PNAS
www.pnas.org

Supplementary Information for

Allosteric cooperation in a de novo designed two-domain protein.

Fabio Pirro[a,1], Nathan Schmidt[b,1], James Lincoff[b], Zachary X. Widel[c], Nicholas F. Polizzi[b], Lijun Liu[d,e,2], Michael J. Therien[c], Michael Grabe[b], Marco Chino[a], Angela Lombardi[a,*], William F. DeGrado[b,*]

[a]Department of Chemical Sciences, University of Napoli Federico II, Via Cintia, 26, 80126 Napoli, Italy
[b]Department of Pharmaceutical Chemistry and the Cardiovascular Research Institute, University of California at San Francisco, San Francisco, California 94158-9001, United States
[c]Department of Chemistry, 124 Science Drive, Duke University, Durham, North Carolina 27708-0346, United States
[d]State Key Laboratory of Chemical Oncogenomics, School of Chemical Biology and Biotechnology, Peking University Shenzhen Graduate School, Shenzhen 518055, China
[e]DLX Scientific, Lawrence, KS 66049, USA

[1]F.P. and N.S. contributed equally to this work.
[2] Present address: Department of Pharmaceutical Chemistry, University of Kansas, Lawrence, KS 66047, USA
*To whom correspondence should be addressed.
Email: alombard@unina.it; william.degrado@ucsf.edu

This PDF file includes:

Computational design of DFP proteins

Bundle axis determination, co-linear alignment of the two bundles, and generation of all structural poses

A python program, bundle_aligner.py, was written to streamline the generation of all poses used in the MASTER searches to determine designability. The full code is included below.

For all designs the DF1 structure (PDB ID: 1JMB) and PS1 structure (PDB ID: 5TGY), were used. We first identified 12 residue helical segments for each of the four chains in the bundle surrounding the dimetal or porphyrin binding sites. In DF1, helix 1 is chain B residues 5-16, helix 2 is chain C residues 5-16, helix 3 is chain C residues 31-42, and helix 4 is chain B residues 31-42. In PS1, helix 1 is chain A residues 5-16, helix 2 is chain A residues 38-49, helix 3 is chain A residues 60-71, helix 4 is chain A residues 93-104. In both bundles the Cα positions of each helical segment were fit to a cylinder (1). The fit provided the cylinder axis, which was used to approximate the helical axis of a segment. Finally, the four helical axes in the bundle were point-by-point averaged to calculate the bundle axis. Once the helical axes of both bundles were determined, they were aligned along the z-axis by applying the appropriate the translation and rotation matrices. Finally, pdb coordinates of all poses were generated by fixing DF1 and moving PS1 along their shared z-axis by ΔZ = 17 to 26 Angstroms using 0.25 Angstrom steps, and rotating PS1 about the z-axis using angular offset $\Delta\Phi$ =-65^o to 25^o using 2.5^o steps. The coordinate positions of DF1 and PS1 were defined by the midpoint of the helical axis of helix 1 for both bundles, ($\mathbf{X_{DF1}, Y_{DF1}, Z_{DF1}}$) and ($\mathbf{X_{PS1}, Y_{PS1}, Z_{PS1}}$), respectively. The Z distance offset is ΔZ = || $\mathbf{Z_{DF1}}$ - $\mathbf{Z_{PS1}}$||, and the angular offset is Cos[$\Delta\Phi$] = [($\mathbf{X_{DF1}}$ - $\mathbf{X_{PS1}}$)^2 + ($\mathbf{Y_{DF1}}$ - $\mathbf{Y_{PS1}}$)^2]^0.5/[($\mathbf{X_{DF1}}$^2 + $\mathbf{Y_{DF1}}$^2)^0.5 ($\mathbf{X_{PS1}}$^2 + $\mathbf{Y_{PS1}}$^2)^0.5].

MASTER searches

Database of structures

All structures were obtained from the Protein Database. Entries were queried for X-ray crystal structures with resolution less than 2.0 Å. In each retrieved pdb file, a single protein chain was extracted into a new coordinate file that was included in the database. Each of the coordinate files was converted into a binary structural file using MASTER (v1.3) with the createPDS executable. A total of 15,768 chains comprised the library.

Structural searches

Each pose consisting of a pair of disjointed helices was converted into a searchable file using MASTER (v1.3) with the createPDS executable. Next, the pose was used to query the database of structures. Matches were counted if the backbone RMSD of the query structure to a library structure fragment were less than 1.0 Angstrom and the structural match was a continuous chain. The designability of the query structure was quantified by the number of matches found in the library. For a given pair of disjointed helices the designability of 37steps $\times$ 37steps = 1369 distinct poses were scored.

Rosetta design

After constructing the DFP1 backbone, Rosetta was used to design the sequences of backbone fragments connecting the four helices comprising the bundle. All other residue positions were set to the amino acids at those positions in DF1 and PS1.

The Rosetta script file and accompanying resfile are included in SI. Here we briefly summarize the protocol. Backbone minimization was first performed with Cα positions fixed to remove any unphysical backbone geometries resulting from DFP1 construction. Next a round of design was conducted using the soft-repulsive force field 'soft_rep_design' with layer design where packing progressed from protein interior, to the interior-exterior boundary interface, and finally to the protein surface. This was followed with another round of design using the standard talaris2014 'hard' force field with layer design. After this first design iteration, Rosetta backrub was performed on a 16-residue window around each of the four helices used to connect the helical bundles. Following backrub, the bundle was subjected to subsequent iterations of soft and hard force-field design using the GenericMonteCarlo mover. In total, 400 designs were generated, with the vast majority of sequences converging on the final protein sequence of DFP1.

Experimental characterization of DFP proteins

Protein expression and purification

All chemicals were purchased from either Sigma or Fisher Scientific. The gene for DFP1 was introduced into vector pET28a (Novagen), with an N-terminal hexahistidine tag followed by a TEV protease site using NcoI and XhoI restriction site. The genes for DFP2 and DFP3 were introduced into vector pET11a (Novagen), with an N-terminal hexahistidine tag followed by a TEV protease cleavage site, via Gibson assembly. The cloned gene sequences were E. coli codon optimized by Genscript:

DFP1

ATGCACCACCATCATCACCACGAGAACCTGTATTTCCAAGGCGATTACTTGCGCGAGCTTTT
AAAACTGGAGCTGCAAGCAATTAAACAATATGAAAAACTTCGCCAAACTGGAGATGAACTGG
TCCAGGCTTTCCAGCGTCTGCGTGAAATCTTTGACAAGGGCGACGATGACTCCTTGGAACAA
GTATTGGAAGAGATCGAGGAGTTGATTCAGAAGCACCGTCAACTTGCGTCTGAGTTACCAAA
GCTGGAACTTCAAGCGATCAAACAGTACCGTGAGGCTTTAGAGTACGTTAAATTGCCCGTGC
TGGCGAAGATTCTGGAAGATGAAGAGAAACACATTGAGTGGCTTAAGGAAGCGGCCAAGCA
AGGCGATCAGTGGGTACAACTGTTTCAACGCTTTCGCGAAGCCATCGACAAAGGTGATAAA
GATAGTCTTGAGCAGCTGCTGGAGGAACTGGAACAGGCTTTACAAAAGATTCGCGAATTGAC
CGAGAAAACTGGCCGTAAAATCCTTGAAGACGAGGAAAAGCATATCGAGTGGTTGGAAACA
ATCTTAGGGTAA

DFP2

ATGCACCACCATCATCACCACGAGAACCTGTATTTCCAAGGCGATTACCTGCGCGAACTGCT
GAAGGGCGAACTGCAAGGGATCAAGCAGTACGAGAAGCTGCGTCAAACCGGTGATGAACT
GGTGCAGGCGTTCCAACGTCTGCGTGAGATCTTTGACAAGGGCGACGATGACAGCCTGGAA
CAGGTTCTGGAGGAAATCGAGGAACTGATTCAGAAACACCGTCAACTGGCGAGCGAGCTGC
CGAAGGGGGAACTGCAGGGTATTAAACAATACCGTGAGGCGCTGGAATATGTGAAGCTGCC
GGTTCTGGCGAAAATCCTGGAGGATGAAGAGAAGCATATTGAGTGGCTGAAGGAAGCGGCG
AAACAGGGTGATCAATGGGTGCAGCTGTTCCAACGTTTTCGTGAAGCGATCGACAAGGGCG
ATAAAGACAGCCTGGAGCAGCTGCTGGAGGAACTGGAACAGGCGCTGCAAAAGATTCGTGA
GCTGACCGAAAAAACCGGTCGCAAGATTCTGGAGGACGAGGAAAAACACATTGAGTGGCTG
GAAACCATTCTGGGTTAA

DFP3

ATGCACCACCATCATCACCACGAGAACCTGTATTTCCAAGGCGATTACCTGCGCGAACTGCT
GAAGGGCGAACTGCAAGGGATCAAGCAGTACGAGAAGCTGCGTCAAACCGGTGATGAACT
GGTGCAGGCGTTCCAACGTCTGCGTGAGATCTTTGACAAGGGCGACGATGACAGCCTGGAA
CAGGTTCTGGAGGAAATCGAGGAACTGATTCAGAAACACCGTCAACTGGCGAGCGAGCTGC
CGAAGGGGGAACTGCAGGGTATTAAACAATACCGTGAGGCGCTGGAATATACCCACAACCC
GGTTCTGGCGAAAATCCTGGAGGATGAAGAGAAGCATATTGAGTGGCTGAAGGAAGCGGCG
AAACAGGGTGATCAATGGGTGCAGCTGTTCCAACGTTTTCGTGAAGCGATCGACAAGGGCG
ATAAAGACAGCCTGGAGCAGCTGCTGGAGGAACTGGAACAGGCGCTGCAAAAGATTCGTGA
GCTGACCGAAAAAACCGGTCGCAAGATTCTGGAGGACGAGGAAAAACACATTGAGTGGCTG
GAAACCATTCTGGGTTAA

The expressed protein sequences were finally:

DFP1

MHHHHHHENLYFQ/GDYLRELLKLELQAIKQYEKLRQTGDELVQAFQRLREIFDKGDDDSLEQVL
EEIEELIQKHRQLASELPKLELQAIKQYREALEYVKLPVLAKILEDEEKHIEWLKEAAKQGDQWVQ
LFQRFREAIDKGDKDSLEQLLEELEQALQKIRELTEKTGRKILEDEEKHIEWLETILG

DFP2

MHHHHHHENLYFQ/GDYLRELLKGELQGIKQYEKLRQTGDELVQAFQRLREIFDKGDDDSLEQV
LEEIEELIQKHRQLASELPKGELQGIKQYREALEYVKLPVLAKILEDEEKHIEWLKEAAKQGDQWV
QLFQRFREAIDKGDKDSLEQLLEELEQALQKIRELTEKTGRKILEDEEKHIEWLETILG

DFP3

MHHHHHHENLYFQ/GDYLRELLKGELQGIKQYEKLRQTGDELVQAFQRLREIFDKGDDDSLEQV
LEEIEELIQKHRQLASELPKGELQGIKQYREALEYTHNPVLAKILEDEEKHIEWLKEAAKQGDQWV
QLFQRFREAIDKGDKDSLEQLLEELEQALQKIRELTEKTGRKILEDEEKHIEWLETILG

where the "/" defines the cleavage site of TEV protease.

