Chapter 17

Redesigning the Specificity of Protein–DNA Interactions with Rosetta

Summer Thyme and David Baker

Abstract

Building protein tools that can selectively bind or cleave specific DNA sequences requires efficient technologies for modifying protein–DNA interactions. Computational design is one method for accomplishing this goal. In this chapter, we present the current state of protein–DNA interface design with the Rosetta macromolecular modeling program. The LAGLIDADG endonuclease family of DNA-cleaving enzymes, under study as potential gene therapy reagents, has been the main testing ground for these in silico protocols. At this time, the computational methods are most useful for designing endonuclease variants that can accommodate small numbers of target site substitutions. Attempts to engineer for more extensive interface changes will likely benefit from an approach that uses the computational design results in conjunction with a high-throughput directed evolution or screening procedure. The family of enzymes presents an engineering challenge because their interfaces are highly integrated and there is significant coordination between the binding and catalysis events. Future developments in the computational algorithms depend on experimental feedback to improve understanding and modeling of these complex enzymatic features. This chapter presents both the basic method of design that has been successfully used to modulate specificity and more advanced procedures that incorporate DNA flexibility and other properties that are likely necessary for reliable modeling of more extensive target site changes.

Key words Protein–DNA interactions, Computational design, Rosetta, Specificity, In silico prediction, Gene targeting, Direct readout

1 Introduction

Direct interactions between amino acids and DNA nucleotides are an important determinant of the substrate preference of a DNA-binding protein. A position in a binding site where the protein displays a preference for one nucleotide over the others is considered to have high specificity [1–3]. These positions are often characterized by strong direct interactions that are disrupted when the favored base is replaced (Fig. 1, *see* **Notes 1** and **2**). Being able to redesign interface residues to alter this specificity would enable the targeting of a DNA-binding protein to a site of interest (*see* **Note 3**). This technology is particularly useful for targeting genome-specific

David R. Edgell (ed.), *Homing Endonucleases: Methods and Protocols*, Methods in Molecular Biology, vol. 1123, DOI 10.1007/978-1-62703-968-0_17,

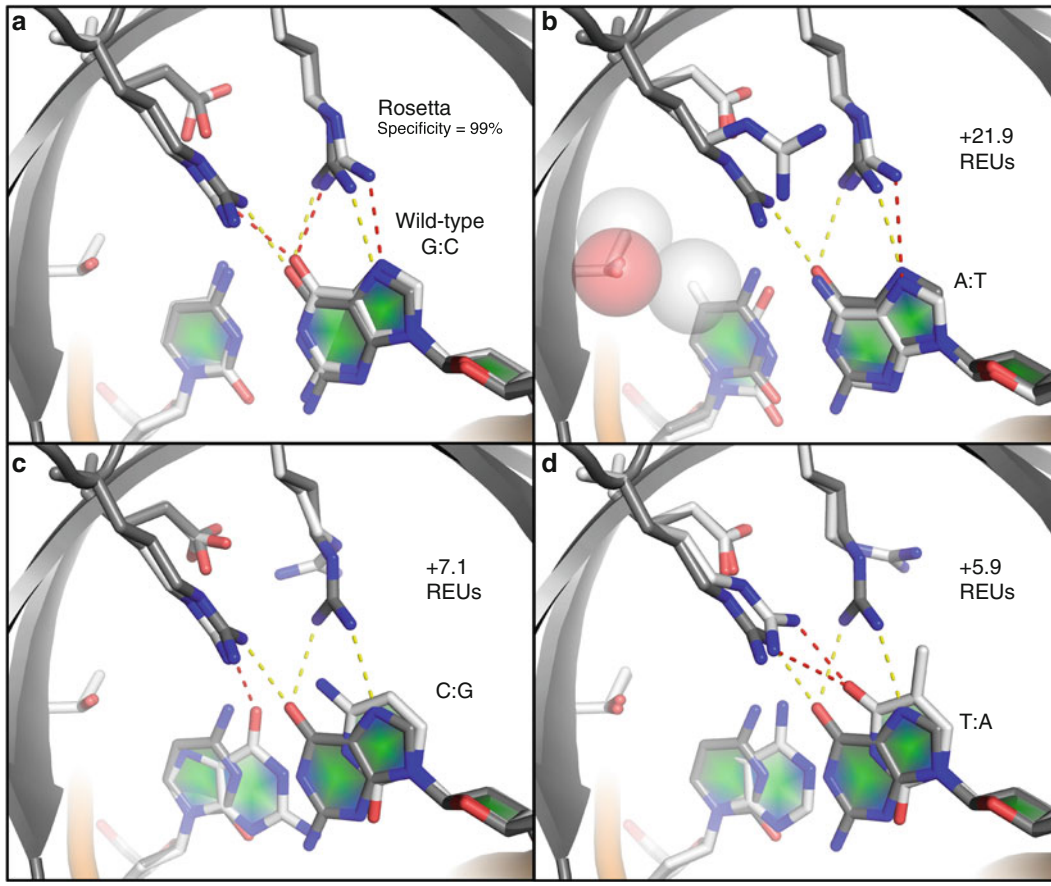

Fig. 1 The predicted role of direct interactions in protein–DNA specificity. Each panel represents the structure with the mean energy from a set of 56 repacks done with Rosetta. The wild-type base pair is the G:C at position −6 or 409 (crystal structure numbering) in the 2QOJ pdb. The native 2QOJ structure is shown in *gray*, with native hydrogen bonds shown in *yellow*, and the Rosetta structures are shown in *white*, with predicted hydrogen bonds shown in *red*. The wild-type base pair has a very high predicted specificity when compared to the three alternative base pairs. (**a**) The repacked structure for the wild-type base pair maintains the energetically favorable direct hydrogen bonds that are in the crystal structure. (**b**) The A:T base pair loses hydrogen bonding, and the methyl group of the thymine nucleotide has significant repulsion (highlighted by *spheres*) with a neighboring threonine side chain. (**c**) The C:G base pair loses hydrogen bonding. (**d**) The T:A base pair loses hydrogen bonding

DNA cleavage enzymes, such as LAGLIDADG endonucleases [4, 5], to sites that are relevant to genome engineering and gene therapy applications [6–9].

