HHS Public Access

Author manuscript

J Chem Inf Model. Author manuscript; available in PMC 2019 May 29.

Published in final edited form as:

J Chem Inf Model. 2018 May 29; 58(5): 895–901. doi:10.1021/acs.jcim.8b00060.

Protocols for Requirement-Driven Protein Design in the Rosetta Modeling Program

Sharon L. Guffy[†,§], **Frank D. Teets**[†,§], **Minnie I. Langlois**[†], and **Brian Kuhlman**[†,‡,*]

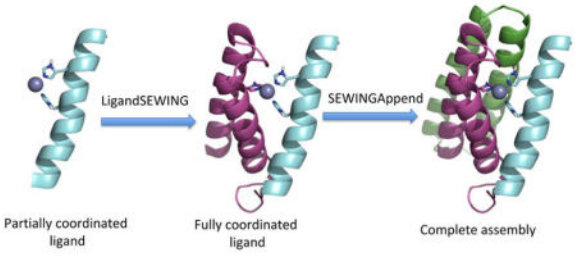
[†]Department of Biochemistry and Biophysics, University of North Carolina at Chapel Hill, 120 Mason Farm Road, Chapel Hill, North Carolina 27599, United States

[‡]Lineberger Comprensive Cancer Center, University of North Carolina at Chapel Hill, Chapel Hill, NC 27599, United States

Abstract

We have developed a set of protocols in the molecular modeling program Rosetta for performing requirement-driven protein design. First, the user specifies a set of structural features that they wish to be present in the designed protein. These requirements can be general, e.g. create a protein with five helices, or they can be very specific and require the correct placement of a set of amino acids to bind a ligand. Next, a large set of protein models are generated that satisfy the design requirements. The models are built using a method we recently introduced into Rosetta, called SEWING, that rapidly assembles novel protein backbones by combining pieces of naturally occurring proteins. In the last step of the process, rotamer-based sequence optimization and backbone refinement are performed with Rosetta, and a variety of quality metrics are used to pick sequences for experimental characterization. Here, we describe the input files and user options needed to run SEWING and perform requirement-driven design, and provide detailed instructions for two specific applications of the process: the design of new structural elements at a protein-protein interface and the design of ligand binding sites.

Graphical Abstract



*Corresponding Author: to whom correspondence should be addressed. bkuhlman@unc.edu.
§These authors contributed equally.

Author Contributions

The manuscript was written through contributions of all authors. All authors have given approval to the final version of the manuscript.

Supporting Information. Sample scripts. This material is available free of charge via the Internet at http://pubs.acs.org.

Introduction

Most studies in *de novo* protein design have begun with the researcher specifying a protein fold they would like to create and then using molecular modeling to identify amino acid sequences that will adopt that fold.[1–3] While such *de novo* designs provide valuable information about the minimal determinants of protein folding and stability, the long term goal for the field is to create *de novo* proteins that also perform useful functions. When designing for protein function, it is no longer critical that the protein adopt a particular type of fold, but rather it must contain specific binding sites that allow it to perform its function, whether it be catalysis, transport, scaffolding, etc. We refer to this type of protein design as requirement-driven design, where the goal is not to create a specific protein topology, but rather create a well-folded protein (from many possible alternative topologies) that satisfies a set of user-specified requirements. These requirements can be quite general, for instance create a protein with a groove of a certain size, or they can be more detailed and require the precise placement of a set of amino acids to create a ligand binding site. Here, we describe a set of protocols we have written in the molecular modeling program Rosetta[4] for performing requirement-driven protein design.

A key component of our approach is a method, called SEWING, we developed for rapidly generating a large set of protein models that have varying sizes and three-dimensional structures, and are inherently designable because they have been assembled from pieces of naturally occurring proteins. In a previous study we showed that SEWING can be used to create novel helical bundles with structures that closely match the design models.[5] SEWING begins by extracting supersecondary structure elements (e.g. helix-loop-helix motifs), called substructures, from native proteins. These substructures are then candidates for combination when the C-terminal secondary structural element of one substructure structurally aligns with the N-terminal secondary structural element of another substructure, and the substructures can be combined without clashes (Figure 1). Note that SEWING is currently only compatible with helical substructures. Beta strands may be present in the segments, starting structure, or partner protein(s), but SEWING cannot hybridize beta strands. Complete backbones are assembled using a Monte Carlo method of adding and deleting substructures using a user-specified score function. In a final step, rotamer-based sequence optimization protocols in Rosetta are used to identify low energy amino acid sequences for the SEWING generated backbone.