Their sequences were confirmed (Genewiz San Francisco), and they were then transformed and expressed in One Shot BL21(DE3) chemically competent *E. coli* (Thermo Fisher Scientific). All the three proteins were expressed and purified as follows. BL21(DE3) cells were grown in LB broth (100 mg ml^{-1} kanamycin) to optical density (OD) 0.6 - 0.8 at 37 °C, then induced by addition of 1 mM Isopropyl β-D-1-thiogalactopyranoside (IPTG) and grown for 4 hours (1L per construct). Cells were harvested by centrifugation and resuspended in 60 mL of buffer A (50 mM HEPES pH 7, 100 mM NaCl, 20 mM imidazole) lysed by sonication at 4 °C, and then centrifuged at 18,000 g at 4 °C for 25 min. The supernatant was applied to 2.5 mL of Ni-NTA beads pre-washed with buffer A and the His-tagged protein were eluted with buffer B (50 mM HEPES pH 7, 100 mM NaCl, 300 mM imidazole). The His-tag was subsequently cleaved with a His-tagged TEV protease in buffer C (50 mM TrisHCl pH 8, 0.5 mM EDTA, 1 mM DTT), incubated overnight at 4 °C. The unreacted His-tagged DFP protein and the His-tagged TEV protease were removed by application of 2.5 mL of Ni-NTA beads pre-washed with buffer C.
Finally, the proteins were exchanged in the final buffer containing 50 mM HEPES 100 mM NaCl pH 7.

The monomeric forms were purified by SEC on an AKTA FPLC (GE) fitted with a Superdex 75 Increase 10/300 or a Superdex 200 Increase 10/300 and eluted at 4°C with HEPES (50 mM, pH 7)/NaCl (100 mM) buffer solution, at a 0.3 mL/min flow rate.
The ZnP-DFP complexes were prepared adding a 2-fold excess of the porphyrin from a 12 mM dimethylsulfoxide (DMSO) stock solution to a 50 mM HEPES, 100 mM NaCl, pH 7 buffer with *apo*-DFP protein (the final DMSO concentrations were kept to <5%). The solution was incubated for 15 min at 70°C and subsequently filtered before injection, to obtain the monomeric complex.

Pump-Probe Transient Absorption Spectroscopy

Ultrafast transient absorption experiments were performed following previously reported methods (2).

DFP1 crystallization and structure determination

To prepare the doubly loaded di-Zn^{2+} -ZnP-DFP1, Chelex 100 was used first to remove all bound metal ions to apo-DFP1. The di-Zn^{2+} -ZnP-DFP1 was then reconstituted from the metal ion-free form as previously described and in excess of Zn^{2+}. Finally, $ZnCl_2$ was added to an additional free Zn ion concentration of 100 μM.
The protein was crystallized by vapor-diffusion hanging-drop method at 25ºC, with a 1:1 (v/v) mixture of protein solution (5mg/mL in 100 mM NaCl, 100 mM HEPES, pH 7.0 and 100 μM $ZnCl_2$) and reservoir solution (22% wt/vol PEG 4000, 100 mM MgCl2, 100 mM Hepes, pH 7.0) equilibrated against the reservoir solution.

The crystals were frozen in liquid nitrogen, and the diffraction data were collected at 100K at the Beamline 8.3.1 of the Advanced Light Source (Berkeley, CA). The X-ray wavelength was 1.11583 Å. The data were processed with XDS (3). The structure was solved by molecular replacement with Phaser using the designed model as a search model (4). There exists four porphyrin-bound helical bundles in the crystallographic asymmetric unit. The structure refinement was done with REFMAC (5). In particular, rigid-body refinement was heavily performed with rigid-body domains down from single helical bundles to single helices. The restrained refinement was

then done, with TLS refinement combined at late stage; for TLS refinement the TLS domains were set down to single helical bundles. During the restrained refinement, the non-crystallographic symmetry restraints were applied among the four bundles. The software COOT was used for structural model adjustments (6). The data processing and structural refinement statistics were shown in Table S1.

The superimposition of the single domains on DFP1 was performed considering only Cα atoms. We considered the total symmetric di-Mn^{2+}-DF1 x-ray structure of (PDB ID: 1JMB), first NMR structure of ZnP-PS1 ensemble (PDB ID: 5TGY) and di-Zn^{2+}-ZnP-DFP1 chain D (PDB ID: 7JH6). The ranges considered for the different metalloproteins were:

- di-Mn^{2+}-DF1: 1-17 and 31-48 chain A and 5 and 43 chain B
- ZnP-PS1: 6-46 and 69-101
- di-Zn^{2+}-ZnP-DFP1: 2-18,67-105 and 158-175 for DF domain and 21-64 and 108-151 for PS1 domain

Synthesis of Fmoc-His-OH (P)

Fmoc-His(trt)-OH (**R**, 0.620 g, 1 mmol) was dissolved in the side chain deprotection solution of 95% TFA, 2.5% TIS, 2.5% H_2O (v/v/v 10 mL). The reaction was carried out for one hour at 0 ° C and one hour at room temperature, under a slight magnetic stirring. The TFA was removed by rotary evaporation to give an oily residue. The latter was dissolved in 10 mL of cold acetonitrile and the final product **P** was purified by crystallization.

The product was analyzed by analytical RP-HPLC, performed with a Shimadzu LC-10ADvp equipped with a SPDM10Avp diode-array detector. ESI-IT/TOF spectra were recorded on a Shimadzu LCMS-IT-TOF system with ESI interface and Shimadzu LC-MS solution Workstation software for the data. The analysis was performed with a Vydac C18 column (2.1 mm x 100 mm; 5µm), eluted with an H_2O 0.05 % trifluoroacetic acid, TFA, (eluent A) and CH3CN 0.05 % TFA (eluent B) in isocratic steps (30 % solvent B for 5 min, 50 % solvent B for 10 min) at 0.2 mL min^{-1} flow rate. The optimized MS parameters were selected as followed: CDL (curved desolvation line) temperature 250 °C; the block temperature 250 °C; the probe temperature 250 °C; detector gain 1.6kV; probe voltage +4.5kV; CDL voltage -15V. Nitrogen served as nebulizer gas (flow rate: 1.5 L min^{-1}).

The retention time of the starting **R** and the final **P** was 12.53 min and 2.65 min, respectively. In the purified product chromatogram, we did observe only the peak of the latter (purity >92 %). Identity was ascertained by high resolution ESI-MS: $[P+H^+]^+$=378.15 Th (theoretical 378.145); $[P+Na^+]^+$=400.129 Th (theoretical 400.127); $[P-H^++2Na^+]^+$=422.108 Th (theoretical 422.108).

Fmoc-His-OH – ZnP binding

ZnP was dissolved in DCM at a final concentration of 1µM. The maximum of the Soret band was at 415 nm, and we did not observe any peak in the CD spectrum, as expected. When Fmoc-His-OH and DIPEA were added at a final concentration of 2mM and 20mM, we observed a red shift of the Soret band from 415 nm to 423 nm, due to the coordination of the histidine to the zinc porphyrin. This shift was already observed upon coordination of DFP3 to ZnP solubilized in buffer in 1 % w/v octylglucopyranoside detergent. However, we did not observe any Cotton effect in the Fmoc-His-OH – ZnP complex.

Thermal denaturations

Melting curves were performed at a total protein concentration of 10 µM in 5 mM HEPES pH 7., in a 0.1 cm cell, using a J-815 spectropolarimeter equipped with a thermostated cell holder (JASCO, Easton, MD, USA). Thermal denaturations were obtained by monitoring the CD signal at 222 nm as a function of temperature from 20 to 100 °C. The temperature was raised with a constant ramp of 0.2 °C min^{-1}. Point were collected every 2.5 °C, with a data averaging of 32 s and 5 nm bandwidth. The thermal unfolding in presence of 4M Gdn-HCl was performed under the same experimental conditions. The melting temperatures were determined considering a two-state transition of a monomer between folded and unfolded forms, with correcting the data for pre- and post-transition linear changes in ellipticity as a function of temperature.(7)

MD simulations

The structure of porphyrin was taken from the crystal structure, and beta hydrogens were added manually in PyMOL. Porphyrin, without the coordinated zinc ion, was then parameterized using Antechamber and the general Amber force field (GAFF) (8) with a net -2 charge to form a neutral complex with the zinc ion. Structure preparation for the holo and apo states was then completed using the AmberTools18 program tleap. The protein was solvated in rectangular boxes with 1.5 nm padding of solvent on each side, and then solvent molecules were randomly replaced with ions to neutralize the system charge and reach 150 mM NaCl. The Amber ff14SB force field was used for the protein (9), with the TIP3P water model (10), and Li/Merz divalent ion parameters for zinc and iron in the porphyrin and active site, respectivelts (11).

Simulations were conducted using GPU-accelerated Amber18 (12). Systems were minimized with 2000 kJ/mol nm^2 restraints on protein and ligand heavy atoms and zinc ions, then heated from 100 K to 310 K over 25 ps in the NVT ensemble with the same restraints. Temperature was controlled using a Langevin thermostat. Five rounds of NPT each 300 ps long were then run, lowering the applied position restraints after each round: 2000, 1000, 400, 200, 20 kJ/mol nm^2. Final equilibration was run with no restraints for 1 ns. The Berendsen barostat was used to maintain pressure throughout NPT equilibration. Production was then run, using the Langevin thermostat with a Monte Carlo barostat, for 1 μs using a 2 fs timestep with coordinates saved every 50 ps.

A trial simulation of the porphyrin-bound state revealed that the zinc ion in the porphyrin distorted the protein structure by coordinating with the backbone oxygen of THR 152. A pair of distance restraints between the porphyrin zinc and the beta and carboxyl carbons of THR were applied to maintain normal structure of the protein backbone.

.

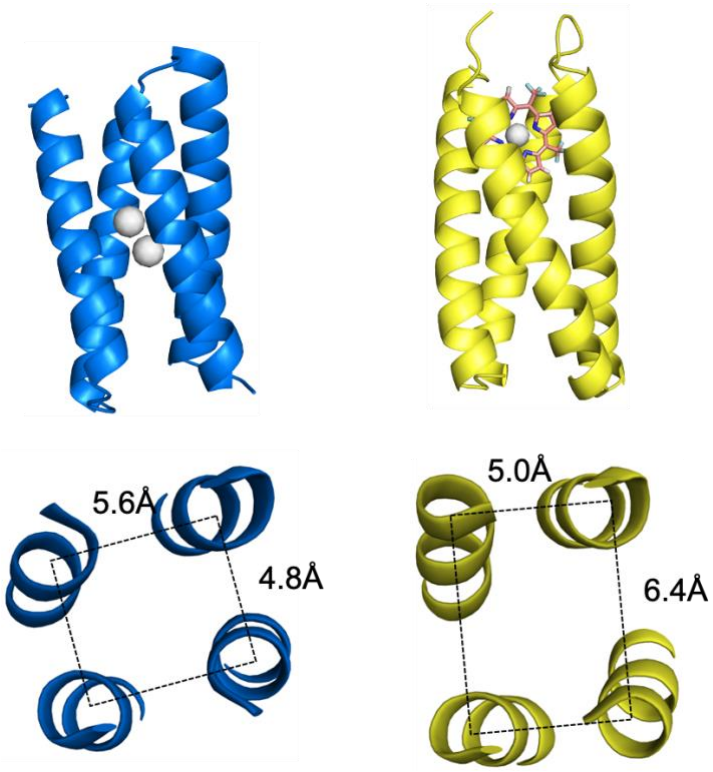

Fig. S1. Differences between helical bundle geometries. DF1 (left) and PS1 (right) are both 4-helix bundle proteins. The bundle geometry at the planes of connection between DF1 and PS1 (bottom left and right respectively) are substantially different from one another.