Computational methods for engineering specificity offer an efficient alternative to more labor-intensive experimental procedures, such as directed evolution [10–13]. Additionally, experimental and in silico approaches are not mutually exclusive, as the predicted results can also be used to guide the design of libraries for screening and enhance the likelihood of successfully identifying an active variant [14–16]. The Rosetta program for macromolecular modeling and design [17] has been successfully used to alter the specificity of

several LAGLIDADG homing endonucleases [18–21]. Describing the protocols used to computationally redesign endonuclease specificity with Rosetta is the focus of this chapter.

The main design algorithm in Rosetta searches protein sequence and rotameric [22] space to find a set of amino acids that is compatible with the DNA sequence being targeted (*see* **Note 4**). Each attempted amino acid combination is evaluated with a physically based energy function in order to identify the lowest-energy sequence [17, 23]. The majority of the previous endonuclease redesign successes [18–21] utilized a standard fixed-backbone algorithm, in which both the protein and DNA backbones in the starting crystal structure are not flexible. This chapter details this basic method and additionally introduces some alternative DNA interface design protocols. These advanced methods include flexibility on one or both sides of the protein–DNA interface [21, 24], explicit design for specificity using a genetic algorithm [19, 21], and the use of libraries of native-like interactions (called motifs) to guide rotamer sampling [23, 25]. These approaches provide ways to diversify design results over the fixed-backbone approximation available in release versions of Rosetta.

Computational design is an efficient approach for altering specificity, as long as the particular problem of interest is feasible with the currently available technology. The majority of the previously published successes were limited to single base-pair switches in the twenty base-pair target sequence characteristic of LAGLIDADG endonucleases [18–20]. The exception that stands out was design for a triple base pair. However, the crystal structure of this variant revealed extensive DNA movement and interface coordination that were not predicted by the standard Rosetta modeling [21]. Directed evolution methods have produced several large-scale specificity shifts, but achieving consistent success and maintaining the exquisite specificity characteristic of the natural endonucleases [19, 26] are still a challenge for every approach [27–29]. A structure obtained for one of these evolved enzymes also showed extensive interface rearrangements that are not considered in standard computational design protocols [27]. All of these results indicate that there are features of the LAGLIDAG interface that are not being accurately captured by the models.

LAGLIDADG endonucleases have highly integrated interfaces, in which binding and catalysis are coordinated [19]. While this characteristic is advantageous for a gene-targeting reagent, it significantly increases the challenge of specificity modulation because there is no currently understood recognition code ([30, 31], *see* **Note 5**). This lack of a recognition code makes computation even more necessary, albeit harder, because multiple base-pair specificity switches need to be engineered as one unit, instead of being engineered separately and then recombined. Directed evolution approaches are limited in how many amino acids can be

simultaneously randomized. One way to utilize the power of computational design is to identify variants with low levels of a desired feature and then use these proteins as starting points for directed evolution optimization [32–34]. Another use of computation is that it can suggest the inclusion of only certain amino acid types at each position in a protein library allows for many more positions to be concurrently explored. For example, the core positions that buttress the DNA-interacting residues and can be important for activity [35] are often excluded from libraries [11, 29] in favor of focusing on the interface residues that make direct contacts. A concerted approach is important because it is likely that a stringent alignment of the N- and C-terminal domains is required to facilitate catalysis and it is mediated by residues not in the protein–DNA interface [27, 28, 36]. There has been effectively no success at altering the specificity of the central four base pairs where catalysis occurs, presumably due to these alignment criteria and indirect readout of the DNA [37, 38], neither of which is modeled or well understood. Engineering pipelines that iterate between computational design, directed evolution, and detailed kinetic analyses are poised to discover the missing components of these computational models.

2 Materials

1. The latest release version of the Rosetta software suite (Rosetta 3.4 as of 2012) is available from http://www.rosettacommons.org and is free of charge for academics and nonprofit users. A comprehensive manual for the software is also available from the same website. For conducting protocols that are not included in the release version, the developer's version of the code must be obtained. Protocols that require these extended capabilities are noted throughout this chapter. A sponsor from a Rosetta lab is required for access to this repository, and a partial list of labs with members that can provide sponsorship or collaboration is included in **Note 6**.
2. Compiling the Rosetta code requires either an external compiling software or Python (version >= 2.2) to run the included scons.py script that runs a local version of the compiling software SCons that comes packaged with Rosetta.
3. The Rosetta software runs on multiple platforms (see manual for list). However, it is suggested that a Unix or Linux cluster be used in order to submit many runs in parallel and enhance calculation efficiency.
4. A high-resolution crystal structure (preferably <3.0 Å) of the protein of interest bound to DNA (*see* **Note 7**).