To use SEWING for requirement-driven design, filters and/or score terms are included during the assembly process to favor the generation of structures that satisfy the requirements. For instance, to build structures with a metal binding site a score term is added that favors structures with residues placed in the correct position to coordinate with the metal. Additionally, the user can specify a starting substructure for SEWING that includes some of the structural features needed to satisfy a requirement. For example, the starting structure may include one or two critical contacts with a ligand, and then SEWING can be used to complete the ligand binding site.

In this paper we outline how to use SEWING to perform requirement-driven design (Figure 2). The primary interface to SEWING is RosettaScripts, an XML-based scripting interface

for Rosetta[6]. A typical RosettaScript for SEWING contains sections to specify the database of substructures that will be used, the score functions and filters that will be used to bias the assembly process, a starting structure if desired, and parameters that control the Monte Carlo assembly. After generating a set of backbone models, a separate RosettaScript is used to perform the final sequence optimization. To make this process more concrete, we also present two specific applications of SEWING: the design of new contacts at a protein-protein interface and the design of ligand binding sites.

Walkthrough and General Guidelines

To run SEWING, users must have a compiled copy of Rosetta[4]. Directions for obtaining a Rosetta license (free for academic users), downloading Rosetta source code, and compiling the code are available on the RosettaCommons web site (https://www.rosettacommons.org/software/license-and-download). Rosetta applications, including SEWING, are run through a command-line interface using command-line options and associated input files. A single compute node can be sufficient to generate new backbones with SEWING, but the full design pipeline with rotamer-based sequence optimization generally requires access to a cluster with multiple processors.

SEWING is accessed through the RosettaScripts application which allows users to specify their protocol as an XML document and provides support for distributed computing.[6] Example RosettaScripts for several SEWING applications can be found in the Supplementary Materials. Briefly, each script will include a SEWING mover object (described below) in the MOVERS section. Options are defined within the mover tag, and the AssemblyScorers and AssemblyRequirements subelements allow score functions and requirements to be defined as separate subtags. Similarly, Ligand elements can be defined within the Ligands subelement and contain subtags listing their contacts and, when using LigandBindingAssemblyMover, their ideal coordination environment. Certain options also allow users to specify behaviors for particular residues using Rosetta ResidueSelector objects. For more general information on RosettaScripts, users should consult the Rosetta documentation.

Available Protocols

SEWING is available through three Rosetta Mover objects,[4] each with its own applications and available parameters. Here, we describe the basic AssemblyMover; AppendAssemblyMover and LigandBindingAssemblyMover will be described in the Applications section below.

The base protocol for SEWING is accessible through the AssemblyMover. This mover is used to generate backbones that fulfill a set of user-defined requirements without incorporating any particular starting structure. Given an input segment file (specified using the model_file_name option), this mover selects a random substructure as a starting point for each new backbone. It then performs a Monte Carlo simulation in which substructures are added, deleted, or “switched” (replaced) at the termini of the structure for a user-defined number of steps (defined by the min_cycles option). If the requirements have not been satisfied at this point, the simulation will continue until the structure satisfies the

requirements or until the user-defined maximum number of cycles (max_cycles) has been reached. To see all of the structures along the path of a SEWING trajectory for illustrative purposes use the output_pose_per_move option.

AssemblyMover performs temperature cooling from the beginning of the simulation until min_cycles is reached (specified by the start_temperature and end_temperature options). At the end of the simulation, the lowest scoring assembly is recovered. The window_width option is used to indicate the minimum number of overlapping residues which should be present when combining two substructures.

Recommended Options

A list of options available for SEWING movers and their defaults can be found in Table S1. The default values for SEWING options are appropriate for typical applications, but modifications may benefit users in certain cases. By default, SEWING trajectories run for a minimum of 10,000 cycles and a maximum of 20,000 cycles. This value is appropriate for users generating ~10,000 to 100,000 structures; however, in some cases better performance can be achieved by generating more structures (>1,000,000) using fewer cycles. When using the recommended number of cycles and temperature schedule, SEWING performs best with relatively low add and delete probabilities (each 0.1 or less). The default values (add_probability = 0.05, delete_probability = 0.005) work well in most cases; however, greater add probabilities may be necessary for large assemblies or small numbers of cycles, and larger delete probabilities may help increase sampling of interior segments. By default, SEWING uses a window width of four residues (approximately one helical turn), which is sufficient for canonical helices. Note that SEWING is not currently recommended for use with beta strands as chimerizable segments.

Command Line Options

Command-line flags for the RosettaScripts application can be provided as a flags file (Figure S1). The -in:file:s or -in:file:l flag indicates the starting structure to be used in AppendAssemblyMover or LigandBindingAssemblyMover; AssemblyMover does not use the input structure, but one must still be specified. Users must also specify the path to their RosettaScripts XML file using the -parser:protocol flag. Motif scoring (-mh) flags should be specified as described in Figure S1. A list of options for other SEWING-related applications, such as the segment_file_generator application, can be found in Table S2.