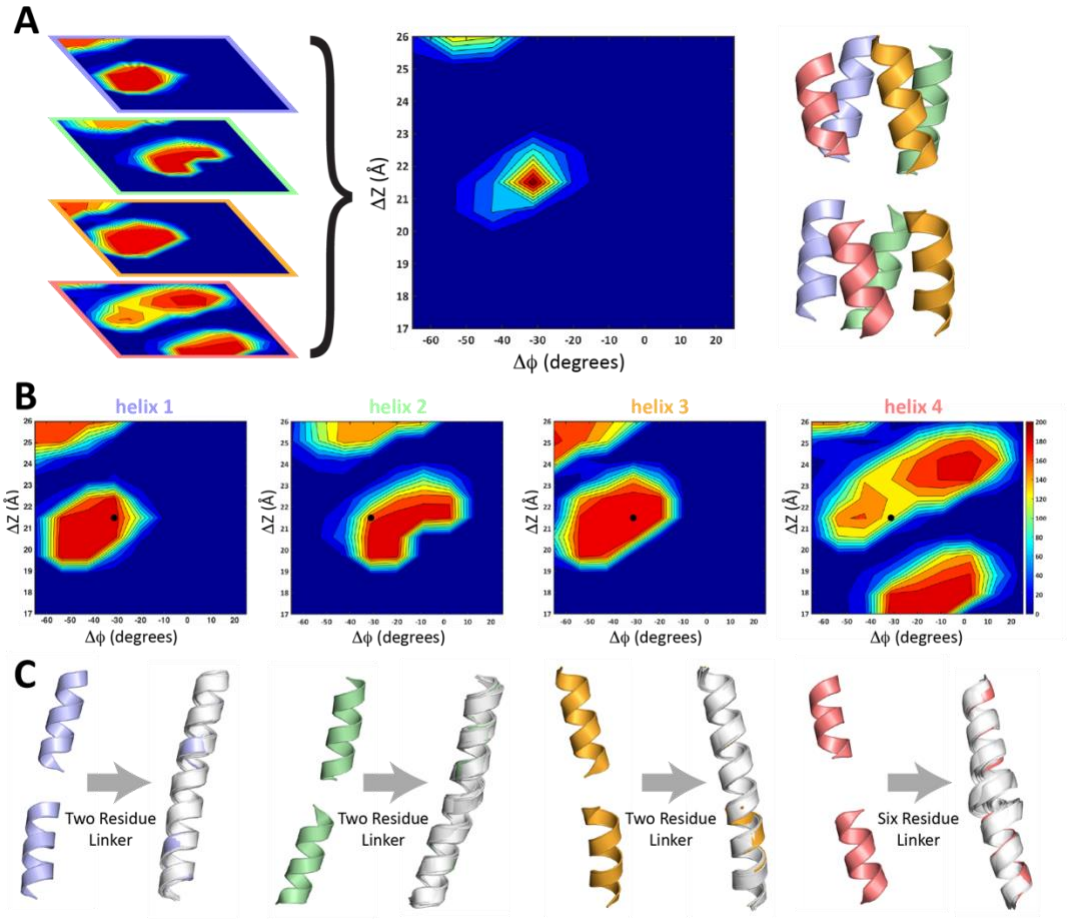

Fig. S2. Structural designability landscapes provide structural solutions to connect each disjointed helical pair. A. The bundle designability landscape (middle), resulting from the landscapes of the four disjointed helices (left, outlined by color to identify each corresponding helix), shows a single maximum corresponding to a specific placement of bundles (shown on the right). **B.** The four helix designability landscapes, with black dots marking the location of the bundle designability maximum from **A. C.** Structures of the four disjointed helix structures (colored) corresponding to the coordinates of the bundle designability maximum (black dots in B), and their top 10 structural matches (superimposed in white). The residue length of the linker necessary to connect the two helices is specified.

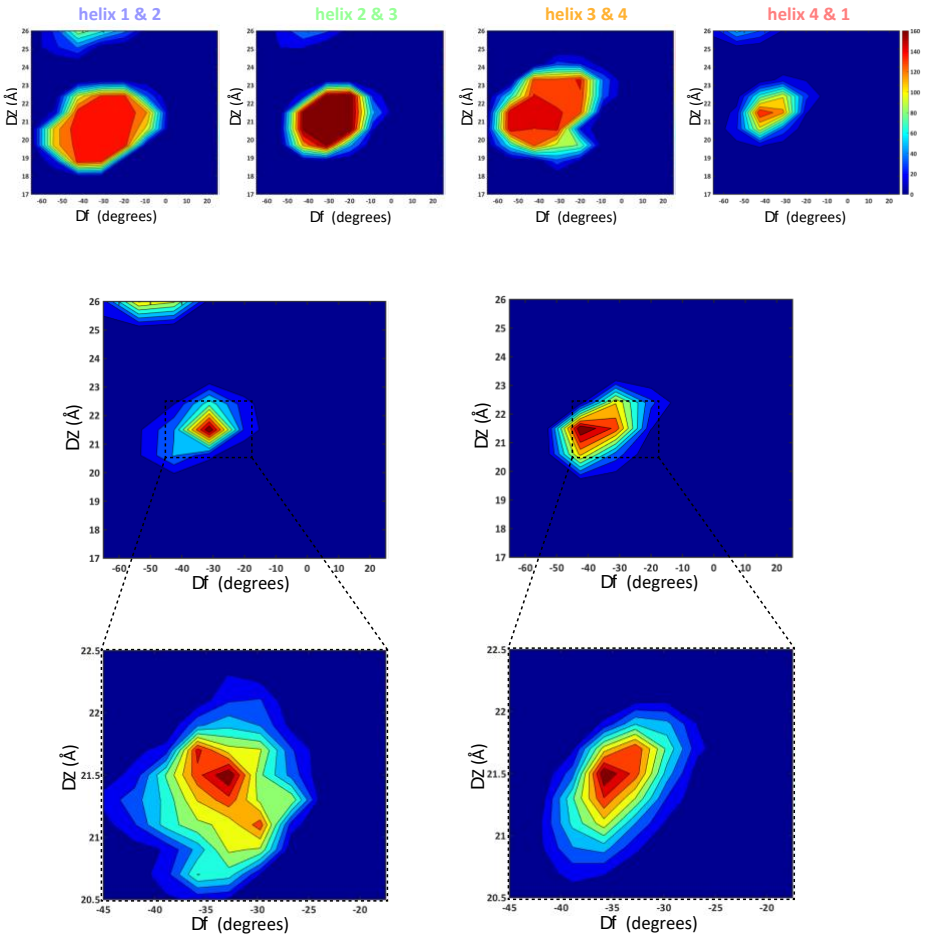

Fig. S3. Searches using adjacent bundle helices have similar designability landscapes as those from single helices. Top four contour plots show the designability landscapes from poses comprised of adjacent helices (1&2, 2&3, 3&4, and 4&1), where the scale to the far left relates number of matches to contour plot color. The bundle designability landscape for adjacent helices on the bottom right is highly similar to bundle designability landscape for single helices on the bottom left. The two bottom contour plots are from single helix (left) and adjacent helices (right) searches using small ΔZ and $\Delta\Phi$ steps of 0.2 Angstroms and 0.5°, respectively over a smaller region in the designability landscape.

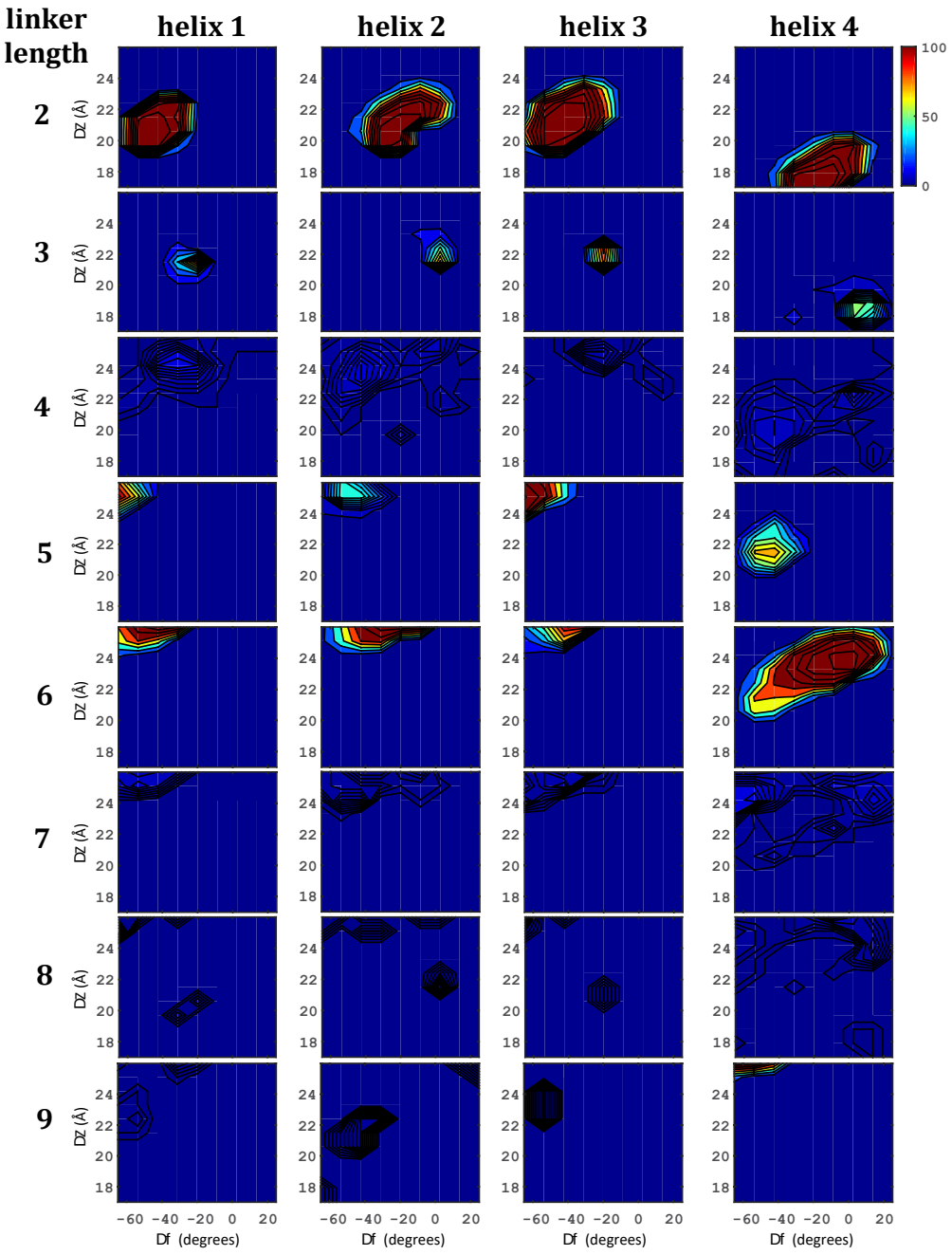

Fig. S4. Designability landscapes for the four disjointed helices separated out by connecting linker length. Helices 1-3 have a large designability hotspot for 2-residue linker lengths at roughly the same ΔZ, $\Delta\Phi$ coordinates, whereas the 2-residue linker hotspot for helix 4 is in an entirely different location in its corresponding landscape. However, the helix 4 designability landscapes for the 5-residue and 6-residue linkers have hotspots that overlap with the hotspots of helices 1-3 with 2-residue linker lengths. A bundle placement at (ΔZ, $\Delta\Phi$) = (21.5Å, -33.8°), with 2-residue linkers for helices 1-3 and 6-residue linker for helix 4 is the best option over the range of ΔZ, $\Delta\Phi$ considered here. Scale in upper right relates number of matches to contour plot color.

