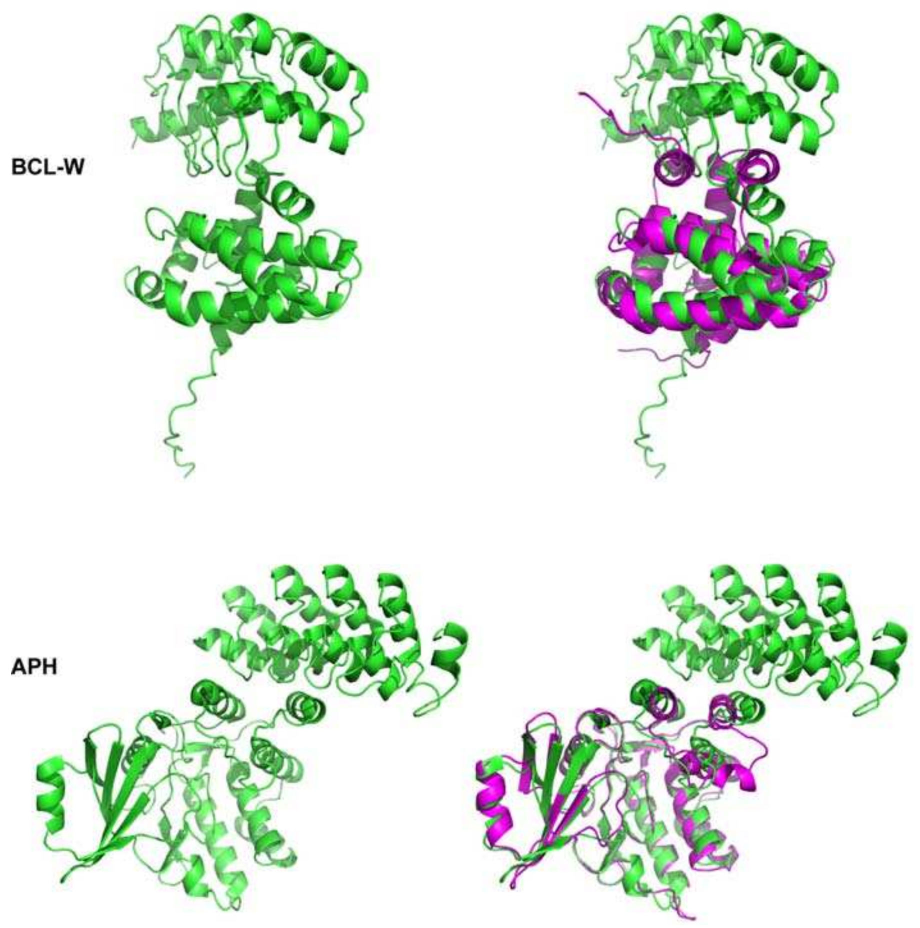
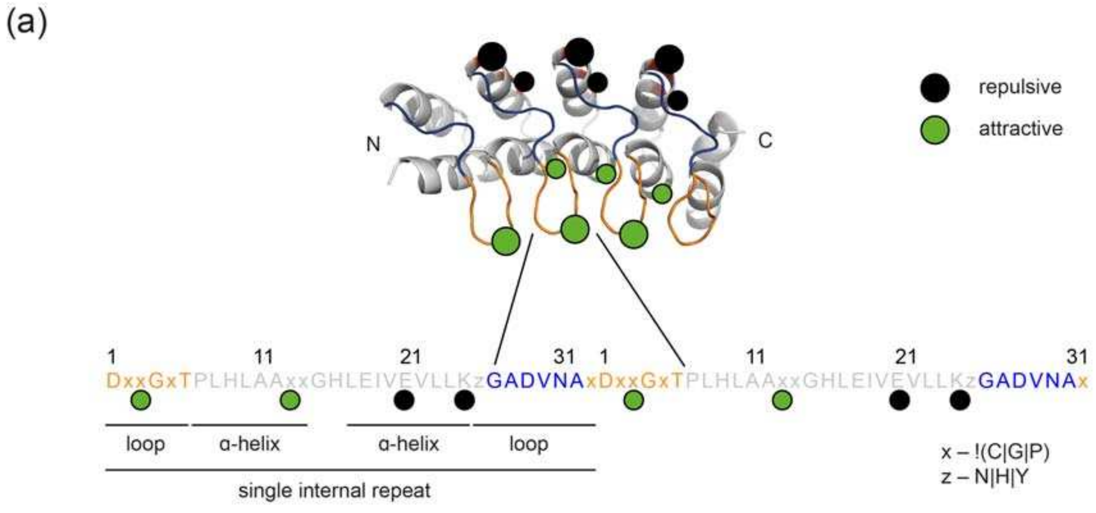


Fig. S1. Structures of the challenging complexes and the structural alignment of bound (green) and unbound (magenta) targets. In the unbound structure of BCL-W, the flexible C-terminus adapts a helical conformation that shields the DARPin's epitope; PDB ID: 4K5A, 1MK3. Two helices in APH spread significantly to accommodate the DARPin; PDB ID: 2BKK, 1J7I.