Refinement, Design, and Filtering

Since SEWING produces backbones but does not perform side chain design, users will also need to perform additional analysis and design; however, we recommend that this analysis be performed in a separate RosettaScript so that SEWING score terms will be output appropriately in the Rosetta score file. Since backbone generation is rapid compared to structure refinement and side chain design, we recommend that users only refine a fraction of generated backbones. For a typical production run we recommend that users generate at least 10,000 assemblies and select the top 10% by total SEWING score for further refinement; for a protein between 100–150 residues it will take on the order of one hundred

CPU-hours to generate 10,000 Assemblies, and approximately 500 to 1000 CPU-hours to perform refinement on 1000 Assemblies (Table S3).

An example of a typical refinement and filtering script for SEWING designs is shown in Figure S2. Although SEWING ignores side chains, it does preserve their coordinates; therefore, since they are packed differently in SEWING assemblies than in their original context, these structures will initially have numerous side chain clashes. First, an initial fixed-backbone design step is performed to remove clashes within the structure. Cartesian minimization is then used to repair any bond length and angle changes introduced during SEWING. Finally, Rosetta FastRelax performs additional rounds of iterative side chain design and backbone minimization.[7,8] Loop residues are constrained to their native identities throughout the design process to encourage proper folding of the structure, and any functional residues (i.e. “required residues” from the initial SEWING run) are held fixed. We recommend including a filter for fragment quality (FragmentLookupFilter), which verifies that each four-residue window adopts a structure compatible with its sequence.[9]

Input Files for SEWING

Segment Files—Before running SEWING, users must generate a “segment file”, which contains coordinates, sequences, and connectivity information for a set of substructures from native proteins. A pre-generated helical segment file can be found in the Rosetta database in main/database/additional_protocol_data/sewing. This segment database was generated using a set of high-resolution structures selected using the PISCES protein sequence culling server with at least 2.2 Ångstrom resolution and no more than 30% sequence identity.[10]

To generate custom segment files, use the segment_file_generator Rosetta application along with a set of input structures. Since beta sheet specific requirements have not yet been developed, we recommend that users do not include beta strands in segment files at this time.

Motif score databases—SEWING also requires files that provide information necessary for scoring using the Motif Score, a sequence-independent metric of designability of neighboring secondary structural elements.[11] These files are specified using command-line options (Figure S1) and can be found in the Rosetta database in main/database/additional_protocol_data/sewing.

Edge Files—By default, SEWING computes alignments between helices on-the-fly, which improves performance for most helical assemblies. SEWING may also be run with alignments precomputed to reduce redundancy, providing more thorough checks of secondary structure alignment at the expense of speed and memory usage. If this option is enabled, then SEWING will also require an “edge file” (described in the Supplementary Materials) defining which helices can be chimerized. While this feature is generally unnecessary for generating helical proteins, it will likely be necessary for future applications involving more irregular secondary structure elements.

Scoring

SEWING does not model amino acid side chains during assembly; all scorers and binary requirements are evaluated based on backbone coordinates, and side chain coordinates are restored at the end of each simulation. Therefore, it must rely on a low-resolution score function that evaluates potential packing based solely on these backbone coordinates. For this we primarily use MotifScore, which based on the relative position of two residues, approximates the best possible attractive energy that could be achieved between the residues by mutating the residues to alanine, valine, isoleucine, or leucine.[11] Specialized score terms are also included for interface design and/or ligand binding protein design. A list of all available scorers can be found in Table S4. For each protocol, users may specify any combination of scorers, and design trajectories will be based on the weighted sum of those scores. If no scorers are specified, SEWING will use the MotifScorer and InterModelMotifScorer with the recommended weights. This default score function is appropriate for simple backbone design when no binding partners or ligands are present. Note that specifying any scorers will override the default score function, so the MotifScorer and InterModelMotifScorer will not be included unless they are also specified.

Requirements

One of the key features of SEWING is the user's ability to place boolean restrictions on designed structures without specifying a particular fold, to be checked after every change to the Assembly and trigger a reversion to the previous state if failed. We refer to these restrictions as requirements. Requirements can range from simple (e.g. the ClashRequirement, which prevents clashes in the designed protein) to more complex requirements targeting specific regions of the structure. A full list of available requirements is found in Table S5. If no requirements are specified, then the ClashRequirement will be applied by default; we recommend this requirement for all design cases. Note that specifying any additional requirements will override the default, and the ClashRequirement will not be included unless it is also specified by the user. Note also that these requirements are not to be confused with the "requirements" in the more general sense referenced previously, which refer to all of the information provided to the SEWING protocol and used to guide backbone generation. Requirements are intended to work in concert with Scorers and the various options available to the SEWING movers to define the desired features of the finished Assembly.