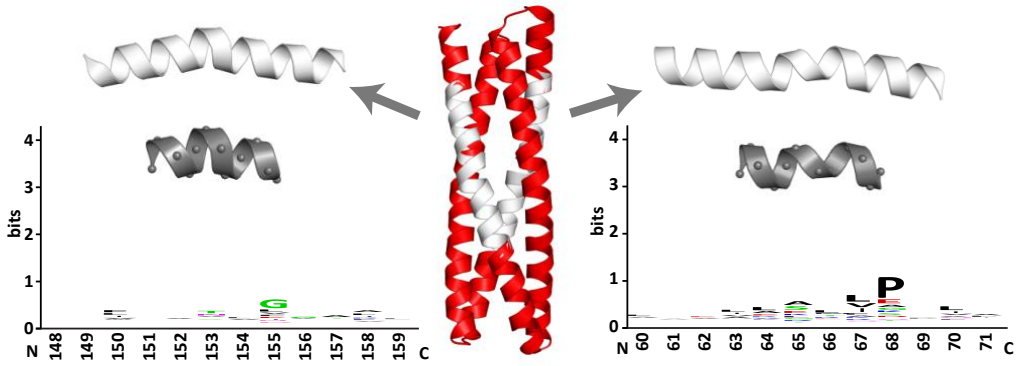

Fig. S5. Sequences of structural fragment matches from MASTER searches. Cartoon model of DFP1 (center) with structural fragments used to connect Helix 4 (left) and Helix 2 (right). Sequence logos over these 12-residue regions show the π-bulge in Helix 4 places sequence constraints at residue positions around it, as does the 3-10 helical turn in Helix 2. These deviations to α-helical geometry help create a single helical bundle. Sequence Logos for Helices 1&3 (not shown) are uninformative because their continuous α-helical geometries are highly designable.

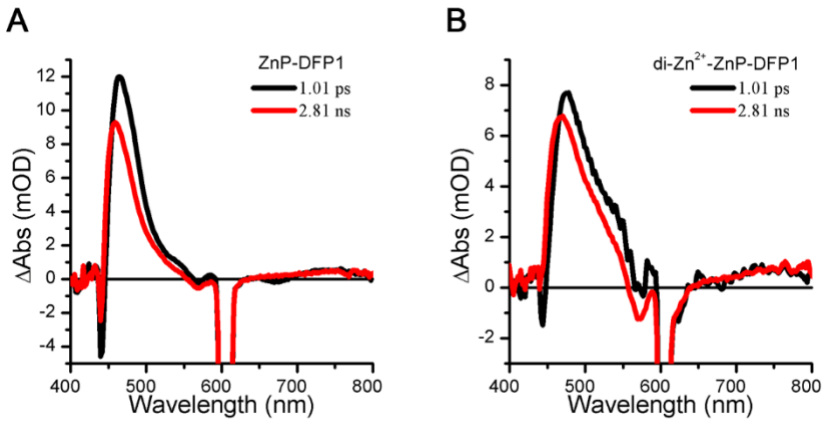

Fig. S6. Ultrafast pump-probe spectroscopy of DFP1. A. Pump-probe transient absorption spectra acquired for of ZnP bound in the interior of DFP1 [ZnP-DFP1] following excitation at 600 nm. The ps and ns time scale transient absorption spectra are characteristic of the respective $S_1 \rightarrow S_N$ and $T_1 \rightarrow T_N$ absorptions of the ZnP chromophore. **B.** Corresponding pump-probe transient absorption spectral data acquired for di-Zn^{2+}-ZnP-DFP1. **Experimental conditions:** solvent = 50 mM NaPi, 100 mM NaCl, pH 7.5; excitation wavelength = 600 ± 5 nm; magic-angle polarization between pump and probe pulses; pump–probe cross-correlation of ~250 fs.

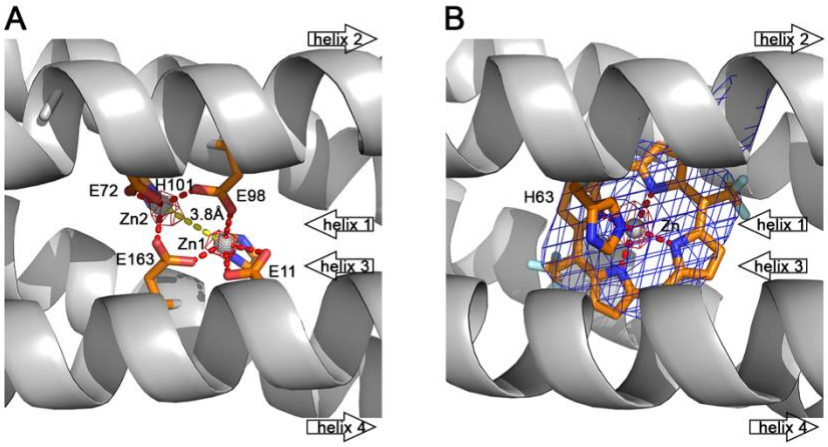

Fig. S7. DFP1 binds both metal cofactors $2F_o$-F_c electron density map of metal cofactors, **A.** di-Zn^{2+} and **B.** ZnP, bound to DFP1, contoured at 4.0 sigma and 1.0 sigma, respectively.

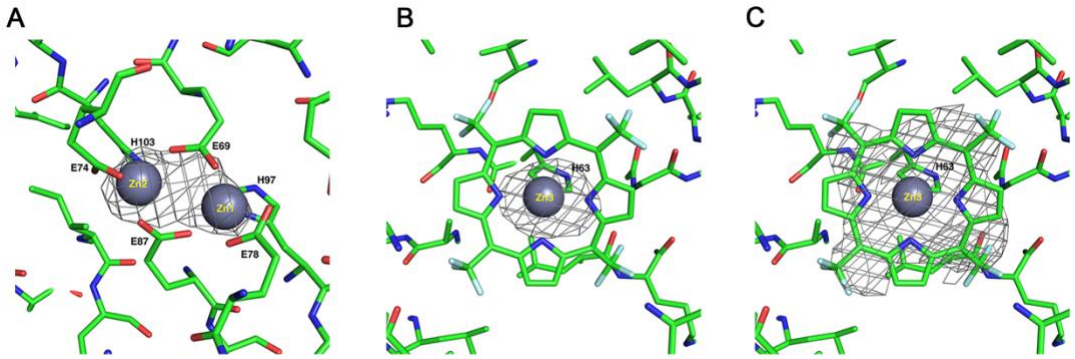

Fig. S8. The 2Fo-Fc omit maps of the di-Zinc cluster and the ZnP bound to DFP1. The omit maps for **A.** Zn1 and Zn2, **B.** Zn3 of ZnP, and **C.** ZnP are contouered at 3.0 sigma, 3.0 sigma and 1.0 sigma, respectively. To calculate the 2Fo-Fc omit maps, the zinc ions as well as ZnP were removed from the model, and the model was refined with simulated annealing to eliminate the bias before map calculation with PHENIX.

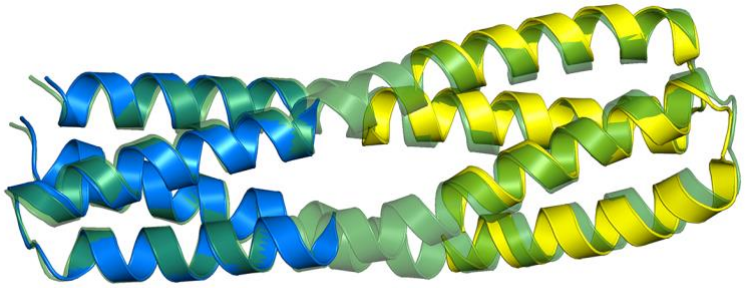

Fig. S9. Structures of di-Mn^{2+}-DF1 (PDB ID: 1JMB) and ZnP-PS1 (PDB ID: 5TGY), shown as solid blue and yellow cartoon, respectively, superimposed onto di-Zn^{2+}-ZnP-DFP1 (PDB ID: 7JH6, in green), shown as green cartoon in transparency. For di-Mn^{2+}-DF1 and ZnP-PS1, only the residues considered for the structural alignment are displayed.

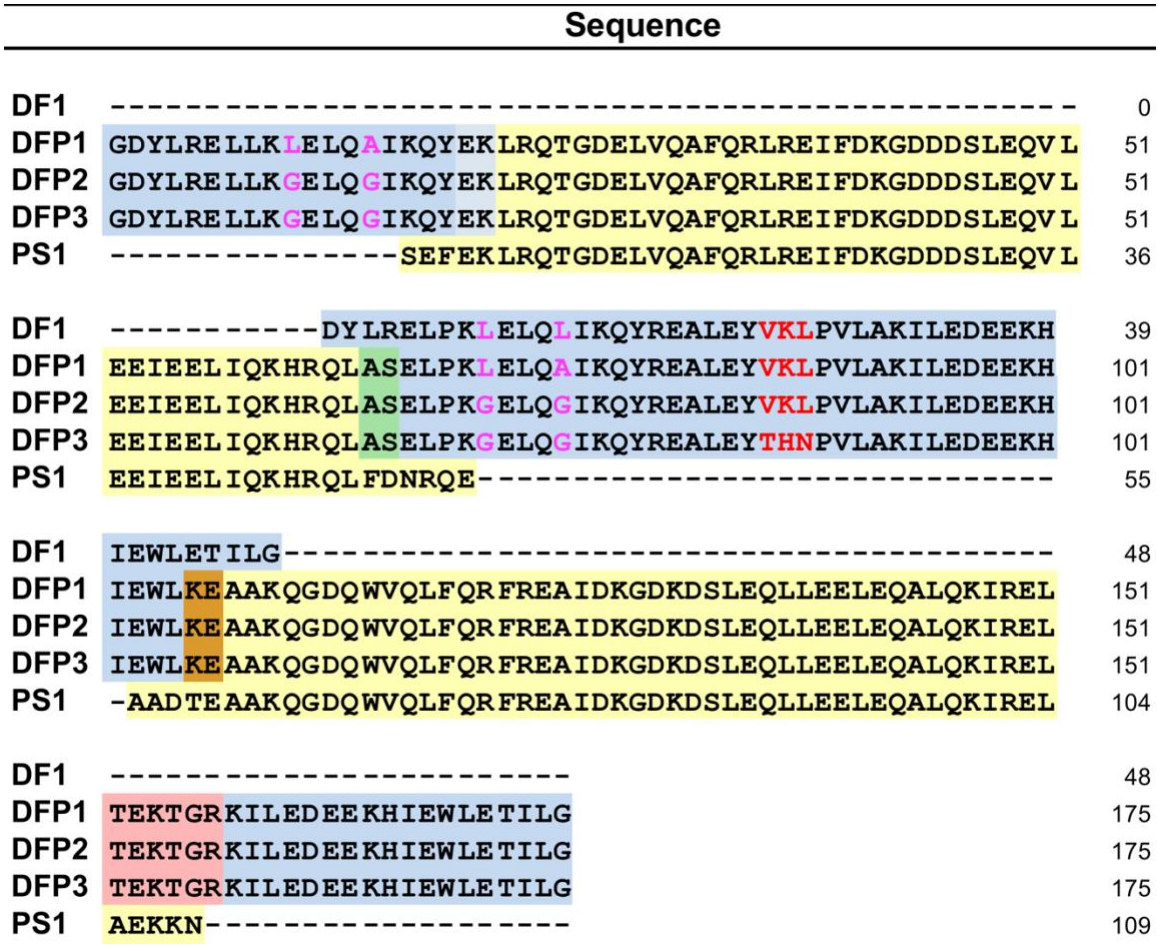

Fig. S10. Protein sequences of DFP analogues. DF1 and PS1 fragments are shown with the background in blue and yellow, respectively. The other color backgrounds represent the designed helical junctions. the residues lining the active site channel and at the loop of DF domain are represented in bold magenta and red, respectively.

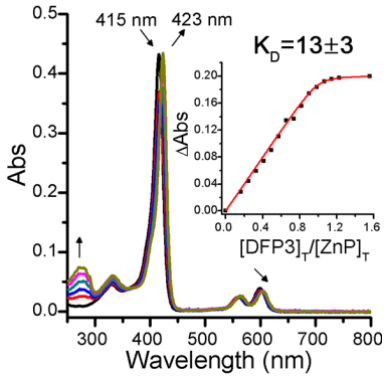

Fig. S11. UV-Vis spectral changes of a 2.2 μM $(CF_3)_4$PZn solution upon addition of *apo*-DFP3 in presence of zinc ions at pH 7 (HEPES 50 mM, NaCl 100 mM octyl-b-D-glucopyranoside 1% w/v). Inset: K_D determination of ZnP-DFP3 complex in presence of zinc ions.