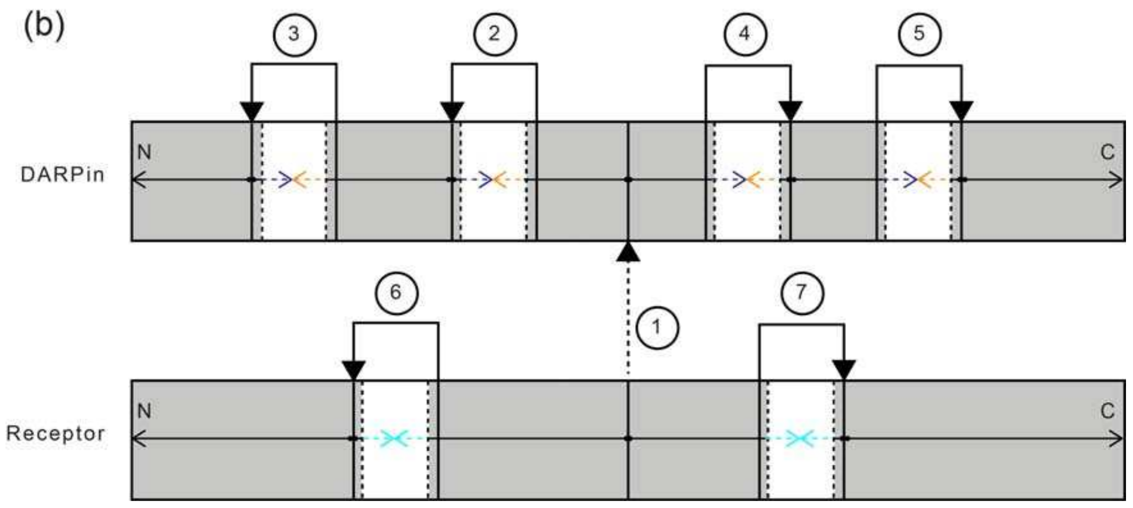


Fig. S2. Strategy for flexible docking. (a) Schematic representation of a DARPin structure. Residues constrained to be involved in interaction with the receptor or explicitly not involved are indicated as green or black spheres, respectively. Only two of the green spheres (one in α-helix and one in loop) need to be involved. (b) A scheme of relations between rigid and flexible protein fragments (fold tree) designed for flexible docking of DARPins with Rosetta. Circled are jump numbers (virtual bonds between residues). Gray regions or solid arrows indicate protein segments considered as rigid (backbone atoms). White regions with dashed lines represent flexible parts: DARPin loops (blue and orange; as in (a)) and receptor regions defined as flexible according to our Rosetta backrub-based protocol (as in Fig. 3). Note that each receptor will have a different number of flexible regions with very different spacing. Jump 1 (dashed arrow) is a connection between the centers of mass of both molecules and is also flexible (this allows motion of molecules relative to each other).