Applications

Expanding Protein Interfaces

AppendAssemblyMover, also known as SEWING Append, uses the same method as AssemblyMover to generate a structure from native elements; however, instead of beginning with a random substructure, it incorporates a user-specified starting structure into all designed backbones as shown in Figure 3. Therefore, users can incorporate functional motifs, such as protein-binding peptides or ligand binding sites, into their designed backbones provided that the motifs are local in sequence. Since the only requirement for this starting structure is that SEWING can append to at least one of its termini using the provided segment file, this protocol can produce final structures containing elements that

SEWING would normally not be able to generate and optimize those elements' packing using the SEWING score function. For instance, although users may not yet use SEWING to chimerize across beta sheets, they can use AppendAssemblyMover to add structure to an existing beta sheet, provided that the starting structure has at least one terminal helix available for chimerization.

The required_resnums or required_selector options in AppendAssemblyMover can be used to guarantee that a set of residues will be preserved during assembly. To enforce this guarantee, SEWING must not include these residues in the overlapping regions of segments being combined; therefore, if a required residue is within window_width residues of the terminus of the user's starting structure, then SEWING will be unable to append to that terminus. Users in this scenario may need to append additional helical structure to the appropriate terminus before running SEWING Append.

The starting structure will by default be converted into a series of segments according to its secondary structure determined by the Dynamic Secondary Structure Prediction (DSSP) algorithm[12]. As in segment file generation, SEWING will ignore single-residue loops by default. This can be disabled by setting the strict_dssp_changes option (in this case, an XML option) to false. Users may also override the automatically detected segments by providing a series of segment start points, end points, and DSSP codes (Figure S3). The segment start and end points are provided as comma-separated lists to the pose_segment_starts and pose_segment_ends options, respectively, and the segments' secondary structure types can optionally be provided to the pose_segment_dssp option as a string with each character corresponding to one segment.

The modifiable_terminus option can be used to specify which termini can be extended during SEWING. Note that the secondary structure of this terminus should match that of the segment file being used. Extend mode is available for cases in which users simply want to extend a secondary structure element.

Adding other proteins to the simulation—One potential application for SEWING Append is to extend protein interfaces to improve binding affinity or specificity (Figure S3). In this scenario the aligned binding partner(s) is included as a separate input file using the partner_pdb option. Binding partners are considered by certain scorers and requirements but are not modified or moved relative to the starting structure. This can be used to ensure that no part of the Assembly will clash with the partner, but it can also be used in concert with the PartnerMotifScorer to bias the assembly towards placing segments in proximity to the partner, thereby allowing residue-level design to create new interface contacts. In cases where contacts to a particular region of the partner protein are desirable, the SubsetPartnerMotifScorer biases the assembly toward placing structure in proximity to a specified region of the protein. Furthermore, the termini of the designed protein may be positioned through the use of the TerminusMotifScorer, which biases the placement of terminal residues within a user-defined volume as measured by proximity to specific residues on the partner protein. This can be used to produce designs that are more likely to accommodate FRET pairs or other useful motifs without clashing with either the design or the partner.

Guidelines for use—AppendAssemblyMover will by its nature restrict the sample space available to the algorithm, since every run starts with the same substructure and potentially the same excluded volume. The addition of a partner PDB, and therefore of further constraints on the volume available for the design to fill, further restricts the SEWING algorithm. Users should therefore run many short SEWING runs in preference to fewer runs with more cycles, as longer runs may produce duplicate structures. Users should adjust the TerminusMotifScorer weights such that the range of expected score values results in a TerminusMotifScorer value similar to that of the MotifScorer terms, which can be expected to return values from −2 to 0 multiplied by the weight; while other scorers are constrained by the nature of the MotifScore to comparable ranges of possible values, TerminusMotifScorers are geometric in nature and can therefore return values large enough to dominate the score and return otherwise undesirable designs that satisfy the TerminusMotifScore requirements at the expense of all other terms.

Sample Results—To demonstrate the SEWING-Append protocol, we generated helical backbones starting from the VBS1 helix of talin in complex with vinculin (PDB code 1T01). The script shown in Figure S3 was used to extend the existing helix to create up to 5-helical bundles, producing a total of 4627 designs; the top 463 designs by total weighted SEWING score were then refined as described in the Refinement, Design, and Filtering section (Figure S2) to produce designs with an average per-residue score of −3.03 ± 0.4 kcal/mol with the Ref2015 score function. The entire design process used 252 CPU-hours.