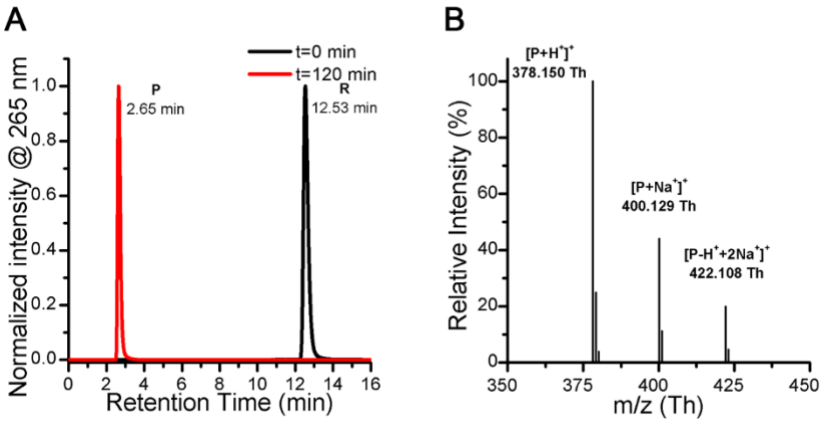

Fig. S12. A. LC-MS characterization of Fmoc-His-OH RP-HPLC chromatogram (265 nm trace) of the Fmoc-His(trt)-OH (**R**) deprotection reaction in Fmoc-His-OH (**P**) at time 0 min (in black) and 120 min (in red). **B.** ESI-TOF spectrum relative to the purified **P**.

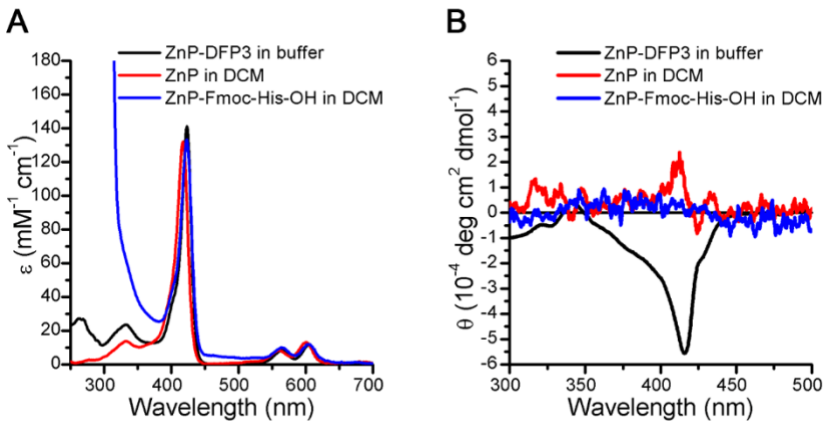

Fig. S13. A. ZnP Cotton effect arises from binding with the DFP3 UV-Vis spectrum of ZnP-DFP3 (in black) at pH 7 (HEPES 50 mM, NaCl 100 mM), ZnP (in red) in DCM and ZnP bound to Fmoc-His-OH in DCM. **B.** CD spectrum in the Soret region ZnP-DFP3 (in black) at pH 7 (HEPES 50 mM, NaCl 100 mM), ZnP (in red) in DCM and ZnP bound to Fmoc-His-OH in DCM.

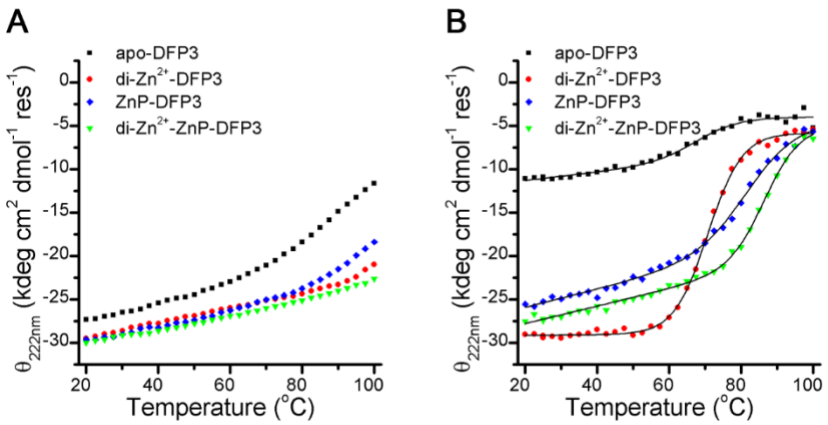

Fig. S14. DFP3 shows enhanced thermal stability Thermal denaturation curves of apo-DFP3 (squares), di-Zn^{2+}-DFP3 (circles), ZnP-DFP3 (blue triangles) and di-Zn^{2+}-ZnP-DFP3 (green triangles) at 10 μM protein concentration and pH 7 (HEPES 5 mM: **A.** in absence of Gdn-HCl and **B.** in presence of 4M Gdn-HCl. Only in presence of Gdn-HCl was possible to perform van't Hoff analysis (black line), considering a two-state transition.

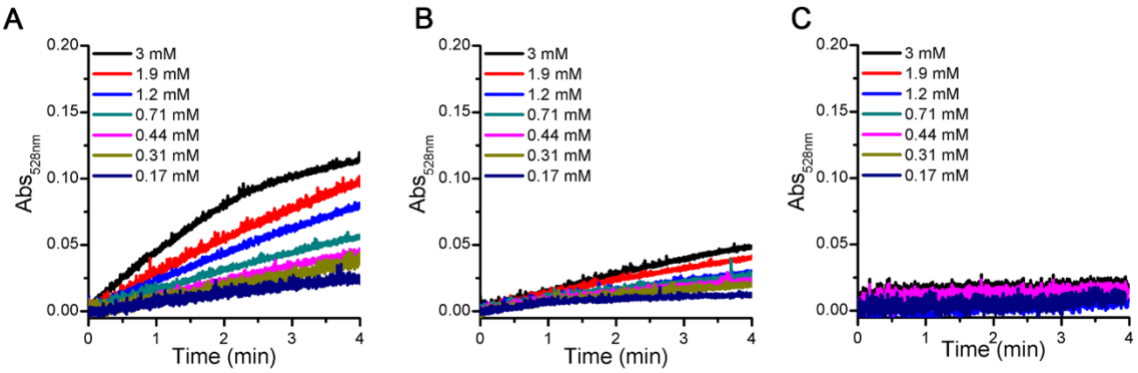

Fig. S15. Dependency of the initial rate the phenol oxidation in function of substrate concentration Kinetic curves of 4AP oxidation, at different substrate concentration, followed at 528 nm catalyzed by **A.** di-Fe^{3+}-DFP3 and **B.** di-Fe^{3+}-ZnP-DFP3, and **C**. in blank condition.

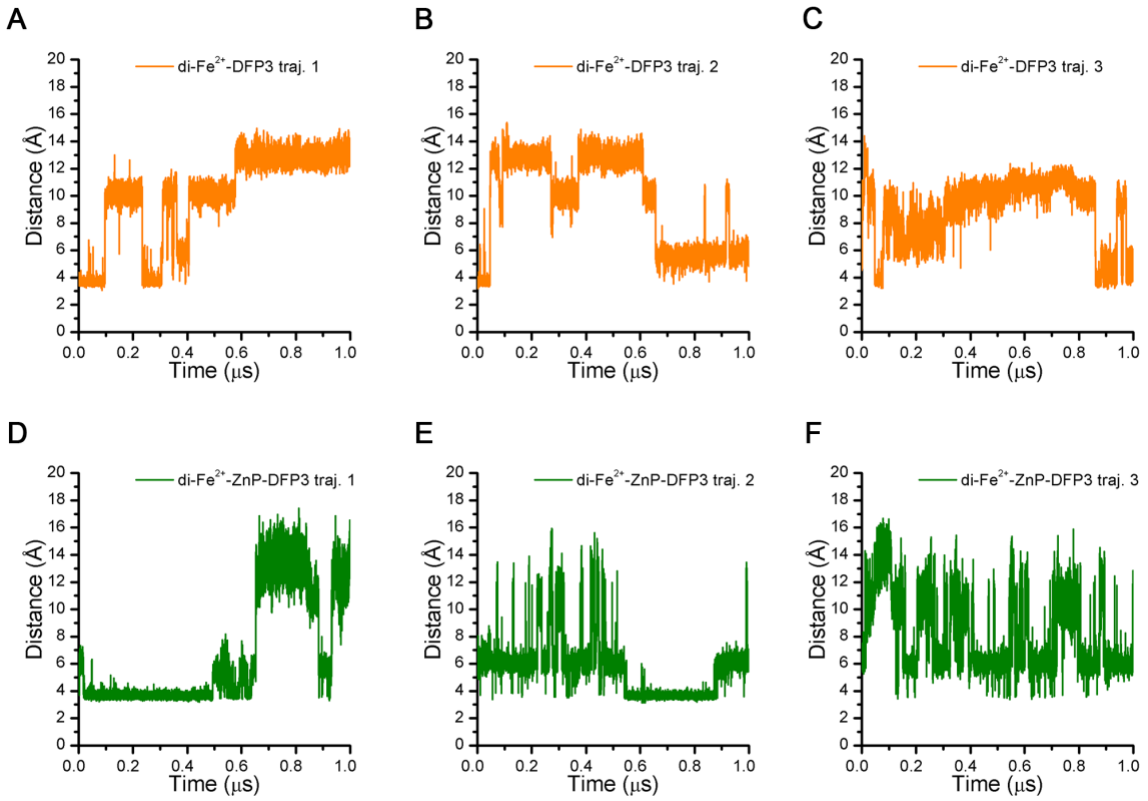

Fig. S16. Differences in Tyr18-Glu72 distance between di-Fe^{2+}-DFP3 and di-Fe^{2+}-ZnP-DFP3. Time series data of the Tyr18-Glu72 hydroxyl-carboxylate distance in the different trajectories of di-Fe^{2+}-DFP3 (**A**, **B** and **C**) and di-Fe^{2+}-ZnP-DFP3 (**D**, **E** and **F**), in orange and green respectively.

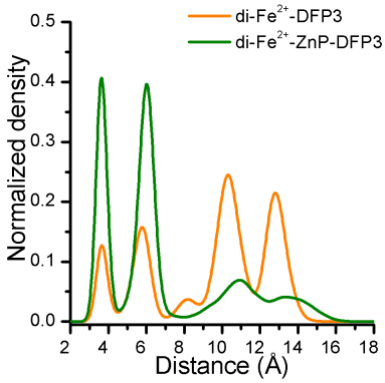

Fig. S17. Summed and renormalized population densities of the Tyr18-Glu72 hydroxyl-carboxylate distance across the three independent simulations of di-Fe^{2+}-DFP3 and di-Zn^{2+}-ZnP-DFP3, in orange and green respectively. A gaussian kernel estimator was used to smooth data from the trajectories.