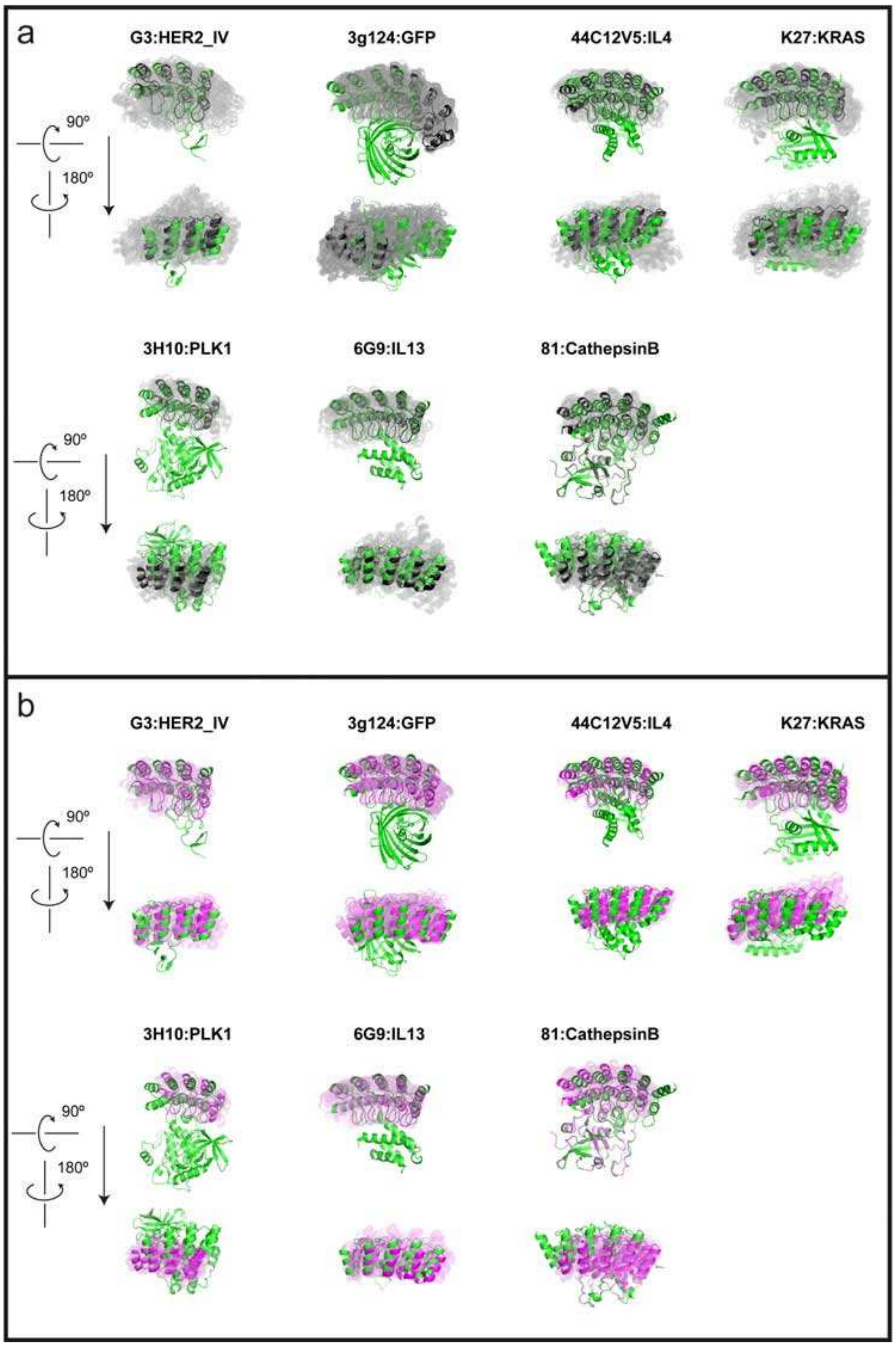


Fig. S3. (a) Models from the largest cluster after clustering within 5 Å (grey) aligned to the crystal structure (green). The center of the cluster is in black. (b) Models from the largest cluster after further clustering within 2 Å (pink) aligned to the crystal structure (green). The center of the cluster is in magenta. Further clustering thus removes decoys that are more remote from the native structure.

Table ST1. Discriminative power of different Rosetta scores and the additional features[a]

Rosetta scores	G3:HER2	3g124:GFP	44C12V5:IL4	K27:KRAS	3H10:PLK1
total_score	0[b]	0	0	0	0
I_sc	0	1	0	1	1
ddg	0	0	0	1	1
dG_separated	0	1	0	1	1
packstatstat	1	0	1	0	1
sc_value	0	0	0	1	1
Additional metrics[c]					
packstat*dG_separated	0	1	1	1	1
score*I_sc	0	1	0	1	1
score*dG_separated	0	1	0	1	0
I_sc*packstat	0	1	1	1	1
I_sc*dG_separated	0	1	0	1	0
packstat*dG_separated*sc_value	1	1	0	1	1
packstat*sc_value	0	0	0	0	1
I_sc*packstat*sc_value	1	1	0	1	1
I_sc*packstat*ddG_separated*sc_value	0	1	0	1	1
I_sc*sc_value	0	1	0	1	1
packstat**2*dG_separated*sc_value (p2gs)	1	1	1	1	1

[a] Average scores of entire clusters (1000 decoys each) are compared.

[b] 1 if the score enables discrimination of the near-native cluster.

[c] Introduced by multiplying some Rosetta scores, i.e., considering them simultaneously.

Table ST2. p2gs scores of sets of decoys derived from 3 models after sequential clustering

cluster number[a]	G3:HER2	3g124:GFP	44C12V5:IL4	K27:KRAS	3H10:PLK1	6G9:IL13	81:CathepsinB
1	-7.52[b]	-6.43	-5.57	-6.56	-6.01	-4.58	-5.31
2	-7.16	-6.24	-4.76	-5.13	-5.47	-5.20	-5.30
3	-6.97	-5.83	-4.30	-4.28	-7.31	-3.62	-7.84

[a] Rank after sequential clustering.

[b] Lowest score corresponds to near-native cluster.