3 Methods

3.1 Standard Protein–DNA Interface Design

1. Obtain a current copy of the release version of Rosetta (*see* Subheading 2).
2. Open a terminal window (*see* **Note 8**).
3. Enter the Rosetta source directory that contains the scons.py file. Type "scons bin mode=release extras=static" to compile a production speed version of the code that can be ported to different platforms and computer systems (*see* **Note 9**).
4. If the code is going to be run on a different computer system than it was compiled, the rosettaDNA executable must be moved to that system by typing "scp ./bin/rosettaDNA.static.linuxgccrelease computerwhereitwillberun." The entire rosetta_database folder must also be moved by typing "scp -r ../rosetta_database/ computerwhereitwillberun."
5. Make a directory where the code will be run and the output collected by entering the desired location and typing "mkdir nameofdirectory" (*see* **Notes 10** and **11**).
6. Make a file that contains the arguments read by the Rosetta program with your favorite text editor (Fig. 2). The editor Vi is likely present in your Linux/Unix system. To use Vi to make the arguments file, type "vi nameofargsfile," enter insertion mode by typing "i," and then type the desired flags using Fig. 2 as a guide (*see* **Note 12**).
7. Make an XML script file (*see* **Note 13**) that contains protocol instructions given to the program through RosettaScripts [17, 39]. This file can be made by using Fig. 3 as a guide for the content and following the same Vi instructions described in **step 6** (all other files in future steps can also be made or modified with Vi).
8. The protein interface positions to be designed will be automatically calculated based on the "dna_defs" and "z_cutoff" flags that are part of the operations (TASKOPERATIONS) included in the XML file (Fig. 3). However, a type of file known as a resfile (Fig. 4) is available if the user would instead prefer to allow only a subset of amino acid types and designable positions. The addition of the line "-resfile nameoffile" to the args file (Fig. 2) will enable the resfile to override automatic detection of the interface residues. The XML script should also be modified to add the task operation "<ReadResfile name=RRF/>" and replace the use of AUTOprot with RRF in the mover. The "dna_def" option is also no longer necessary in the DnaInt operation because the target base is specified in the resfile.
9. Choose an energy function that is optimized for protein–DNA interactions [21–24] and make a file containing the necessary

```
-in:ignore_unrecognized_res # ignore anything in the pdb structure that is not recognizable
-file:s 2QOJ.pdb # input structure
-mute all # no output into an output file, skip this flag off when debugging and include for large-scale runs
-unmute protocols.dna # unmute a subset of the output if desired
-score::weights rosetta_database/scoring/weights/optimizedenergyfxn.wts # energy function for evaluating structures (see Fig. 5)
-score:output_residue_energies # include information in the pdb about the interaction energies of residues in the design
-run:output_hbond_info # include information in the pdb about the hydrogen bonding of residues in the design
-database rosetta_database # required Rosetta database, see Note 17 for useful changes to the database
-ex1 # extra rotamer sampling around chi angle 1
-ex2 # extra rotamer sampling around chi angle 2
-ex1aro::level 6 # even more extra rotamer sampling for aromatic residues around chi angle 1. This flag is recommended because aromatic residues can have large repulsion scores if the rotamer is not in the optimal position.
-ex2aro::level 6 # even more extra rotamer sampling for aromatic residues around chi angle 2
-exdna::level 4 # use DNA rotamers and include extra sampling (inclusion of this flag is highly advised for protein-DNA design)
-jd2:dd_parser # use the parser protocols
-parser:protocol XML.scriptfile # XML script (see Fig. 3)
-overwrite # if a pdb with the same name already exists in the directory where the design occurring, then overwrite the old pdb
-out:prefix design_ # an optional prefix to add to the name of designs
```

Fig. 2 Example arguments file. This file controls the parameters of the design run or specificity calculation. All writing after the # mark is a comment that is not read in by the Rosetta program

weights for each energy function component (Fig. 5). The name of the energy function is the input for the flag "-score::weights nameoffile" (Fig. 2).

10. Modify the Rosetta database to go with the optimized energy function shown in Fig. 5. The necessary changes are listed in **Note 14** [23].
11. Run code by submitting to whatever computer cluster you are using or by typing "rosettaDNA.static.linuxgccrelease @ nameofargsfile" (*see* **Notes 15** and **16**).

3.2 Assessment of Designs Using Specificity and Binding Energy Calculations

Follow instructions in Subheading 3.1 with the following described variations to the XML script (Fig. 3) and arguments files (Fig. 2). The specificity and binding energy calculations enable the user to identify the designs with the most desirable properties (*see* **Note 17**).

```
<dock_design>
 <TASKOPERATIONS>
  <InitializeFromCommandline name=IFC/> # use the information in the args file to supplement this XML
   <IncludeCurrent name=IC/> # includes the rotamers in the input structure (may not want to use)
   <RestrictDesignToProteinDNAInterface name= DnaInt base_only =1 z_cutoff =6.0 dna_defs =Z.409.GUA/> #
make the target site substitution of interest (chainID.crystalposition.type) and designate the sphere of residues
surrounding it that are designable and packable
     <OperateOnCertainResidues name=AUTOprot> # works with the DnaInt operation to enable residues to be
chosen for design and packing if they are marked as AUTO
    <AddBehaviorRLT behavior=AUTO/>
    <ResidueHasProperty property=PROTEIN/>
   </OperateOnCertainResidues>
  </TASKOPERATIONS>
  <SCOREFXNS>
   <DNA weights=optimizedenergyfxn/> # energy function for design evaluation, this file must be put in the
directory (ie, rosetta_database/scoring/weights/optimizedenergyfxn.wts)
 </SCOREFXNS>
 <FILTERS>
   <FalseFilter name=falsefilter/> # RosettaScripts has the ability to only output designs that pass a designated
filter. This functionality is not being used here.
 </FILTERS>
 <MOVERS>
    <DnaInterfacePacker name=DnaPack scorefxn=DNA task_operations=IFC,IC,AUTOprot,DnaInt/>
  </MOVERS>
  <PROTOCOLS>
   <Add mover_name=DnaPack/>
  </PROTOCOLS>
</dock_design>
```

Fig. 3 Example RosettaScripts XML file. This file can be used to set up and modify Rosetta protocols. All writing after the # mark is a comment that is not read in by the Rosetta program

3.2.1 Automatic Specificity and Binding Energy Prediction Following Fixed-Backbone Design

The simplest method of specificity prediction [2, 21] is the addition of the two lines to the XML file. This method allows for multiple repacks to be done, but it is not suitable for protocols that involve any backbone movement because the backbone is optimized for the base pair originally designed for.