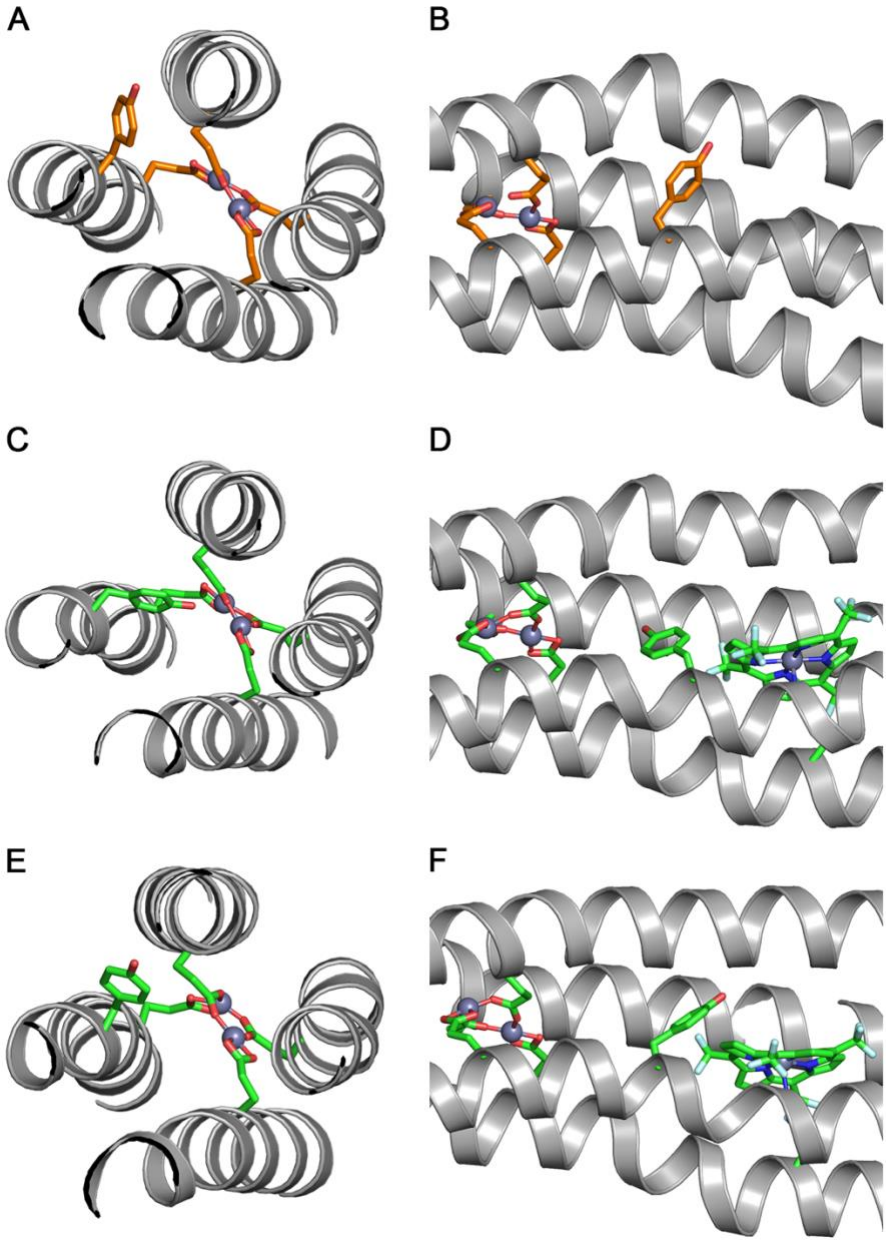

Fig. S18. Tyr18 shows conformational variability. In di-Zn^{2+}-DFP3, Tyr18 adopts a stable conformation, in which reorients its sidechain away from Glu72 to interact with residues on the protein surface: **A.** front and **B.** side views. In di-Zn^{2+}-ZnP-DFP3, Tyr18 sidechain assumes two different orientations: in the main conformation, Tyr18 maintains a water-bridged or direct interaction with Glu72 (**C.** front and **D.** side views); in the second one, Tyr18 transiently exposes its sidechain to the surface (**E.** front and **F.** side views). Zn^{2+} ions are represented as gray balls; active side residues are represented as orange or green sticks in di-Zn^{2+}-DFP3 (**A** and **B**) and di-Zn^{2+}-ZnP-DFP3 (**C** - **F**), respectively.

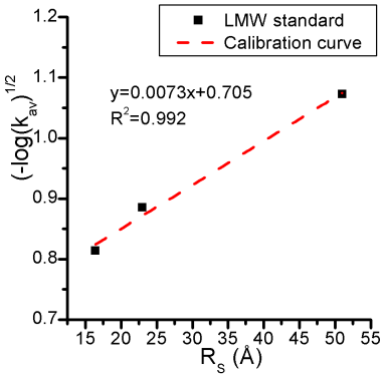

Fig. S19. Calibration curve with standard proteins: ribonuclease A, carbonic anhydrase and conalbumin

Table S1. Summary of the data collection and refinement statistics for DFP1

Wavelength	1.116
Resolution range	86.24 - 3.5 (3.625 - 3.5)
Space group	C2
Unit cell	172.478 27.825 188.494 90 117.162 90
Total reflections	66230 (6450)
Unique reflections	10708 (1008)
Multiplicity	6.2 (6.4)
Completeness (%)	99.63 (100.00)
Mean I/sigma(I)	6.49 (0.80)
Wilson B-factor	125.21
R-merge	0.1418 (2.184)
R-meas	0.1552 (2.379)
R-pim	0.06219 (0.931)
CC1/2	0.997 (0.322)
CC*	0.999 (0.698)
Reflections used in refinement	10692 (1008)
Reflections used for R-free	503 (52)
R-work	0.261
R-free	0.283
Number of non-hydrogen atoms	6081
macromolecules	5880
ligands	200
solvent	1
Protein residues	696
RMS(bonds)	0.005
RMS(angles)	0.97
Ramachandran favored (%)	96.37
Ramachandran allowed (%)	3.63
Average B-factor	154.5
macromolecules	155.0
ligands	138.5

Values in the parentheses are corresponding to the outmost resolution bin.
Randomly selected 5% of the reflections were omitted from refinement but used for R-free factor calculation.

Table S2. Stokes radius calibration curve parameters

	$\sqrt{(-\log(K_{av}))}$	R_S (Å)
LMW standards		
Ribonuclease A	0.81	16.4
Carbonic Anhydrase	0.88	23
Conalbumin	1.07	51
DFP proteins		
apo -DFP2 (R_T = 10.6min)	0.91	27 (21)
apo -DFP3 (R_T = 11.0min)	0.86	21 (21)

Values in the parentheses are corresponding to the theoretical value calculated with HYDRONMR, starting from the x-ray structure of DFP1