Table ST3. Quality of top models sorted by p2gs score

rank p2gs	G3:HER2	3g124:GFP	44C12V5:IL4	K27:KRAS	3H10:PLK1	6G9:IL13	81:CathepsinB
1	0.57 / 10.45 / 3.39	0.33 / 14.58 / 4.94	0.62 / 3.70 / 1.68	0.38 / 9.22 / 3.19	0.75 / 8.01 / 3.00	0.57 / 5.70 / 2.68	0.40 / 11.63 / 3.05
2	0.69 / 3.51 / 1.97	0.48 / 9.82 / 2.90	0.56 / 3.66 / 1.61	0.30 / 12.24 / 5.27	0.79 / 5.89 / 1.94	0.66 / 2.23 / 1.75	0.51 / 8.50 / 2.75
3	0.69 / 3.44 / 2.08	0.46 / 12.70 / 3.70	0.66 / 2.22 / 1.41	0.62 / 7.50 / 2.50	0.81 / 5.49 / 1.82	0.69 / 2.73 / 1.96	0.76 / 6.09 / 2.54
4	0.71 / 2.90 / 1.92	0.63 / 8.78 / 2.43	0.60 / 3.85 / 1.65	0.48 / 9.11 / 3.33	0.79 / 5.23 / 1.93	0.66 / 2.12 / 1.75	0.73 / 6.27 / 2.18
5	0.63 / 3.37 / 2.02	0.76 / 3.70 / 1.42	0.46 / 5.11 / 2.43	0.42 / 8.54 / 3.17	0.64 / 10.13 / 2.30	0.60 / 4.04 / 2.04	0.71 / 6.41 / 2.23
6	0.67 / 2.88 / 1.93	0.33 / 13.93 / 4.75	0.40 / 6.34 / 2.62	0.55 / 7.48 / 3.69	0.51 / 9.84 / 2.48	0.69 / 2.17 / 1.93	0.02 / 18.33 / 6.84
7	0.67 / 3.42 / 1.93	0.83 / 4.78 / 1.51	0.29 / 8.61 / 4.18	0.85 / 3.05 / 1.79	0.83 / 4.38 / 1.61	0.69 / 2.55 / 1.97	0.53 / 7.51 / 3.16
8	0.75 / 3.64 / 1.98	0.87 / 1.86 / 1.29	0.65 / 3.60 / 1.58	0.90 / 2.54 / 1.70	0.83 / 5.65 / 1.78	0.71 / 3.12 / 1.97	0.76 / 6.39 / 2.53
9	0.47 / 12.18 / 3.88	0.91 / 4.00 / 1.51	0.65 / 1.80 / 1.42	0.50 / 9.40 / 3.26	0.77 / 9.07 / 3.16	0.64 / 3.16 / 1.95	0.80 / 5.74 / 2.31
10	0.63 / 3.25 / 1.99	0.33 / 14.00 / 4.00	0.41 / 6.10 / 2.71	0.77 / 4.53 / 2.07	0.81 / 4.88 / 1.76	0.74 / 3.03 / 2.06	0.89 / 3.21 / 1.17

[a] f(nat) / L-RMSD / I-RMSD. Highest value of f(nat) and lowest values of L-RMSD and I-RMSD among top 10 are underlined.
CAPRI quality criteria:

High quality: f(nat) ≥ 0.5 AND (L-RMSD ≤ 1.0 OR I-RMDS ≤1.0)

Medium quality: f(nat) ≥ 0.3 AND (1.0 < L-RMSD ≤ 5 OR 1.0 < I-RMDS ≤2.0)

Acceptable quality: f(nat) ≥ 0.1 AND (5 < L-RMSD ≤ 10 OR 2 < I-RMDS ≤4)

Table ST 4. Bound docking in ClusPro

DARPin:target	PDB ID (chains)	Rank of first near-native (< 10 Å L-RMSD)	Additional remarks
G3:HER2	4HRN (BC)	2	
3g124:GFP	5MA6 (AB)	3	
44C12V5:IL4	4YDY (AI)	1	
K27:KRAS	5O2S (AB)	1	
3H10:PLK1	2V5Q (BC)	1	
6G9:IL13	5KNH (DI)	1	
81:CathepsinB	5MBL (AB)	3	
D12:BCL-W	4K5A (AB)	2	
AR_3a:APH	2BKK (CD)	1	
9.29:HER2	4HRL (AC)	4	
A-C2:tubulin	5EYP (BF)	9	>400 aa target
D7.18:caspase7	4LSZ (CDF)	3	Dimer. Monomer with repulsive restraints
1D5:NEMO CC2-LZ	2V4H (ABC)	1	Binding to dimer. Dimer receptor used for docking
1D3:knob	4ATZ (AD)	—	Rank #1 was 12 Å off
off7:MBP	1SVX (AB)	—	
8.4:caspase8	2Y1L (CDEG)	—	Dimer. Monomer with repulsive restraints
E2_79:IgE Fc	4GRG (AD)	—	Dimer. Monomer with repulsive restraints

Supplementary Methods

Generation of templates for DARPin modeling

Two molecules of N3C consensus DARPin (2QYJ or 2XEE) were loaded into Pymol and repeat 3 without the terminal residue — because it differs in the last repeat of the consensus DARPin from other repeats — of molecule 2 was aligned to the corresponding residues in repeat 2 of molecule 1 (all-atom). Coordinates of N-cap + repeat 1 of molecule 1 and repeat 3 + C-cap of molecule 2 were saved separately and combined into one molecule with Chimera software. Chains were renamed and renumbered accordingly. The peptide bond between Lys-Asp was automatically recognized.