1. Replace *line a* with *line b* in the XML file (Fig. 3) and run the protocol exactly as described in Subheading 3.1, but with this new XML file instead of the original:

line a: <DnaInterfacePacker name=DnaPack scorefxn=DNA task_operations=IFC,IC,AUTOprot,DnaInt/>

line b: <DnaInterfacePacker name=DnaPack scorefxn=DNA task_operations=IFC,IC,AUTOprot,DnaInt binding=1 probe_specificity=3/>

2. The number following the added options refers to the number of repacks, the lowest energy of which is used in the calculations. Three is a good choice for reducing noise in the results. A repack is a search similar to the design procedure except that only the rotameric state is varied while amino acid types are fixed.
3. The calculation results are located inside the output pdb file for each design. Open the file with a text-editing program to view the data (*see* **Note 18**).

```
AUTO # all protein positions not
explicitly noted are to be marked as
AUTO, the same as using the
AUTOprot operation
start
28 A PIKAA L # forces amino acid L
at position 28 on chain A
83 A PIKAA R
-12 C NATRO # g, fixes the native
rotamer
-11 C NATRO #  c
-10 C NATRO #  a
-9 C NATRO # g
-8 C NATRO #  a
-7 C NATAA #  a, fixes the native
residue type, but allows different
rotamers
-6 C TARGET GUA # c, target
base, same as using the dna_def
option, but DNA is required to be
explicit in the resfile
-5 C NATAA # g
-4 C NATRO # t
-3 C NATRO #  c
-2 C NATRO # g
-1 C NATRO # t
1 D NATRO #  a
2 D NATRO #  c
3 D NATRO # g
4 D NATRO #  a
5 D NATAA #  c
6 D TARGET CYT # g
7 D NATAA # t
8 D NATRO # t
9 D NATRO #  c
10 D NATRO # t
11 D NATRO # g
12 D NATRO #  c
```

Fig. 4 Example resfile. This file is used if specific protein positions or amino acid types need to be forced in the design run. It is an alternative to allowing the location of the target substitution to control the designable protein positions. All writing after the # mark is a comment that is not read in by the Rosetta program

3.2.2 Protocol for Specificity Calculation That Is Suitable Following Any Design Procedure

The main feature of a specificity calculation is that it is an exploration of rotameric and potentially backbone space in order to find and compare the energy of a set of given sequences. Therefore, the protocol used for the design procedure may not be optimal for doing these analyses. For example, the discreteness of rotamers is an approximation that is necessary because of computational limits when all amino acids are being considered. However, when the amino acid sequence is fixed, the number of rotamers included in the calculation can be greatly increased, and any negative effect of the approximation is lessened [23]. Flexible backbone calculations for specificity enable the protein backbone to be optimized for each particular base, reducing any energetic bias for the base pair in the crystal structure over the competing base types.

```
METHOD_WEIGHTS ref  -0.3 -0.7 -0.75 -0.51 0.95 -0.2 0.8
-0.7 -1.1 -0.65 -0.9 -0.8 -0.5 -0.6 -0.45 -0.9 -1.0 -0.7 2.3 1.1 #
reference weights that are for each amino acid type

fa_atr 0.95 # attractive forces between residues
fa_rep 0.44 # repulsive forces between residues
fa_intra_rep 0.004 # repulsion within a sidechain
fa_sol 0.65 # one component of desolvation
lk_ball 0.325 # newer orientation-dependent desolvation
lk_ball_iso -0.325 # newer orientation-dependent desolvation
hack_elec  0.5 # coulombic electrostatics
fa_dun 0.56 # probability for each approximated rotamer
ref 1 # weight for the reference energies
hbond_lr_bb 1.17 # hydrogen bonding
hbond_sr_bb 1.17 # hydrogen bonding
hbond_bb_sc 1.17 # hydrogen bonding
hbond_sc 1.17 # hydrogen bonding
p_aa_pp 0.64 # probability of amino acid type given
backbone
dslf_ss_dst 0.5 # disulphides
dslf_cs_ang 2 # disulphides
dslf_ss_dih 5 # disulphides
dslf_ca_dih 5 # disulphides
pro_close 1.0 # proline ring closure
```

Fig. 5 Example energy function file. This energy function was optimized to produce high sequence recovery of protein–DNA interactions over a benchmark set of proteins [23]. All writing after the # mark is a comment that is not read in by the Rosetta program

1. Modify the XML script to fix the protein sequence of the structure being analyzed (most likely the output of a previous design calculation). In the TASKOPERATIONS section of the XML file, the operation to fix the protein sequence must be added by adding the following four lines:

<OperateOnCertainResidues name=ProtNoDes>

<RestrictToRepackingRLT/>

<ResidueHasProperty property=PROTEIN/>

</OperateOnCertainResidues>

   To use this operation, the DnaInterfacePacker mover must be changed to the following:

<DnaInterfacePacker name=DnaPack scorefxn=DNA task_operations=IFC,IC,AUTOprot,ProtNoDes,DnaInt/>

2. If desired, modify the arguments file to increase the number of rotamers. The addition of the flags "-ex3" and "-ex4" is a reasonable increase. Further increases can be enabled by using the "::level #" addition to any of the -ex flags. The available levels are 1–7. An advanced XML user can add the extra rotamers through the ExtraRotamersGeneric operation and complete this specificity calculation directly after design in one run.
3. Set up four separate runs, one for each base type (or more if the target has multiple base-pair substitutions. Do runs for whichever competing states are to be compared).