Ligand Binding

Compatibility with existing ligands—Partial and complete ligand binding sites offer another set of possible functions to be incorporated using SEWING Append. Incorporating non-protein residues raises additional challenges, however, which have required special adaptations to the SEWING framework.

Since ligands often bind with multiple protein residues that can be distant in sequence space, ligands are stored as specialized residues which are not part of any one segment. These ligand residues store their contacts, which can either be specified manually by the user or, in the case of coordinated metal ions or covalent bonds to ligand atoms, can be automatically detected. Contacts are stored as required residues which must be preserved throughout the protocol, and the ligand is anchored to its most N-terminal contact residue. Any movements of that residue will cause the ligand to move as well. Ligand-protein contacts are scored using a new LigandMotifScorer (Table S4) which uses the distance and angle of Cα-Cβ bond vectors with hydrophobic atoms in the ligand to favor new hydrophobic ligand contacts; hydrophilic ligand atoms are not scored. A new LigandClashRequirement (Table S5) identifies protein-ligand clashes by detecting ligand atoms and protein Cα atoms within a user-specified clash radius.

Incorporating new ligand contacts—SEWING can also add contacts to partially-coordinated ligands using the new LigandBindingAssemblyMover, or LigandSEWING (Figure S4B). This mover is a specialization of SEWING Append which, in addition to adding new structural elements to a ligand binding site, also adds new ligand-protein

contacts that satisfy user-specified geometric requirements. LigandBindingAssemblyMover is solely intended for adding new ligand contacts and is typically used as a first step in a larger protocol (Figure 2); users who begin with a fully coordinated ligand, as well as those who have already completed their ligand's coordination using this mover, should use AppendAssemblyMover as described above.

To add new ligand contacts using SEWING, the user must specify the ideal geometry of the contacts. This geometry is treated as two components. First, the ligand stores ideal bond lengths and angles for each atom that will form new contacts; these can be specified as IdealCoordination subtags of the Ligand tag in RosettaScripts (Figure S5). The user must also specify the possible positions of new coordinating residues relative to the these atoms. This information is pre-generated by the user in a separate Rosetta application (the zinc_statistic_generator application is available for polar tetrahedral atoms) using the following procedure: An isolated residue is mutated in turn to all possible coordinating residues for the ligand atom. This residue then cycles through its most common rotamers. For each rotamer, the application then builds each possible position of the coordinated ligand atom and converts that atom's coordinates into the coordinate frame defined by the residue's three main-chain atoms (Figure S4A). These coordinates, along with the residue type, chi angles, atom names, and secondary structure type, are stored in a ligand coordination file (Figure S6). Each IdealCoordination tag is provided with the name(s) of these files for the ligand atom being considered.

During assembly, LigandSEWING adheres to a user-specified "segment distance cutoff" which determines how much additional structure can be built on either side of the original site. This cutoff helps prevent the mover from continuing to add structural elements that are too far from the ligand to potentially form new contacts. Each time new substructures are added to the assembly, they are first checked to see whether any added segment comes within a user-specified distance of the ligand atom being coordinated. If a segment meets this cutoff, then each added residue's position relative to the ligand is compared to the possible positions defined in the provided ligand coordination file. If a match is found, the residue is then mutated to the residue type and rotamer specified in the hash, and the overall geometry of the ligand binding site is scored. Once this process is complete for all new residues, the best-scoring mutation is kept if its geometry score meets a user-specified score threshold. When the ligand atom's coordination is complete, LigandSEWING either outputs the completed site immediately or, optionally, transitions into a SEWING Append protocol using the completed ligand binding site as the starting structure.

Guidelines for use—Since the sampling space of LigandBindingAssemblyMover is highly restricted, users should use relatively high temperature values (between 2.0 and 3.0) to ensure diverse sampling. Due to this restricted (usually exhaustive) sampling and because it accepts the first ligand contact discovered that passes its scoring thresholds, LigandSEWING does not perform temperature ramping or cooling; instead, the max_temperature option sets the value for the entire protocol. Instead, temperature cooling is performed using AppendAssemblyMover in the latter steps of the protocol, either by setting the build_site_only option to true or by performing a separate Append step after site completion.

The ligand_binding_cycles option sets the maximum number of moves that the mover will perform (not counting the optional Append step, which uses the min_cycles and max_cycles options); we recommend a value of 20000 cycles for protocols adding a single ligand contact.

To prevent wasted sampling, a segment distance cutoff should not exceed two. Note that this cutoff counts the number of substructures added to each terminus of the starting structure rather than the number of segments added. A max_distance of 8.0 Å is also recommended to prevent unnecessary calculations in segments too distant to form ligand contacts.