Protocol for modeling of DARPin-target complexes

Homology modeling of a DARPin

1. Identify the type of DARPin to be modelled (number of internal repeats, type of C-cap) [1-3].
2. Choose the corresponding PDB template file for modeling (2QYJ for an N3C DARPin with the original C-cap, 2XEE for an N3C DARPin with the Mut5 cap, or corresponding templates for N2C DARPins (all available both in *DARPin_fixbb* and *DARPin_remodel*).

   In most cases only substitution mutations will be necessary, which will be done with fixed backbone design application [4], described in section 3 A . If a DARPin contains insertions or deletions, use the remodel tool [5] and move to section 3 B.
3. **A.** Only mutations in DARPin template are necessary:

   i. Modify *<template>.resfile* in *DARPin_fixbb* to introduce required mutations:

   Example:

```
30  D  NATRO
31  D  PIKAA E
32  D  NATRO
```

   Here residues 30 and 32 of chain D remain unchanged (NATRO: natural rotamer) and residue number 31 will be mutated to glutamic acid (E) (PIKAA: pick any amino acid in a single-letter code).

   ii. Run Rosetta **fixbb**:

```
fixbb.linuxgccrelease\
-in:file:s ./<template>.pdb\
-out:path:all ./rosetta_output\
-resfile <template>.resfile\
-use_input_sc\
-ex1\
-ex2\
-minimize_sidechains\
-nstruct 1
```

Note that the Rosetta numbering of the sequence begins with residue 1, e.g., the N-terminal Asp 13 (D13) will be thus considered as Asp 1. The templates are renumbered to match this scheme. We recommend to align the sequence of the produced model to the desired sequence to make sure that all the intended mutations were introduced correctly.

B. Alternative: Making insertions and/or deletions in DARPin

i. Modify the blueprint file *<template>.remodel* to introduce required mutations:

Example:

```
30 A .
31 K . PIKAA R
32 W D PIKAA W
34 G D PIKAA Q
36 Y . PIKAA T
```

Here residue 33 will be deleted. Residue 32 will be remodelled (side chain and backbone angles) but not mutated, residue 34 will be remodelled and mutated, and residues 31 and 36 will be mutated but the backbone will stay fixed. Residue 30 will remain unchanged. We recommend to allow remodeling of 3 residues that flank the insertion/deletion site. D before PIKAA allows modeling any secondary structure. We found this option better than forcing modeling loops or helices.

Rosetta **`remodel`** can also be used for substitution mutations only, as an alternative to **`fixbb`** (although we have not done this for our cases).

ii. Run the Rosetta **remodel**:

```
remodel.linuxgccrelease\
-in:file:s ./<template>.pdb\
-out:path:all ./rosetta_output\
-run::chain D\
-remodel:blueprint <template>.remodel\
-use_input_sc\
-ex1\
-ex2\
-nstruct 30
```

iii. List the produced models

```
ls -d $PWD/<models>* > files.txt
```

iv. Cluster the models with Rosetta **cluster**:

```
custer.linuxgccrelease\
-in:file:l files.txt\
-in:file:fullatom\
-cluster:radius 0.2\
-run:shuffle
```

v. Select best-scoring (lowest-energy) model from the largest cluster.

4. Refine the produced model with Rosetta **relax**:

```
relax.linuxgccrelease\
-in:file:s ./<model>.pdb\ #output from fixbb/remodel
-in:file:fullatom\
-out:path:all ./rosetta_output\
-out:file:scorefile output.sc\
-out:file:silent output.out\
-nstruct 40\
-relax:thorough
```

5. Cluster the models with Rosetta **cluster**:

```
cluster.linuxgccrelease\
-in:file:silent output.out
-in:file:fullatom
-cluster:radius 0.3
-run:shuffle
```

6. Select the best scoring model (lowest energy) from the largest cluster.

Receptor modeling

1. Identify the unbound structure of the receptor in the Protein Data Bank.

   If more than one structure exist, choose the one that is most complete, unliganded and at best resolution. Most small ligands do not usually affect the structure.

2. Prepare the structure for modeling in Rosetta

   ```
   grep "^ATOM" <receptor>.pdb > <receptor>.clean.pdb
   ```

   Then in Pymol:

   ```
   alter (all),resi=str(int(resi)-<identifier_of_first_res> + 1)
   #renumber for Rosetta
   alter (all), chain ='A' #rename chain to A
   ```

3. Relax the structure with all-heavy-atom constraints.

   ```
   relax.linuxgccrelease\
   -in:file:s ./<receptor>.clean.pdb\
   -in:file:fullatom\
   -out:path:all ./rosetta_output\
   -out:file:scorefile output.sc\
   -nstruct 1\
   -ex1\
   -ex2\
   -use_input_sc\
   -flip_HNQ\
   -no_optH false\
   -relax:constrain_relax_to_start_coords\
   ```

```
-relax:coord_constrain_sidechains\
-relax:ramp_constraints false\
-ignore_zero_occupancy false
```

This step relaxes the structure into the Rosetta energy function, i.e., bond geometries that score particularly poor in Rosetta are idealized, i.e., those bond lengths and angles are set to values that are statistically more frequent in proteins (according to the rotamer library).