4. Complete a minimum of ten runs per base type for a fixed-backbone approach and at least 50 (4× or more) for any approach involving flexible backbone.
5. Collect the total_score value from inside of each pdb. The specificity can be calculated from the lowest-energy structure or from the mean or median of the energies of all structures. A comparison of all these three specificity calculations is most informative (*see* **Notes 19** and **20**).
6. The simplest way to access these values without writing a script is to execute the command "grep total_score *pdb" in the directory that contains the pdbs you are interested in analyzing.

3.3 Advanced Design Modes

Follow instructions in Subheading 3.1 with the following described variations to the XML script (Fig. 3) and arguments files (Fig. 2). Protein backbone flexibility is accessible through the parser protocols and is in the release version of the code.

3.3.1 Protein Flexibility

1. Modify the XML file to include a second mover before the standard design mover (DnaInterfacePacker). The line to add is <DesignProteinBackboneAroundDNA name=bb scorefxn=DNA task_operations=IFC,IC,AUTOprot,DnaInt type=ccd gapspan=4 spread=3 cycles_outer=3 cycles_inner=1 temp_initial=2 temp_final=0.6/>
2. Additionally, the following line must be added after the line "<Add mover_name=DnaPack/>":
   <Add mover_name=bb/>
3. The DesignProteinBackboneAroundDNA enables the ccd backbone movement [40, 41]. An advanced user of RosettaScripts and the XML format could explore the protein backbone space with alternative protocols and then use those structures as input for standard design (*see* **Note 21**).
4. The diversity of the results will be significantly increased; thus, more runs are required to explore the design possibilities.

3.3.2 Multistate Design

Multistate design is a method to explicitly design for one state and against others [42, 43]. In the case of protein–DNA design, those states are the targeted bases and the alternative bases [19, 21]. This method is accessible through the parser protocols and is in the release version of the code. Follow instructions in Subheading 3.1 with the following variations to the XML script (Fig. 3).

1. Modify the XML file by replacing the standard DNA design mover with the following mover for doing multistate:
   <DnaInterfaceMultiStateDesign name=msd scorefxn=DNA task_operations=IFC,IC,AUTOprot,DnaInt pop_size=20 num_packs=1 numresults=0 boltz_temp=2 anchor_offset=15 mutate_rate=0.8 generations=5/>

2. Additionally, the line "<Add mover_name=DnaPack/>" must be replaced with the line:
<Add mover_name=msd/>
3. All of the parameters of the genetic algorithm can be varied, and the ones in the above line are testing parameters. Refer to cited literature [19, 21, 42] to identify good starting parameters for a particular design challenge.

3.3.3 DNA Flexibility

Crystal structures of engineered proteins indicate that DNA flexibility is a critical component of target site recognition [21, 27]. The DNA movement protocols in Rosetta [23, 24] are more experimental than the standard design methods and are undergoing significant development.

1. Acquire access to the developer's version of the code (*see* Subheading 2 and **Note 22**).
2. There is no RosettaScripts capability outside of the trunk version of Rosetta. Therefore there are separately compiled apps required for each protocol instead of one app with access to many movers through an XML file.
3. Compile the dna_fragment_rebuild_with_motifs app as a first step toward using DNA flexibility, or contact the people listed in **Note 22** to receive instructions or begin collaborations to access more advanced versions of the code.
4. Exact instructions and arguments files required for using this app are available in the supplemental material of reference [23].

3.3.4 High-Temp Packer

Multiple low-energy solutions exist for most protein engineering challenges. The standard design method tends to produce two or three different solutions at the most. Using computation to guide library design [14, 15] depends on having multiple designs to combine in the selection process. Flexible backbone methods can increase the number of solutions, but these solutions may not reflect the true potential movements of backbones because modeling of flexibility is a challenging problem. The high-temp packer approach increases the temperature that the algorithm driving the design process converges to and thus increases the chance of producing a design that is low energy, but not the predicted lowest energy. The energy function used in the design process may not be perfectly optimized for every design situation. Being able to produce designs that are not predicted to be the lowest energy, but that still contain high-quality contacts, is one way to alleviate the impact of an imperfect energy function on the design process. The supplemental methods of reference [23] describes the two changes required to use this method. The changes can be made to any version of Rosetta, and then the code must be recompiled.

3.3.5 Motifs

One downfall of the necessary rotamer approximation is that favorable interactions can be missed. Design procedures are limited in how large of a rotamer set can be used. One way to get around this limit is to use motifs, libraries of interactions seen in crystal structures, to increase rotamer sampling. The protocol consists of a search procedure using a greatly expanded rotamer set to see if one of these native-like interactions can be made with a target base pair. Once rotamers from the expanded set are identified, they are added to the standard rotamer set to bias the sampling for these likely favorable interactions. An energetic bonus can also be given to these rotamers to overcome potential inaccuracies in the energy function. Motif-based protocols are only available in the developer's version of the code, and their usage is described in detail in the supplemental methods of reference [23].