Recommended geometry score thresholds will vary depending on the coordination environment of the specified atom. Lower scores generally indicate more ideal geometry (and therefore a stricter cutoff). Since this score includes angles and dihedrals between all pairs of contacts to the atom in question, it scales nonlinearly with the number of contacts; for example, a value of 5.0 is appropriate for a tetrahedrally coordinated atom forming a third contact whereas a value of 20.0 will give similarly ideal geometries when forming a fourth contact. Non-ideal coordination geometry in the starting contacts also lead to increased scores. Therefore, we recommend that users try several thresholds for their specific use case before running full-scale simulations and select one which gives the highest possible success rate while yielding satisfactory ligand geometry.

Sample Results—To demonstrate the Ligand SEWING protocol (Figure S4B), we generated a set of helical proteins containing a tetravalent zinc ion beginning with a partially coordinated zinc. The script shown in Figure S5 was first used to add a fourth coordinating residue to zinc ions with three coordinating histidines using LigandBindingAssemblyMover. Five starting sites were used as input and were run for 2000 trajectories each; however, only 99 unique new contacts were identified due to the highly limited sampling space of LigandBindingAssemblyMover. These 99 sites were then used as input to AppendAssemblyMover to generate 1000 assemblies per starting site, and the top 10% of generated assemblies proceeded to refinement. After filtering designs as described in the Refinement, Design, and Filtering section (Figure S2) along with additional project-specific filters, 139 refined assemblies were available for further filtration and selection. Scores of these designs ranged from −4.2 to −3.4 kcal/mol per residue (median = −3.9 kcal/mol/res, mean = −3.9 kcal/mol/res) with the REF2015 score function.

Conclusions

SEWING provides an alternative to traditional methods for *de novo* protein design which rely on strict predetermined structural restrictions. Due to its ability to quickly generate diverse backbones, it is a promising approach for applications that benefit from having a large library of possible starting structures (e.g. enzyme design). Importantly, SEWING is also able to combine multiple functional elements (e.g. a protein binding partner and a ligand) into a single protein. By rapidly sampling a large structural space, SEWING is able to find ways to accommodate arbitrary sets of user requirements, thus encouraging functional protein design.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments

Funding Sources

This work was supported by NIH grant, R01GM117968, to B.K.

We thank Andrew Leaver-Fay and Tim Jacobs for advice during protocol development.

ABBREVIATIONS

SEWING	structural extension with native fragment graphs
DSSP	dynamic secondary structure prediction

References

1. Kuhlman B, Dantas G, Ireton GC, Varani G, Stoddard BL, Baker D. Design of a Novel Globular Protein Fold with Atomic-Level Accuracy. Science (80-). 2003; 302:1364–1368.
2. Der BS, Machius M, Miley MJ, Mills JL, Szyperski T, Kuhlman B. Metal-mediated affinity and orientation specificity in a computationally designed protein homodimer. J Am Chem Soc. 2012; 134:375–85. [PubMed: 22092237]
3. Joh NH, Wang T, Bhate MP, Acharya R, Wu Y, Grabe M, Hong M, Grigoryan G, DeGrado WF. De novo design of a transmembrane Zn^{2+}-transporting four-helix bundle. Science. 2014; 346:1520–4. [PubMed: 25525248]
4. Leaver-Fay A, Tyka M, Lewis SM, Lange OF, Thompson J, Jacak R, Kaufman K, Renfrew PD, Smith Ca, Sheffler W, Davis IW, Cooper S, Treuille A, Mandell DJ, Richter F, Ban YEA, Fleishman SJ, Corn JE, Kim DE, Lyskov S, Berrondo M, Mentzer S, Popović Z, Havranek JJ, Karanicolas J, Das R, Meiler J, Kortemme T, Gray JJ, Kuhlman B, Baker D, Bradley P. ROSETTA3: an object-oriented software suite for the simulation and design of macromolecules. Methods Enzymol. 2011; 487:545–74. [PubMed: 21187238]
5. Jacobs TM, Williams B, Williams T, Xu X, Eletsky A, Federizon JF, Szyperski T, Kuhlman B. Design of structurally distinct proteins using strategies inspired by evolution. Science. 2016; 352:687–90. [PubMed: 27151863]
6. Fleishman, SJ., Leaver-Fay, A., Corn, JE., Strauch, EM., Khare, SD., Koga, N., Ashworth, J., Murphy, P., Richter, F., Lemmon, G., Meiler, J., Baker, D. RosettaScripts: A Scripting Language Interface to the Rosetta Macromolecular Modeling Suite. In: Uversky, VN., editor. PLoS One. Vol. 6. 2011. p. e20161
7. Tyka MD, Keedy DA, André I, DiMaio F, Song Y, Richardson DC, Richardson JS, Baker D. Alternate States of Proteins Revealed by Detailed Energy Landscape Mapping. J Mol Biol. 2011; 405:607–618. [PubMed: 21073878]
8. Nivón, LG., Moretti, R., Baker, D. A Pareto-Optimal Refinement Method for Protein Design Scaffolds. In: Zhang, Y., editor. PLoS One. Vol. 8. 2013. p. e59004
9. Dang B, Wu H, Mulligan VK, Mravic M, Wu Y, Lemmin T, Ford A, Silva DA, Baker D, DeGrado WF. De novo design of covalently constrained mesosize protein scaffolds with unique tertiary structures. Proc Natl Acad Sci U S A. 2017; 114:10852–10857. [PubMed: 28973862]
10. Wang G, Dunbrack RL. PISCES: a protein sequence culling server. Bioinformatics. 2003; 19:1589–1591. [PubMed: 12912846]
11. Fallas JA, Ueda G, Sheffler W, Nguyen V, McNamara DE, Sankaran B, Pereira JH, Parmeggiani F, Brunette TJ, Cascio D, Yeates TR, Zwart P, Baker D. Computational design of self-assembling cyclic protein homo-oligomers. Nat Chem. 2016; 9:353–360. [PubMed: 28338692]