4. Run Rosetta **backrub** on the entire receptor:

```
backrub.mpi.linuxgccrelease\
-in:file:s ./<receptor>.clean_0001.pdb\ #relaxed receptor
-in:file:fullatom\
-out:file:scorefile receptor.backrub250.sc\
-nstruct 250\
-backrub:ntrials 10000\
```

This step generates ensembles over the entire structure of the receptor.

The calculation can be accelerated using the Message Passing Interface (MPI) on multiple processor cores. The command then begins with:

```
mpiexec/mpirun -np <number_of_cores>
```

5. Rename the generated models (250 in this case) with the Python script provided in the Supplementary Files:

```
python rename.py
```

6. Load the renamed models into a Pymol session, together with the relaxed receptor from point 3 and run the *pymol_rmsf.py* script provided in the Supplementary Files within Pymol:

```
run pymol_rmsf.py
```

then call functions: `fit_unbound`, `dev_fragment` with the arguments described within the script.

`fit_unbound` aligns all ensembles to the input receptor. `dev_fragment` will output standard deviation of RMSD (called root mean square fluctuation; RMSF) along segments of the protein chain (by default, segments are 3 amino acid long). A calculation of a single RMSD value in Pymol takes ~0.5 s, so a full calculation may take up to several hours (e.g., for 100 residue protein, calculating all possible 3-residue segments for all 250 ensembles takes ~4 h).

7. Consider segments of RMSF > 0.2 Å as flexible.
8. Run Rosetta **backrub** on the flexible loops of the receptor:

```
backrub.mpi.linuxgccrelease\
-in:file:s ./ receptor.clean.relaxed.pdb\
-in:file:fullatom\
-out:file:scorefile receptor.backrub_loop.sc\
-nstruct 20\
-backrub:ntrials 10000\
-pivot_residues <flexible_residues> #here put numbers of flexible
residues
```

9. Select 20 generated loop ensembles for the next step.

Remarks:

Rosetta backrub outputs the best scoring poses within a number of backrub trials (models ending with _0001 or _low) or last-sampled poses (ending with _last). Use _last poses in order to retain more diverse conformations.

Rigid-body docking with ClusPro and sequential clustering

1. Register on https://cluspro.bu.edu/
2. Start a new job for each of the receptor loop ensembles (20 in total):
    1. Upload PDB files for the receptor and the ligand (the latter is always the same DARPin model, only the receptor inputs are varied)
    2. In the menu Attraction and Repulsion add repulsive constraints to the DARPin chain. These are Glu and Lys residues at the backside of the DARPin (indicated in **Error! Reference source not found.**). For a typical N3C

DARPin without insertions/deletions, in Rosetta numbering scheme, this would be:

```
d-52 d-56 d-85 d-89 d-118 d-122
```

For N2C:

```
d-52 d-56 d-85 d-89
```

Check numbering if insertions/deletions are present.

3. Download the top 10 models from each docking simulation and place them into corresponding folders (en1-en20 in Supplementary Files).
4. Run *ClusPro_rename.py* script to rename the files and to move them to another folder for prepacking.
5. Navigate to the *docking_prepack* folder and list the models:

```
ls -d $PWD/en* > files.txt
```

6. Prepack the models with Rosetta **docking_prepack_protocol** [6]:

```
docking_prepack_protocol.linuxgccrelease\
-in:file:l ./files.txt\
-out:path:all ./rosetta_output\
-docking:partners A_D #receptor_ligand chains
```

Remarks:

This step relaxes side chains into the Rosetta energy function. Make sure that chain names in `partners A_D` are correct.

7. Move output .pdb files (ending with _0001) to the *sequential_clustering* folder and list them:

```
ls -d $PWD/en* > files.txt
```

8. Cluster the models within 5 Å radius with Rosetta **cluster**:

```
cluster.linuxgccrelease\
-in:file:l files.txt\
-in:file:fullatom\
-cluster:radius 5\
-run:shuffle
```

Cluster application outputs .pdb files labelled as c.<cluster_number>.<model_number>. c.0, c.1 and c.2 are the largest clusters within a set. <mode_number>=0 is the center of the cluster.