Code S1. *Bundle_aligner.py*

Begin Code:

```
import math
import numpy as np
from prody import *
from scipy.optimize import leastsq, minimize

class bundle_aligner:
    '''This Class defines all methods needed to input two PDB files of helical
    bundles, then align their best fit bundle axes along the Z-axis. The defined first
    chain in each bundle is placed on the X-axis. The script then offsets the
    second bundle relatvie to the first one in both the Z-direction (del_Z) and
    via rotation in the XY plane (del_angl).
        The user specifies the two pdb files, filename01, and filename02, as
    well as delZ in Angstroms and del_angl in degrees.
    '''

    def __init__(self, filename01, filename02):
        '''
        Parameters:
            filename01: First bundle to align
            filename02: Second bundle to align
            del_Z: amount to offset the second bundle relative to the first in
            Angstroms
            del_angl: Amount to rotate the second bundle in degrees
        '''
        self.bundle01 = filename01 #fetchPDB(filename01)
        self.bundle02 = filename02 #fetchPDB(filename02)

    def align_zaxis(self, chains01, chains02, residues01, residues02):
        '''
        This function aligns both bundle axes along the Z-axis
        Parameters:
            residues01: An n by 2 array consisting of
            [[start_residue_(1) end_residue(1)],...[[start_residue(n)
            end_residue(n)]] where n is the number of helices in bundle01
            residues02: Same as residues01 for bundle02
            chains01: A 1 by n array consisting of
            [chain(1) ... chain(n)] where n is the number of helices in bundle01
            chains02: A 1 by n array consisting of
            [chain(1) ... chain(n)] where n is the number of helices in bundle02

        Note that the number of rows in the array, n, must equal helices
        '''

        self.p01 = parsePDB(self.bundle01)
        self.p02 = parsePDB(self.bundle02)
        num_res = np.absolute(residues01[0][0] - residues01[0][1]) + 1
        helices01 = np.array(np.zeros((len(chains01), num_res, 3)))
        helices02 = np.array(np.zeros((len(chains02), num_res, 3)))
        fit01 = np.array(np.zeros((len(chains01),6)))
        fit02 = np.array(np.zeros((len(chains02),6)))
        a01 = np.array(np.zeros((num_res-4, 3)))
        a02 = np.array(np.zeros((num_res-4, 3)))
        AxisPoints01 = np.array(np.zeros((len(chains01), num_res, 3)))
        AxisPoints02 = np.array(np.zeros((len(chains01), num_res, 3)))
        h01 = []
        h02 = []
        v01 = []
        v02 = []
        z_axis = []
        helices01 = np.array(np.zeros((len(chains01), num_res, 3)))
        BundleAxis01 = np.array(np.zeros((num_res, 3)))
        BundleAxis02 = np.array(np.zeros((num_res, 3)))
        Rot_01 = np.array(np.zeros((3, 3)))
        Rot_02 = np.array(np.zeros((3, 3)))

        for m in np.arange(len(chains01)):
            atoms01 = self.p01.select(' '.join(['chain', chains01[m], 'ca', 'resnum',
```

```
                        str(residues01[m][0]), ':', str(residues01[m][1] + 1)]))
            atoms02 = self.p02.select(' '.join(['chain', chains02[m], 'ca', 'resnum',
                        str(residues02[m][0]), ':', str(residues02[m][1] + 1)]))
            ''' Get XYZ coords for the CA positions'''
            helices01[m, :, :] = atoms01.getCoords()
            helices02[m, :, :] = atoms02.getCoords()

            ''' Find the helical axis for each helix by fitting the CA coords
            to a cylinder with axis defined by unit vector a = (ax, ay, az) and
            point r0 = (x0, y0, z0)'''

            ''' cons constrains the vector a to be a unit vector that is
            perpendicular to r0, np.dot(a, r0) = 0 '''
            cons = [{'type': 'eq', 'fun': cons1},
                    {'type': 'eq', 'fun': cons2}]

            ''' Determine initial guess for a as the average of vectors between
            Calpha_i and Calpha_i+4 for i = 1,..., num_res-4'''
            for p in np.arange(num_res)-4:
                a01[p, :] = helices01[m, p+4, :] - helices01[m, p, :]
                a02[p, :] = helices02[m, p+4, :] - helices02[m, p, :]
            a01_est = np.average(a01, axis=0)/np.linalg.norm(np.average(a01, axis=0))
            a02_est = np.average(a02, axis=0)/np.linalg.norm(np.average(a02, axis=0))

            ''' Determine initial guess for r0 by projecting all the Calpha
            atoms onto the plane perpendicular to a, then fit them to a circle
            and let the center of the circle be r0.'''
            h01 = np.dot(helices01[m, :, :], a01_est)
            h01_proj = helices01[m, :, :] - h01[:, np.newaxis]*a01_est
            h02 = np.dot(helices02[m, :, :], a02_est)
            h02_proj = helices02[m, :, :] - h02[:, np.newaxis]*a02_est
            r01 = leastsq(circle_fit, np.append(h01_proj[0, :], 2.3), args=(h01_proj,))
            r02 = leastsq(circle_fit, np.append(h02_proj[0, :], 2.3), args=(h02_proj,))

            ''' Finally fit the Calphas to a cylinder with axis a, and point r0
            on its axis. Note you must have at least 6 Calphas from a helix to
            use this fit.
                For example, the axis, a, and point, r0, for the mth helix in the
            first bundle are given by fit01 = [ax, ay, az, r0x, r0y, r0z]'''
            vars01 = np.array([a01_est[0], a01_est[1], a01_est[2], r01[0][0], r01[0][1],
r01[0][2]])
            vars02 = np.array([a02_est[0], a02_est[1], a02_est[2], r02[0][0], r02[0][1],
r02[0][2]])
            F1 = minimize(cylinder_fit, vars01, args = (helices01[m,:,:],),
constraints=cons, method='SLSQP')
            fit01[m, :] = F1.x
            F2 = minimize(cylinder_fit, vars02, args = (helices02[m,:,:],),
constraints=cons, method='SLSQP')
            fit02[m, :] = F2.x
            ''' Project the Calpha positions in the helix onto the helical
            axis that was just calculated.'''
            AxisPoints01[m, :, :] = ClosestPointOnLine(fit01[m, :], helices01[m, :, :])
            AxisPoints02[m, :, :] = ClosestPointOnLine(fit02[m, :], helices02[m, :, :])
            ''' Calculate the bundle axis by averaging the m Axispoints for each
            layer (res_num) along the bundle. Let the first chain be defined as
            parallel to the bundle axis. Determine the orientation of each
            subsequent chain, n, in the bundle by taking the dot product of the
            vectors connecting the first and last Calpha in the first chain and
            the nth chain. If > 0 then keep data points as is, if < 0, then flip
            the data to make chains parallel.'''
            Dot01 = np.dot((AxisPoints01[0, 0, :] - AxisPoints01[0, num_res-1, :])
            , (AxisPoints01[m, 0, :] - AxisPoints01[m, num_res-1, :]))
            Dot02 = np.dot((AxisPoints02[0, 0, :] - AxisPoints02[0, num_res-1, :])
            , (AxisPoints02[m, 0, :] - AxisPoints02[m, num_res-1, :]))
            if Dot01 < 0:
                AxisPoints01[m, :, :] = np.flipud(AxisPoints01[m, :, :])
            if Dot02 < 0:
                AxisPoints02[m, :, :] = np.flipud(AxisPoints02[m, :, :])

        BundleAxis01 = np.average(AxisPoints01, axis=0)
        BundleAxis02 = np.average(AxisPoints02, axis=0)
```

```
        ''' Determine the best fit line for the bundle axis. The function
        LineFitXYZ takes array of data and returns a 1 by 6 array:
        [vx, vy, vz, R0x, R0y, R0z] describing the line:
        (vx*t + R0x, vy*t + R0y, vz*t + R0z)'''
        BundleAxis_line01 = LineFitXYZ(BundleAxis01)
        BundleAxis_line02 = LineFitXYZ(BundleAxis02)
        v01 = BundleAxis_line01[0:3]
        v02 = BundleAxis_line02[0:3]

        ''' Calculate rotation matrix necessary to align bundle axis to the
        z-axis, or [0, 0, 1]'''
        z_axis = [0, 0, 1]
        RotZ_01 = VectorAlign(v01, z_axis)
        RotZ_02 = VectorAlign(v02, z_axis)
        '''Apply the CM translation and the rotation matrix on the AxisPoints
        from the first chain of the bundle, to align them along the z-axis'''
        FirstHelix01 = (np.dot(RotZ_01, (AxisPoints01[0, :, :] -
BundleAxis_line01[3:6]).T)).T
        FirstHelix02 = (np.dot(RotZ_02, (AxisPoints02[0, :, :] -
BundleAxis_line02[3:6]).T)).T
        '''Determine the angle between the vector pointing from the origin to
        the first AxisPoint, then use it to calculate the rotation matrix
        necessary to rotate this point about the z-axis so that it lies on the x-axis'''
        ang01 = np.arctan2(FirstHelix01[0, 1], FirstHelix01[0, 0])
        ang02 = np.arctan2(FirstHelix02[0, 1], FirstHelix02[0, 0])
        Rot2X_01 = np.array([[np.cos(-ang01), -np.sin(-ang01), 0],
                             [np.sin(-ang01), np.cos(-ang01), 0],
                             [0, 0, 1]])
        Rot2X_02 = np.array([[np.cos(-ang02), -np.sin(-ang02), 0],
                             [np.sin(-ang02), np.cos(-ang02), 0],
                             [0, 0, 1]])
        '''Finally apply the CM translation and two Rot matrices to all atoms in
        the PDB file'''
        StructureCoords01 = self.p01.getCoords() - BundleAxis_line01[3:6]
        StructureCoords02 = self.p02.getCoords() - BundleAxis_line02[3:6]
        StructureCoords01 = np.dot(Rot2X_01, np.dot(RotZ_01, StructureCoords01.T)).T
        StructureCoords02 = np.dot(Rot2X_02, np.dot(RotZ_02, StructureCoords02.T)).T
        self.p01.setCoords(StructureCoords01)
        self.p02.setCoords(StructureCoords02)
        writePDB('Test_n1ck.pdb',self.p01)
        writePDB('Test_1jmb.pdb',self.p02)

    def offset_bundles(self, del_Z, del_angl, chains01, chains02, residues01,
residues02):

        Zoff_val = np.linspace(del_Z[0], del_Z[1], del_Z[2], endpoint=True)
        angl_val = np.linspace(del_angl[0], del_angl[1], del_angl[2],
endpoint=True)*np.pi/180

        dummy_p01, header_p01 = parsePDB(self.bundle01, header=True)
        dummy_p02, header_p02 = parsePDB(self.bundle02, header=True)
        id_number = 1
        for m in np.arange(len(Zoff_val)):
            for n in np.arange(len(angl_val)):
                RotXY = np.array([[np.cos(angl_val[n]), -np.sin(angl_val[n]), 0],
                                  [np.sin(angl_val[n]), np.cos(angl_val[n]), 0],
                                  [0, 0, 1]])
                newcoords = np.dot(RotXY,
                            (self.p01.getCoords() + [0, 0, Zoff_val[m]]).T).T
                dummy_p01.setCoords(newcoords)
                bundle01_coords = dummy_p01.select(' '.join(['chain',chains01[0],
'resnum',
                        str(residues01[0][0]), ':', str(residues01[0][1] + 1)]))
                bundle02_coords = self.p02.select(' '.join(['chain', chains02[0],
'resnum',
                        str(residues02[0][0]), ':', str(residues02[0][1] + 1)]))
                for p in np.arange(len(chains01)-1) + 1:
                    bundle01_coords = bundle01_coords + dummy_p01.select('
'.join(['chain', chains01[p], 'resnum',
                            str(residues01[p][0]), ':', str(residues01[p][1] + 1)]))
```

```
                    bundle02_coords = bundle02_coords + self.p02.select('
'.join(['chain', chains02[p], 'resnum',
                            str(residues02[p][0]), ':', str(residues02[p][1] + 1)]))
                writePDB('temp01.pdb', bundle01_coords)
                writePDB('temp02.pdb', bundle02_coords)
                p = parsePDB('temp01.pdb')
                q = parsePDB('temp02.pdb')
                i = 65 + len(chains01) + len(chains02)
                for j in np.arange(p.getHierView().numChains()):
                    chids01 = p.select(' '.join(['chain', np.unique(p.getChids())[0]]))
                    chids01 = chids01.setChids(chr(i))
                    i += 1
                i += 10
                for k in np.arange(q.getHierView().numChains()):
                    chids02 = q.select(' '.join(['chain', np.unique(q.getChids())[0]]))
                    chids02 = chids02.setChids(chr(i))
                    i += 1
                writePDB(str(self.bundle01)+'_'+str(self.bundle02)+
                                '_'+str(id_number)+
                                '_'+'delZ'+str(np.round(Zoff_val[m], decimals=1))
                                +'_'+'angl'+str(np.round(angl_val[n]*180/np.pi))
                                +'.pdb', p + q)
                id_number += 1


def cylinder_fit(vars, b):
    ax = vars[0]
    ay = vars[1]
    az = vars[2]
    rx = vars[3]
    ry = vars[4]
    rz = vars[5]
    m = np.dot(b, [ax, ay, az])
    d = b - [rx, ry, rz] - m[:, np.newaxis]*[ax, ay, az]
    return np.sum(np.sqrt((np.linalg.norm(d, axis=1) - np.average(np.linalg.norm(d,
axis=1)))**2))

def cons1(vars):
    return np.dot([vars[0], vars[1], vars[2]], [vars[3], vars[4], vars[5]])

def cons2(vars):
    return np.linalg.norm([vars[0], vars[1], vars[2]]) - 1

def circle_fit(vars, data):
    x = vars[0]
    y = vars[1]
    z = vars[2]
    r = vars[3]
    return (x - data[:,0])**2 + (y - data[:,1])**2 + (z - data[:,2])**2 - r**2

def ClosestPointOnLine(fit, p):
    a = [fit[0], fit[1], fit[2]]
    r0 = [fit[3], fit[4], fit[5]]
    proj = np.dot(p-r0, a)/np.dot(a, a)
    point = r0 + proj[:, np.newaxis]*a
    return point

def LineFitXYZ(data):
    datamean = np.average(data, axis=0)
    u, d, v = np.linalg.svd(data - datamean)
    return np.append(v[0], datamean)

def VectorAlign(a, b):
    v = np.cross(a, b)
    s = np.linalg.norm(v)
    c = np.dot(a, b)
    vx = np.array([[0, -v[2], v[1]], [v[2], 0, -v[0]], [-v[1], v[0], 0]])
    I = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1]])
    return I + vx + np.dot(vx, vx)/(1 + c)
```

```
BB = bundle_aligner('5tgy', '1jmb') #PDB IDs of the two bundles

del_Z = [17, 26, 37] #[Start Zdisp, End Zdisp, number of points]
del_angl = [-65, 25, 37] #[Start angle, End angle, number of points]
ch01 = ['A', 'A', 'A', 'A'] #Bundle 1 chains Helices [1, 2, 3, 4]
ch02 = ['B', 'C', 'C', 'B'] #Bundle 2 chains Helices [1, 2, 3, 4]
res01 = [[5, 16], [38, 49], [60, 71], [93, 104]] #Bundle 1 start to end residues on
Helices [1, 2, 3, 4]
res02 = [[5, 16], [5, 16], [31, 42], [31, 42]] #Bundle 2 start to end residues on Helices
[1, 2, 3, 4]
choff01 = ['A', 'A'] #Export coordinates for Bundle 1 chains - here just Helices [1, 2]
choff02 = ['B', 'C'] #Export coordinates for Bundle 2 chains - here just Helices [1, 2]
resoff01 = [[5, 16], [38, 49]] #Export coordinates for Bundle 1 residues - here just
Helices [1, 2]
resoff02 = [[5, 16], [5, 16]] #Export coordinates for Bundle 2 residues - here just
Helices [1, 2]
BB.align_zaxis(ch01, ch02, res01, res02)
BB.offset_bundles(del_Z, del_angl, choff01, choff02, resoff01, resoff02)
```