12. Kabsch W, Sander C. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. Biopolymers. 1983; 22:2577–2637. [PubMed: 6667333]

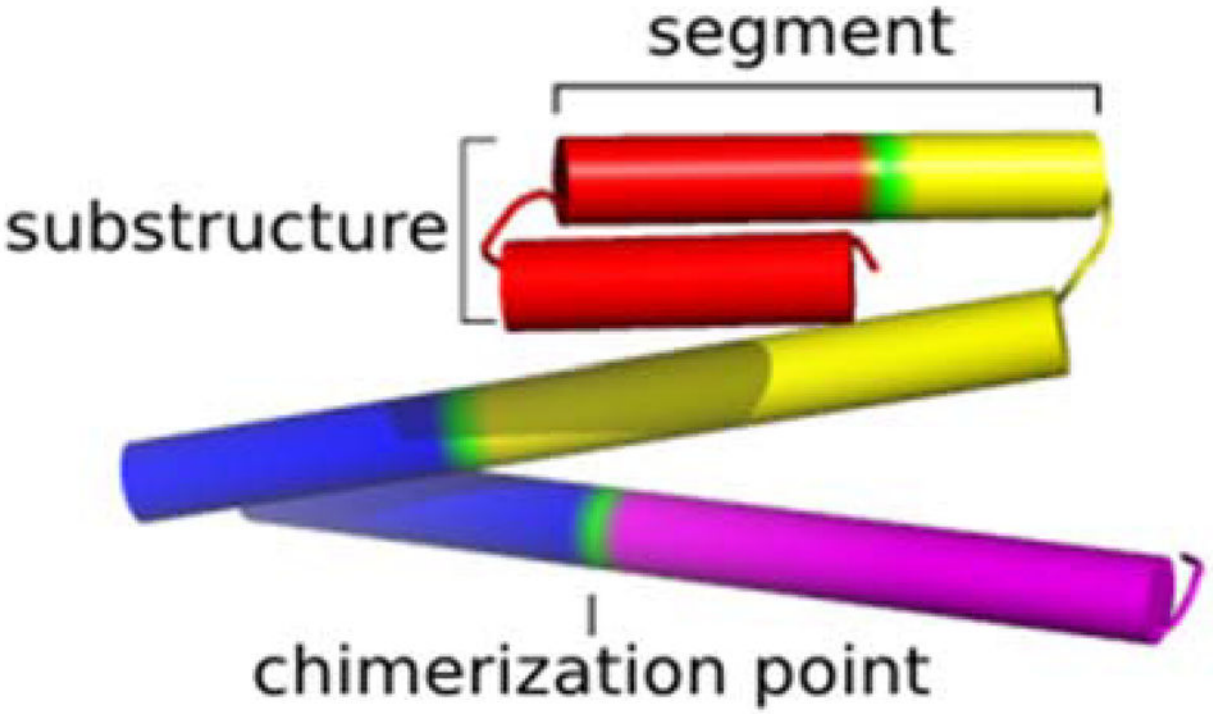


Figure 1.

A sample SEWING output, or Assembly. Individual segments, contiguous sets of residues with a common secondary structure, are sourced from crystal structures as contiguous substructures and recombined into an Assembly by transforming a residue of one segment onto a residue of another (collectively a basis pair) by pair-fitting the backbone atoms and saving the N-terminal region of one segment and the C-terminal region of another.