9. Move models from clusters to corresponding subfolders and rename them:

```
python rename.py
```

10. List renamed files in each subfolder:

```
ls -d $PWD/m* > files.txt
```

11. Cluster the models from each subfolder within 2 Å radius with Rosetta **cluster**:

```
cluster.linuxgccrelease\
-in:file:l files.txt\
-in:file:fullatom\
-cluster:radius 2\
-run:shuffle
```

12. Rename the center of the largest subcluster (rename the subcluster center c.0.0 of cluster c.1 to c.1.0.0, etc.)

Flexible docking with Rosetta and final ranking

1. Move the 3 best models (centers of 3 subclusters) to *Rosetta_flexible_docking* folder and list them:

```
ls -d $PWD/c.* > files.txt
```

2. Modify the constraint file *constraint.cst* if necessary.
Example:

```
SiteConstraint CA 52D A SIGMOID 8.0 -2.0
```

This is a repulsive constraint where residue 52 of a DARPin (chain D) is penalized for contacting the receptor (chain A).

The templates for full length DARPins are provided. Correct the numbering if a DARPin contains insertions/deletions. Check the receptor chain name and modify if necessary.

3. Set up the movemap and the fold tree according to Fig S2b

   Flexible parts of the receptor are as determined with `backrub` in Receptor modeling, point 7.

   DARPin flexible loops as schematically depicted in Fig S2b. In PDB numbering, for N3C without insertions/deletions, these are residues:

   ```
   G37-T49, G70-T82, G103-T-115, G136-T148
   ```

   For N2C:

   ```
   G37-T49, G70-T82, G103-T-115
   ```

   A simple way to calculate the Rosetta numbering of the DARPin loops in complex models is:

   `pdb_number – 12 + receptor_length` (i.e., G37 would be 125 if the receptor is 100-aa long)

   Specify flexible parts in Rosetta script: modify MoveMap in MinMover in *DARPin_flex.xml*. Change only values in 'begin' and 'end'.

   To design a fold tree, one needs information about the centers of mass. This can be quickly checked by starting a rigid-body docking with default options. Copy one of the models to *rigid_dock_to_check_centers* and run:

   ```
   docking_protocol.linuxgccrelease\
   -in:file:s ./<name>.pdb
   -nstruct 1
   -use_input_sc
   -partners A_D
   ```

Under ‘new fold tree’ a fold tree for rigid molecules is created and the centers of masses are joined by jump 1. If a center of mass happens to be in a flexible region, pick the first closest rigid residue.

Set up the fold tree for flexible docking in *fold_tree_DARPin.txt*:

Example:

```
EDGE 185 209 -1
EDGE 217 210 -1
EDGE 203 217 4
EDGE 217 242 -1
```

This part of a fold tree includes a flexible loop between residues 204-216. There is a rigid virtual connection between the residues surrounding it (jump 4 between residues 203-217). There is an artificial chain break between residues 209/210. Artificial chain breaks are introduced in the middle of the flexible loops in receptors and between A42/K43, A75/K76, A108/K109, A141/Q142 (PDB numbering) in the full-length N3C DARPin.

Because of Rosetta numbering, the DARPin in a model of the complex will again change its residue identifiers. The DARPin residue that was numbered N in homology modeling will now be N + the length of the receptor (e.g., Asp13 in PDB numbering, that became Asp1 for DARPin homology modeling will now be Asp101 if this DARPins is docked to a 100-aa residue receptor).

Creating a fold tree is definitely the most difficult part of the procedure. We recommend to draw it by hand and test it in a single-trajectory simulation (For this, set *confidence="0"* for ddg filter in *DARPin_flex.* and set `-nstruct 1` in option flags in the following point.)

4. Run the flexible docking protocol for all models:

```
rosetta_scripts.mpi.linuxgccrelase\
-in:file:l ./files.txt
-out:path:all ./rosetta_output
```

```
-out:file:scorefile <name>.sc
-nstruct 1000
-jd2:ntrials 100
-ex1
-ex2aro
-use_input_sc
-partners A_D
-dock_pert 3 8
-score:docking_interface_score 1
-parser:protocol DARPin_flex.xml
-parser:view
-constraints:cst_file constraint.cst
-constraints:cst_fa_file constraint.cst
-cst_weight 5
-cst_fa_weight 5
```

5. Navigate to output files and list the 1000 models coming from each input model separately:

   ```
   ls -d $PWD/c.<0/1/2>* > files_c.<0/1/2>.txt
   ```

6. Analyze 1000 models from each simulation independently with **InterfaceAnalyzer**:

   ```
   InterfaceAnalyzer.mpi.linuxgccrelease\
   -in:file:l files_c.<0/1/2>.txt\
   -in:file:fullatom\
   -add_regular_scores_to_scorefile\
   -out:file:scorefile Interface_c.<0/1/2>.txt\
   -out:file:score_only\
   -compute_packstat 1
   ```

7. Calculate the average dG_separated, packstat and sc_value over models generated from each input structure and compute packstat × packstat × dG_separated × sc_value (p2gs) values for each cluster. Pick the one with the

lowest score as the near-native cluster. This can be done, e.g., with the provided script:

```
python p2gs_calculator.py
```