1. Acquire access to the developer's version of the code (*see* Subheading 2 and **Note 22**).
2. Compile the motif_dna_packer_design app. This application is available in both trunk Rosetta and in the more experimental branch of Rosetta that focuses on improving modeling of DNA flexibility.
3. Collect a library of motifs by compiling the dna_motif_collector app, downloading all protein–DNA complexes under some resolution cutoff (<2.8 is reasonable), and running the application by following the instructions in reference [23].
4. Add the line "special_rot 1.0" to the energy function (Fig. 5).
5. Add flags to the args file ([23], Fig. 2) to load in the motif library, set up cutoffs for acceptance of a motif rotamer, pick a rotamer level for the expanded motif rotamer library, and pick the energetic bonuses to try for these added rotamers.
6. If the trunk version of Rosetta is being used, add the line "-patch_selectors SPECIAL_ROT" to the args file.

4 Notes

1. DNA-interacting proteins can have either or both high activity and specificity [2]. An example of an enzyme with high activity and low specificity is DNA polymerase. Nonspecific proteins with high activity often use DNA backbone contacts to gain binding energy. Homing endonucleases are highly specific and their levels of activity vary. High-specificity proteins can also be low fidelity, meaning that they can have tolerance at some of the nucleotide positions in their target sites while maintaining an overall high level of specificity due to a long target site. Homing endonucleases tend to have low fidelity in order to maintain activity in the face of genetic drift of their target sites [5].

2. Another potential cause of high specificity is indirect readout [37, 38]. While direct readout is characterized by direct interactions with the target site, such as hydrophobic packing and hydrogen bonds, indirect readout is related to the DNA bending preferences of a target site sequence. There is some knowledge of the rules of indirect readout, but the energetics of indirect readout are just beginning to be incorporated into the Rosetta models [24], and the relationship of DNA bending to cleavage of target DNA by endonucleases is not understood.
3. Avoiding a reduction in specificity is an important consideration when engineering these reagents. Sometimes an endonuclease can maintain high levels of activity while losing interface interactions and specificity. Some target positions are nonspecific with the native enzymes because of the evolutionary pressure to maintain cleavage of a target site that is subject to genetic drift [5]. Computational redesigns at these positions can gain interface contacts and have enhanced specificity [19].
4. A rotamer is a low-energy conformation of an amino acid [22]. The computational methods rely on these discrete states to make the calculations feasible. The protocol to identify the lowest-energy design relies on a simulated annealing algorithm [17].
5. While there is no recognition code currently understood for homing endonucleases, it is possible that the rules governing specificity will be revealed as more native endonucleases are characterized. Collecting data on the specificity of many LAGLIDADG endonucleases, and analysis of their interface interactions most likely through homology modeling, is the only way to determine whether there is an understandable code that can be incorporated into the modeling.
6. David Baker (University of Washington), Philip Bradley (Fred Hutchinson Cancer Research Center), Jens Meiler (Vanderbilt), Jeffrey Gray (Johns Hopkins), Brian Kuhlman (UNC Chapel Hill), Tanja Kortemme (UCSF), Jim Havranek (Washington University of St. Louis), Richard Bonneau (NYU), Rhiju Das (Stanford), John Karanicolas (University of Kansas), Sarel Fleishman (Weizmann Institute of Science), Ora Furman (Hebrew University), Ingemar André (Lund University), and Sagar Khare (Rutgers).
7. It is possible to do design starting from a high-quality homology model instead of from a crystal structure. The model must include DNA and it can carry the DNA backbone from a starting template. This procedure requires that there is a homologue of the protein of interest that has been crystallized bound to DNA. Procedures to accomplish this work are not currently published, but they will be available to the public in the near future, and an advanced Rosetta user could accomplish such modeling with currently available tools.

8. A basic understanding of Linux/Unix commands is essential for running Rosetta. There are many available resources online, and one tutorial for a beginner user is located at the following web address: http://www.ee.surrey.ac.uk/Teaching/Unix/.
9. The mode = release command builds the release version instead of the debug version, and it is at least ten times faster than of a debugging executable. Only leave out the "mode = release" if you are developing code that needs to be debugged. The "extras = static" command means that static linking of shared libraries is done and that the code can be ported to other platforms. The only downside is that the sizes of the compiled executables are larger, but that is a worthwhile trade-off for portability. The command "-j #" can be used to parallelize the build into multiple threads if you are compiling on a multiprocessor machine (i.e., -j 20 for splitting work over 20 machines).
10. If the user plans on running parallel multiple trajectories of the same code, the output of these trajectories needs to go into different directories to avoid overwriting each other. A good strategy is to create internal directories labeled job0-job55 (or however many runs you want to complete). The command "mkdir job{0..55}" will generate those directories. Each parallel trajectory must write to a single one of these directories. The other option is to run jobs sequentially by using commands in the arguments file or by using capabilities within RosettaScripts [39]. The issue with running jobs sequentially is that it is much less time effective if the job is long. The job can be long if it is a complex protocol or if the pocket is multiple base pairs because that necessitates that more interface positions are being designed simultaneously.
11. The program GNU parallel is one (highly recommended) way to run multiple jobs in parallel on a multiprocessor system that does not have a job submission system in place. The website explaining the program is http://www.gnu.org/software/parallel/. A command to use GNU parallel to submit jobs 5 at a time and to have the results go into separate job# directories is the following: nice -19 ./bin/parallel -j 5 'cd {.}; ./bin/rosettaDNA.static.linuxgccrelease @../args>log;cd ../' ::: job* &
12. Many tutorials for using Vi are available online (i.e., http://www.infobound.com/vi.html).
13. The XML files are a part of RosettaScripts [39]. This system for protocol development is an integral part of the recent versions of Rosetta. It provides a flexible environment in which movers and operations can be recombined into different protocols without having to recompile Rosetta.