Code S2 . Rosetta XML Script for DFP1 helical bundle design

Begin code:

```
<ROSETTASCRIPTS>
	<SCOREFXNS>
		<hard_clean weights=talaris2014.wts />

		 <hard weights=talaris2014.wts >
			<Reweight scoretype=hack_aro weight=1 />
			<Reweight scoretype=rg weight=1 />
		</hard>

		<up_ele weights=talaris2014.wts >
			<Reweight scoretype=fa_elec weight=2.0 />
			<Reweight scoretype=hbond_sc weight=2.0 />
		</up_ele>

		<soft weights=soft_rep_design >
			<Reweight scoretype=hack_aro weight=1 />
			<Reweight scoretype=rg weight=1 />
		</soft>

		<hard_bb weights=talaris2014.wts >
			<Reweight scoretype=coordinate_constraint weight=2.0 />
			<Reweight scoretype=cart_bonded weight = 0.5 />
		</hard_bb>
	</SCOREFXNS>
	<RESIDUE_SELECTORS>
	</RESIDUE_SELECTORS>
	<TASKOPERATIONS>
		<ReadResfile name=resfile filename="DFP1.resfile" />
		<IncludeCurrent name=current />
		<LimitAromaChi2 name=arochi />
		<ExtraRotamersGeneric name=ex1_ex2 ex1=1 ex2=1 ex1_sample_level=3 ex2_sample_level=3
extrachi_cutoff=0 />
		<ExtraRotamersGeneric name=ex1 ex1=1 ex1_sample_level=3 />
		<RestrictToRepacking name=no_mutations />

		<LayerDesign name=all_layers layer=others make_pymol_script=0 >
			<CombinedTasks name=symmetric_interface_core>
				<SelectBySASA state=bound mode=mc core=1 probe_radius=2.0 core_asa=35
surface_asa=55 verbose=0 />
				<all copy_layer=core />
				<Helix append="ADEFGHIKLMNPQRSTVWY" />
			</CombinedTasks>

			<CombinedTasks name=symmetric_interface_boundary>
				<SelectBySASA state=bound mode=mc boundary=1 probe_radius=2.0 core_asa=35
surface_asa=55 verbose=0 />
				<all copy_layer=boundary />
				<Helix append="ADEFGHIKLMNPQRSTVWY" />
			</CombinedTasks>

			<CombinedTasks name=symmetric_interface_surface>
				<SelectBySASA state=bound mode=mc surface=1 probe_radius=2.0 core_asa=35
surface_asa=55 verbose=0 />
				<all copy_layer=surface />
				<Helix append="ADEFGHIKLMNPQRSTVWY" />
			</CombinedTasks>
		</LayerDesign>

		<SelectBySASA name=select_core state=bound mode=mc core=1 probe_radius=2.0 core_asa=35
surface_asa=55 verbose=0 />
		<SelectBySASA name=select_boundary state=bound mode=mc boundary=1 probe_radius=2.0 core_asa=35
surface_asa=55 verbose=0 />
```

```
		<SelectBySASA name=select_surface state=bound mode=mc surface=1 probe_radius=2.0 core_asa=35
surface_asa=55 verbose=0 />
	</TASKOPERATIONS>
	<FILTERS>
		<ShapeComplementarity name=sc_filter min_sc=0 verbose=0 jump=1 write_int_area=1 />
		<BuriedUnsatHbonds name=uhb scorefxn=hard_clean confidence=0 jump_number=1 cutoff=6 />
		<PackStat name=packstat threshold=0.58 confidence=0.0 repeats=3 />
		<ScoreType name=score_eval scorefxn=hard_clean threshold = 0 />
	</FILTERS>
	<MOVERS>
		<AddConstraintsToCurrentConformationMover name=add_cst use_distance_cst=0 max_distance=12.0
coord_dev=1.0 min_seq_sep=8 CA_only=1 />
		<MinMover name=hardmin_bb scorefxn=hard_bb type=lbfgs_armijo_nonmonotone tolerance=0.0001 chi=1
bb=1 bondangle=0 bondlength=0 jump=all />
		<ClearConstraintsMover name=clearconstraints />

		<Backrub name=rub_monomer pivot_residues=10-25,57-72,97-112,144-163 require_mm_bend=1
min_atoms=3 max_atoms=34 />
		<Sidechain name=sidechain task_operations=no_mutations />

		<PackRotamersMover name=softpack_core scorefxn=soft
task_operations=all_layers,resfile,select_core,current,arochi />
		<PackRotamersMover name=softpack_boundary scorefxn=soft
task_operations=all_layers,resfile,select_boundary,current,arochi />
		<PackRotamersMover name=softpack_surface scorefxn=soft
task_operations=all_layers,resfile,select_surface,current,arochi />

		<MinMover name=hardmin_sc scorefxn=hard chi=1 bb=0 bondangle=0 bondlength=0 tolerance=0.005/>

		<PackRotamersMover name=hardpack_core scorefxn=hard
task_operations=all_layers,resfile,select_core,current,arochi,ex1_ex2 />
		<PackRotamersMover name=hardpack_boundary scorefxn=hard
task_operations=all_layers,resfile,select_boundary,current,arochi,ex1_ex2 />
		<PackRotamersMover name=hardpack_surface scorefxn=up_ele
task_operations=all_layers,resfile,select_surface,current,arochi,ex1 />

		<ParsedProtocol name=backrub_protocol mode=single_random >
			<Add mover_name=rub_monomer apply_probability=0.75 />
			<Add mover_name=sidechain apply_probability=0.25 />
		</ParsedProtocol>

		<GenericMonteCarlo name=backrub_MC mover_name=backrub_protocol scorefxn_name=hard_clean
trials=200 temperature=1.2 preapply=0 />


		<ParsedProtocol name=monte_carlo_move mode=sequence >
			<Add mover_name=softpack_core />
			<Add mover_name=softpack_boundary />
			<Add mover_name=softpack_surface />
			<Add mover_name=hardmin_sc />
			<Add mover_name=hardpack_core />
			<Add mover_name=hardpack_boundary />
			<Add mover_name=hardpack_surface />
		</ParsedProtocol>

		<GenericMonteCarlo name=big_mover mover_name=monte_carlo_move scorefxn_name=hard_clean
temperature=0.4 trials=3 drift=1 preapply=false recover_low=1 />

	</MOVERS>
	<APPLY_TO_POSE>
	</APPLY_TO_POSE>
	<PROTOCOLS>

		<Add mover=add_cst />
		<Add mover=hardmin_bb />
		<Add mover=clearconstraints />
```

```
                <Add mover=softpack_core />
                <Add mover=softpack_boundary />
                <Add mover=softpack_surface />


                <Add mover=hardmin_sc />

                <Add mover=hardpack_core />
                <Add mover=hardpack_boundary />
                <Add mover=hardpack_surface />

                <Add mover=backrub_MC />
                <Add mover=big_mover />

                <Add filter=score_eval />
                Add filter=uhb />
                <Add filter=packstat />

        </PROTOCOLS>
        <OUTPUT />
</ROSETTASCRIPTS>
```

DFP1.resfile:

```
START

1 A PIKAA D
2 A PIKAA Y
3 A PIKAA L
4 A PIKAA R
5 A PIKAA E
6 A PIKAA L
7 A PIKAA L
8 A PIKAA K
9 A PIKAA L
10 A PIKAA E
11 A PIKAA L
12 A PIKAA Q
13 A PIKAA A
14 A PIKAA I
15 A PIKAA K
16 A PIKAA Y
17 A NOTAA C
18 A NOTAA C
19 A PIKAA K
20 A PIKAA L
21 A PIKAA R
22 A PIKAA Q
23 A PIKAA T
24 A PIKAA G
25 A PIKAA D
26 A PIKAA E
27 A PIKAA L
28 A PIKAA V
29 A PIKAA Q
30 A PIKAA A
31 A PIKAA F
32 A PIKAA Q
33 A PIKAA R
34 A PIKAA L
35 A PIKAA R
36 A PIKAA E
37 A PIKAA I
38 A PIKAA F
39 A PIKAA D
40 A PIKAA K
41 A PIKAA G
```

```
42 A PIKAA D
43 A PIKAA D
44 A PIKAA D
45 A PIKAA S
46 A PIKAA L
47 A PIKAA E
48 A PIKAA Q
49 A PIKAA V
50 A PIKAA L
51 A PIKAA E
52 A PIKAA E
53 A PIKAA I
54 A PIKAA E
55 A PIKAA E
56 A PIKAA L
57 A PIKAA I
58 A PIKAA Q
59 A PIKAA K
60 A PIKAA H
61 A PIKAA R
62 A PIKAA Q
63 A PIKAA L
64 A NOTAA CRKQW
65 A NOTAA C
66 A PIKAA E
67 A PIKAA L
68 A PIKAA L
69 A PIKAA K
70 A PIKAA L
71 A PIKAA E
72 A PIKAA L
73 A PIKAA Q
74 A PIKAA A
75 A PIKAA I
76 A PIKAA K
77 A PIKAA Q
78 A PIKAA Y
79 A PIKAA R
80 A PIKAA E
81 A PIKAA A
82 A PIKAA L
83 A PIKAA E
84 A PIKAA Y
85 A PIKAA V
86 A PIKAA K
87 A PIKAA L
88 A PIKAA P
89 A PIKAA V
90 A PIKAA L
91 A PIKAA A
92 A PIKAA K
93 A PIKAA I
94 A PIKAA L
95 A PIKAA E
96 A PIKAA D
97 A PIKAA E
98 A PIKAA E
99 A PIKAA K
100 A PIKAA H
101 A PIKAA I
102 A PIKAA E
103 A PIKAA W
104 A NOTAA C
105 A NOTAA C
106 A PIKAA E
107 A PIKAA A
108 A PIKAA A
109 A PIKAA K
110 A PIKAA Q
111 A PIKAA G
112 A PIKAA D
```

```
113 A PIKAA Q
114 A PIKAA W
115 A PIKAA V
116 A PIKAA Q
117 A PIKAA L
118 A PIKAA F
119 A PIKAA Q
120 A PIKAA R
121 A PIKAA F
122 A PIKAA R
123 A PIKAA E
124 A PIKAA A
125 A PIKAA I
126 A PIKAA D
127 A PIKAA K
128 A PIKAA G
129 A PIKAA D
130 A PIKAA K
131 A PIKAA D
132 A PIKAA S
133 A PIKAA L
134 A PIKAA E
135 A PIKAA Q
136 A PIKAA L
137 A PIKAA L
138 A PIKAA E
139 A PIKAA E
140 A PIKAA L
141 A PIKAA E
142 A PIKAA Q
143 A PIKAA A
144 A PIKAA L
145 A PIKAA Q
146 A PIKAA K
147 A PIKAA I
148 A PIKAA R
149 A PIKAA E
150 A PIKAA L
151 A NOTAA C
152 A NOTAA CRKQW
153 A NOTAA C
154 A NOTAA C
155 A NOTAA C
156 A NOTAA C
157 A PIKAA K
158 A PIKAA I
159 A PIKAA L
160 A PIKAA E
161 A PIKAA D
162 A PIKAA E
163 A PIKAA E
164 A PIKAA K
165 A PIKAA H
166 A PIKAA I
167 A PIKAA E
168 A PIKAA W
169 A PIKAA L
170 A PIKAA E
171 A PIKAA T
172 A PIKAA I
173 A PIKAA L
174 A PIKAA G
```

SI References

1. Åqvist J (1986) A simple way to calculate the axis of an α-helix. *Comput. Chem.* 10(2):97-99.
2. Polizzi NF*, et al.* (2016) Photoinduced Electron Transfer Elicits a Change in the Static Dielectric Constant of a de Novo Designed Protein. *J Am Chem Soc* 138(7):2130-2133.
3. Kabsch W (2010) Xds. *Acta Crystallogr D Biol Crystallogr* 66(Pt 2):125-132.
4. McCoy AJ*, et al.* (2007) Phaser crystallographic software. *J Appl Crystallogr* 40(Pt 4):658-674.
5. Kovalevskiy O, Nicholls RA, Long F, Carlon A, & Murshudov GN (2018) Overview of refinement procedures within REFMAC5: utilizing data from different sources. *Acta Crystallogr D Struct Biol* 74(Pt 3):215-227.
6. Emsley P, Lohkamp B, Scott WG, & Cowtan K (2010) Features and development of Coot. *Acta Crystallogr D Biol Crystallogr* 66(Pt 4):486-501.
7. Greenfield NJ (2006) Using circular dichroism collected as a function of temperature to determine the thermodynamics of protein unfolding and binding interactions. *Nat Protoc* 1(6):2527-2535.
8. Wang J, Wolf RM, Caldwell JW, Kollman PA, & Case DA (2004) Development and testing of a general amber force field. *J Comput Chem* 25(9):1157-1174.
9. Maier JA*, et al.* (2015) ff14SB: Improving the Accuracy of Protein Side Chain and Backbone Parameters from ff99SB. *J Chem Theory Comput* 11(8):3696-3713.
10. Jorgensen WL, Chandrasekhar J, Madura JD, Impey RW, & Klein ML (1983) Comparison of simple potential functions for simulating liquid water. *The Journal of Chemical Physics* 79(2):926-935.
11. Li P, Roberts BP, Chakravorty DK, & Merz KM, Jr. (2013) Rational Design of Particle Mesh Ewald Compatible Lennard-Jones Parameters for +2 Metal Cations in Explicit Solvent. *J Chem Theory Comput* 9(6):2733-2748.
12. Lee TS*, et al.* (2018) GPU-Accelerated Molecular Dynamics and Free Energy Methods in Amber18: Performance Enhancements and New Features. *J Chem Inf Model* 58(10):2043-2050.