8. Identify the best model in the cluster according to binding energy, e.g.,

```
cat Interface_c.<0/1/2>.txt | sort -nk 6 | awk '{print $NF}' | head -
1
```

Supplementary Files

```
├── ClusPro
│   ├── ClusPro_rename.py
│   ├── docking_prepack
│   │   ├── flags
│   │   └── rosetta_output
│   ├── en1
│   ├── en10
│   ├── en11
│   ├── en12
│   ├── en13
│   ├── en14
│   ├── en15
│   ├── en16
│   ├── en17
│   ├── en18
│   ├── en19
│   ├── en2
│   ├── en20
│   ├── en3
│   ├── en4
│   ├── en5
│   ├── en6
│   ├── en7
│   ├── en8
│   ├── en9
│   └── sequential_clustering
│       ├── c.0_cluster2A
│       │   ├── flags_cluster
│       │   ├── rename.py
│       │   └── rename.txt
│       ├── c.1_cluster2A
│       │   ├── flags_cluster
│       │   ├── rename.py
│       │   └── rename.txt
│       ├── c.2_cluster2A
│       │   ├── flags_cluster
│       │   ├── rename.py
│       │   └── rename.txt
│       └── flags_cluster
├── DARPin_model
```

```
│   ├── DARPin_fixbb
│   │   ├── 2qyj.clean.pdb
│   │   ├── 2qyj_N2C.clean.pdb
│   │   ├── 2qyj_N2C.resfile
│   │   ├── 2qyj.resfile
│   │   ├── 2xee_A.clean.pdb
│   │   ├── 2xee_A_N2C.clean.pdb
│   │   ├── 2xee_A_N2C.resfile
│   │   ├── 2xee_A.resfile
│   │   ├── flags
│   │   └── rosetta_output
│   │       └── relax
│   │           ├── flags
│   │           └── rosetta_output
│   │               └── flags_cluster
│   └── DARPin_remodel
│       ├── 2qyj.clean.pdb
│       ├── 2qyj_N2C.clean.pdb
│       ├── 2qyj_N2C.remodel
│       ├── 2qyj.remodel
│       ├── 2xee_A.clean.pdb
│       ├── 2xee_A_N2C.clean.pdb
│       ├── 2xee_A_N2C.remodel
│       ├── 2xee_A.remodel
│       ├── flags
│       └── rosetta_output
│           ├── flags_cluster
│           └── relax
│               ├── flags
│               └── rosetta_output
│                   └── flags_cluster
├── README_docking_DARPins.txt
├── Receptor_model
│   ├── relax_with_constraints
│   │   ├── flags
│   │   └── rosetta_output
│   └── search_flexible
│       ├── Backrub_on_full
│       │   ├── flags
│       │   ├── pymol_rmsf.py
│       │   ├── rename.py
│       │   └── rename.txt
│       └── Backrub_on_loops
│        └── flags
└── Rosetta_flexible_docking
   ├── constraint.cst
   ├── DARPin_flex.xml
   ├── DARPin_loops_pdb_to_rosetta.xlsx
   ├── flags_pert_flex
   ├── fold_tree_DARPin.txt
   ├── loop_exe.sh
   ├── rigid_dock_to_check_centers
   │  └── flags_pert
   └── rosetta_output
      ├── flags_int
      └── p2gs_calculator.py
```

Supplementary Files can be downloaded from ###URL ADDRESS###

References

[1] G. Interlandi, S.K. Wetzel, G. Settanni, A. Plückthun, A. Caflisch. Characterization and further stabilization of designed ankyrin repeat proteins by combining molecular dynamics simulations and experiments, J. Mol. Biol. 375 (2008) 837-854.

[2] M.A. Kramer, S.K. Wetzel, A. Plückthun, P.R. Mittl, M.G. Grütter. Structural determinants for improved stability of designed ankyrin repeat proteins with a redesigned C-capping module, J. Mol. Biol. 404 (2010) 381-391.

[3] J. Schilling, J. Schöppe, A. Plückthun. From DARPins to LoopDARPins: novel LoopDARPin design allows the selection of low picomolar binders in a single round of ribosome display, J. Mol. Biol. 426 (2014) 691-721.

[4] B. Kuhlman, G. Dantas, G.C. Ireton, G. Varani, B.L. Stoddard, D. Baker. Design of a novel globular protein fold with atomic-level accuracy, Science 302 (2003) 1364-1368.

[5] P.S. Huang, Y.E. Ban, F. Richter, I. Andre, R. Vernon, W.R. Schief, et al. RosettaRemodel: a generalized framework for flexible backbone protein design, PLoS One 6 (2011) e24109.

[6] J.J. Gray, S. Moughon, C. Wang, O. Schueler-Furman, B. Kuhlman, C.A. Rohl, et al. Protein-protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations, J. Mol. Biol. 331 (2003) 281-299.