14. Change the 5th and 7th columns of the following five lines in the atom_properties.txt file (./rosetta_database/chemical/atom_type_sets/fa_standard/atom_properties.txt) to the values shown here:

    Phos P 2.1500 0.5850 –4.1000 3.5000 14.7000

    Narg N 1.7500 0.2384 –10.0000 6.0000 11.2000 DONOR ORBITALS

    NH2O N 1.7500 0.2384 –7.8000 3.5000 11.2000 DONOR ORBITALS

    Nlys N 1.7500 0.2384 –16.0000 6.0000 11.2000 DONOR

    ONH2 O 1.5500 0.1591 –5.8500 3.5000 10.8000 ACCEPTOR SP2_HYBRID ORBITALS

    Also change the fifth column of the three HC atoms in the LYS.params file to the value 0.48 from 0.33 to increase the positive charge of lysine. The LYS.params file is found here: "./rosetta_database/chemical/residue_type_sets/fa_standard/residue_types/l-caa/LYS.params"

15. If running many jobs on a multiprocessor system, always submit a single test run to confirm that all paths are correct and that all necessary files are included.

16. The number of runs that should be completed depends on how many base pairs are being mutated in the target site. The number of base pairs controls the number of interface positions that are designed (unless a resfile is used, *see* Fig. 4). As a starting point, a minimum of ten runs should be completed for a fixed-backbone standard design for a one base-pair substitution. At least 50 runs should be completed for a single base-pair pocket with flexibility (either protein or DNA). A triple base-pair pocket with backbone flexibility needs several hundred runs (300–500) to assess the full range of low-energy solutions.

17. These calculations could also be used to improve the models by comparison of results with experimental activity assays and to predict the binding sites for proteins with unknown target preferences.

18. Efficiency can be greatly increased if a script is written to pull these numbers out of each designed pdb file.

19. It is recommended that either the mean or median value of the total_energy for all structures be used for specificity prediction, rather than the score of the lowest-energy structure. The mean or median is a more accurate predictor because the protocol can generate outlier structures with energies much lower than the majority and these outliers are as likely to represent the actual energetic and structural state of the complex. This recommendation is especially true for protocols involving any amount of backbone flexibility.

20. The calculation of specificity is based on the Boltzmann distribution. The value of $k_B T$ can be changed, but a value of 1 is reasonable. The equation for calculating specificity for a guanine base pair is $(2.718\text{^}0)/(2.718\text{^}0 + 2.178\text{^}(-\Delta E_{G\text{-}A}) + 2.178\text{^}(-\Delta E_{G\text{-}C}) + 2.178\text{^}(-\Delta E\text{G-T}))$.
21. Only the DesignProteinBackboneAroundDNA mover will limit protein backbone movement to around the target base pair. Other methods of protein backbone movement will require another way of designating the regions that should be flexible.
22. Contact Summer Thyme at sthyme@gmail.com or Philip Bradley at pbradley@fhcrc.org for information on the most updated branch of the developer's code needed to use DNA flexibility.

Acknowledgements

The authors would like to thank Justin Ashworth, Phil Bradley, and Jim Havranek for their vast contributions to improving protein–DNA interface design, as well as the entire RosettaCommons community for contributions to the Rosetta code base. This work was supported by the US National Institutes of Health (#GM084433 and #RL1CA133832 to DB), the Foundation for the National Institutes of Health through the Gates Foundation Grand Challenges in Global Health Initiative, and the Howard Hughes Medical Institute.

References

1. Jin X, West SM, Joshi R, Honig B, Mann RS (2010) Origins of specificity in protein–DNA recognition. Annu Rev Biochem 79:233–269
2. Ashworth J, Baker D (2009) Assessment of optimization of affinity and specificity at protein–DNA interfaces. Nucleic Acids Res 37:e73
3. Morozov AV, Havranek JJ, Baker D, Siggia ED (2005) Protein–DNA binding specificity predictions with structural models. Nucleic Acids Res 33:5781–5798
4. Stoddard BL (2005) Homing endonuclease structure and function. Q Rev Biophys 38: 39–95
5. Stoddard BL (2011) Homing endonucleases: from microbial genetic invaders to reagents for targeted DNA modification. Structure 19:7–15
6. Gao H et al (2010) Heritable targeted mutagenesis in maize using a designed endonuclease. Plant J 61:176–187
7. Windbichler N et al (2011) A synthetic homing endonuclease-based gene drive system in the human malaria mosquito. Nature 473:212–215
8. Marcaida MJ, Munoz IG, Blanco FJ, Prieto J, Montoya G (2009) Homing endonucleases: from basis to therapeutic applications. Cell Mol Life Sci 67:727–748
9. Perez EE et al (2008) Establishment of HIV-1 resistance in CD4+ T cells by genome editing using zinc-finger nucleases. Nat Biotechnol 26: 808–816
10. Takeuchi R, Certo M, Caprara MG, Scharenberg AM, Stoddard BL (2008) Optimization of in vivo activity of a bifunctional homing endonuclease and maturase reverses evolutionary degradation. Nucleic Acids Res 37:877–890
11. Chames P, Epinat JC, Guillier S, Patin A, Lacroix E, Pâques F (2005) In vivo selection of engineered homing endonucleases using