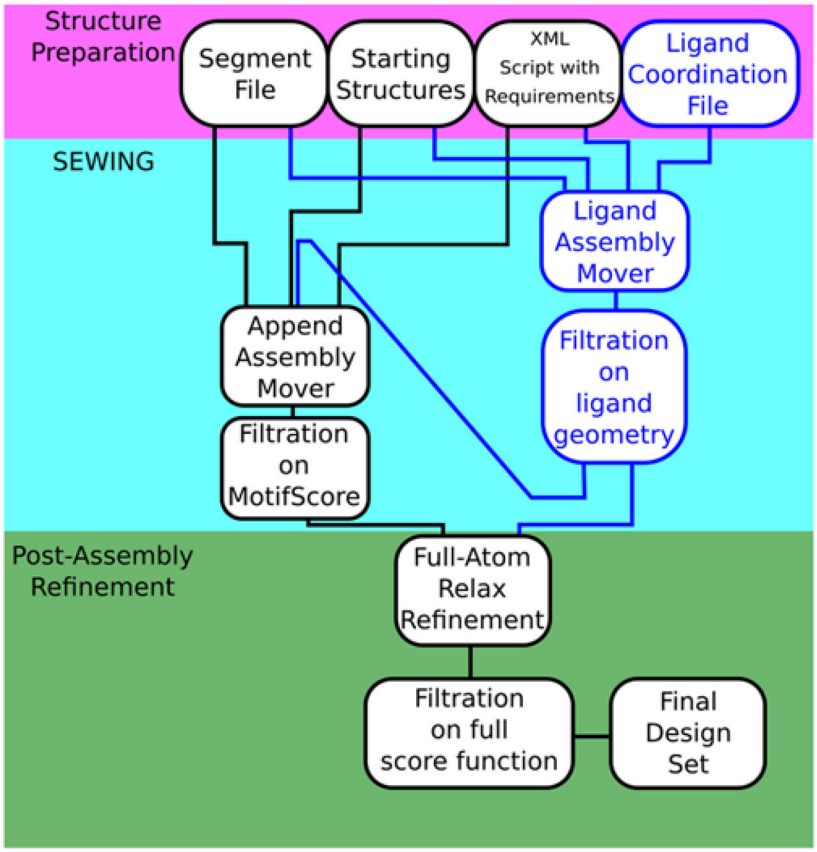


Figure 2.

A flowchart showing the general SEWING workflow. Portions unique to ligand binding site design are shown in blue; after generating assemblies containing the site, design may proceed through AppendAssemblyMover or directly to full-atom design.

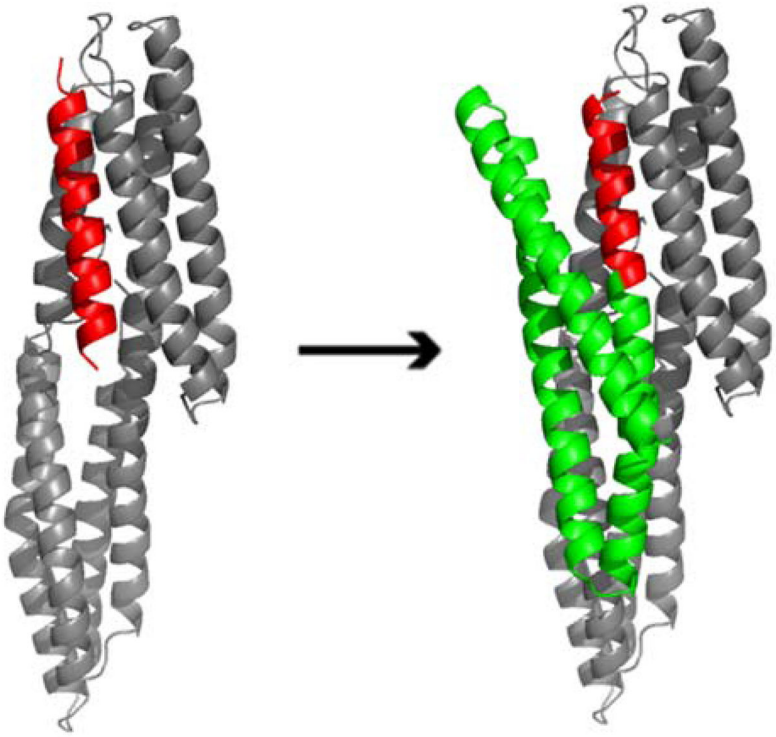


Figure 3.
Interface expansion with SEWING. Starting from a fragment of talin(red) bound to vinculin(grey), the starting fragment is expanded (green) to make additional contacts with vinculin.