double-strand break induced homologous recombination. Nucleic Acids Res 33:e178

12. Doyon JB, Pattanayak V, Meyer CB, Liu DR (2006) Directed evolution and substrate specificity profiling of homing endonuclease I-SceI. J Am Chem Soc 128:2477–2484
13. Jarjour J et al (2009) High-resolution profiling of homing endonuclease binding and catalytic specificity using yeast surface display. Nucleic Acids Res 37:6871–6880
14. Voigt CA, Mayo SL, Arnold FH, Wang Z (2001) Computational method to reduce the search space for directed protein evolution. Proc Natl Acad Sci U S A 98:3778–3783
15. Chen MM, Snow CD, Vizcarra CL, Mayo SL, Arnold FH (2012) Comparison of random mutagenesis and semi-rational designed libraries for improved cytochrome P450 BM3-catalyzed hydroxylation of small alkanes. Protein Eng Des Sel 25:171–178
16. Khersonsky O, Röthlisberger D, Wollacott AM, Murphy P, Dym O, Albeck S, Kiss G, Houk KN, Baker D, Tawfik DS (2011) Optimization of the in-silico-designed kemp eliminase KE70 by computational design and directed evolution. J Mol Biol 407:391–412
17. Leaver-Fay A et al (2011) Rosetta3: an object-oriented software suite for simulation and design of macromolecules. Methods Enzymol 487:545–574
18. Ashworth J, Havranek JJ, Duarte CM, Sussman D, Monnat RJ Jr, Stoddard BL, Baker D (2006) Computational redesign of endonuclease DNA binding and cleavage specificity. Nature 441:656–659
19. Thyme SB, Jarjour J, Takeuchi R, Havranek JJ, Ashworth J, Scharenberg AM, Stoddard BL, Baker D (2009) Exploitation of binding energy for catalysis and design. Nature 461:1300–1304
20. Ulge UY, Baker DA, Monnat RJ Jr (2011) Comprehensive computational design of mCreI homing endonuclease cleavage specificity for genome engineering. Nucleic Acids Res 39:4330–4339
21. Ashworth J, Taylor GK, Havranek JJ, Quadri SA, Stoddard BL, Baker D (2010) Computational reprogramming of homing endonuclease specificity at multiple adjacent base pairs. Nucleic Acids Res 38:5601–5608
22. Dunbrack RL Jr, Cohen FE (1997) Bayesian statistical analysis of protein side-chain rotamer preferences. Protein Sci 6:1661–1681
23. Thyme SB, Baker D, Bradley P (2012) Improved modeling of side-chain–base interactions and plasticity in protein–DNA interface design. J Mol Biol 419:255–274
24. Yanover C, Bradley P (2011) Extensive protein and DNA backbone sampling improves structure-based specificity prediction for C2H2 zinc fingers. Nucleic Acids Res 39:4564–4576
25. Havranek JJ, Baker D (2009) Motif-directed flexible backbone design of functional interactions. Protein Sci 18:1293–1305
26. Li H, Ulge UY, Hovde BT, Doyle LA, Monnat RJ Jr (2011) Comprehensive homing endonuclease target site specificity profiling reveals evolutionary constraints and enables genome engineering applications. Nucleic Acids Res 40:2587–2598
27. Redondo P et al (2008) Molecular basis of xeroderma pigmentosum group C DNA recognition by engineered meganucleases. Nature 456:107–111
28. Takeuchi R, Lambert AR, Mak AN, Jacoby K, Dickson RJ, Gloor GB, Scharenberg AM, Edgell DR, Stoddard BL (2011) Tapping natural reservoirs of homing endonucleases for targeted gene modification. Proc Natl Acad Sci U S A 108:13077–13082
29. Grizot S, Duclert A, Thomas S, Duchateau P, Pâques F (2011) Context dependence between subdomains in the DNA binding interface of the I-CreI homing endonuclease. Nucleic Acids Res 39:6124–6136
30. Pabo CO, Nekludova L (2000) Geometric analysis and comparison of protein–DNA interfaces: why is there no simple code for recognition? J Mol Biol 301:597–624
31. Miller JC et al (2011) A TALE nuclease architecture for efficient genome editing. Nat Biotechnol 29:143–148
32. Fleishman SJ et al (2011) Computational design of proteins targeting the conserved stem region of influenza hemagglutinin. Science 332:816–821
33. Röthlisberger D et al (2008) Kemp elimination catalysts by computational enzyme design. Nature 453:190–195
34. Azoitei ML et al (2011) Computation-guided backbone grafting of a discontinuous motif onto a protein scaffold. Science 334:373–376
35. Szeto MD, Boissel SJS, Baker D, Thyme SB (2011) Mining endonuclease cleavage determinants in genomic sequence data. J Biol Chem 286:32617–32627
36. Baxter S, Lambert AR, Kuhar R, Jarjour J, Kulshina N, Parmeggiani F, Danaher P, Gano J, Baker D, Stoddard BL, Scharenberg AM (2012) Engineering domain fusion chimeras from I-OnuI family LAGLIDADG homing endonucleases. Nucleic Acids Res 40:7985–8000

37. Steffen NR, Murphy SD, Tolleri L, Hatfield GW, Lathrop RH (2002) DNA sequence and structure: direct and indirect recognition in protein–DNA binding. Bioinformatics 18: S22–S30
38. Becker NB, Wolff L, Everaers R (2006) Indirect readout: detection of optimized sequences and calculation of relative binding affinities using different DNA elastic potentials. Nucleic Acids Res 34:5638–5649
39. Fleishman SJ et al (2011) RosettaScripts: a scripting language interface to the Rosetta macromolecular modeling suite. PLoS One 6:e20161
40. Canutescu AA, Dunbrack RL (2003) Cyclic coordinate descent: a robotics algorithm for protein loop closure. Protein Sci 12: 963–972
41. Wang C, Bradley P, Baker D (2007) Protein–protein docking with backbone flexibility. J Mol Biol 373:503–519
42. Havranek JJ, Harbury PB (2003) Automated design of specificity in molecular recognition. Nature Struct Biol 10:45–52
43. Mitchell M (1996) An introduction to genetic algorithms, MIT Press