BAGEL: PROTEIN ENGINEERING VIA EXPLORATION OF AN ENERGY LANDSCAPE

TECHNICAL REPORT

Jakub Lála
Department of Materials
Imperial College London
London
jakublala@gmail.com

Ayham Al-Saffar
Department of Materials
Imperial College London
London
aa1021@imperial.ac.uk

Stefano Angiolleti-Uberti
Department of Materials
Imperial College London
London
sangiole@imperial.ac.uk

July 5, 2025

ABSTRACT

Despite recent breakthroughs in deep learning methods for protein design, existing computational pipelines remain rigid, highly specific, and ill-suited for tasks requiring non-differentiable or multi-objective design goals. In this report, we introduce BAGEL, a modular, open-source framework for programmable protein engineering, enabling flexible exploration of sequence space through model-agnostic and gradient-free exploration of an energy landscape. BAGEL formalizes protein design as the sampling of an energy function, either to optimize (find a global optimum) or to explore a basin of interest (generate diverse candidates). This energy function is composed of user-defined terms capturing geometric constraints, sequence embedding similarities, or structural confidence metrics. BAGEL also natively supports multi-state optimization and advanced Monte Carlo techniques, providing researchers with a flexible alternative to fixed-backbone and inverse-folding paradigms common in current design workflows. Moreover, the package seamlessly integrates a wide range of publicly available deep learning protein models, allowing users to rapidly take full advantage of any future improvements in model accuracy and speed. We illustrate the versatility of BAGEL on four archetypal applications: designing *de novo* peptide binders, targeting intrinsically disordered epitopes, selectively binding to species-specific variants, and generating enzyme variants with conserved catalytic sites. By offering a modular, easy-to-use platform to define custom protein design objectives and optimization strategies, BAGEL aims to speed up the design of new proteins. Our goal with its release is to democratize protein design, abstracting the process as much as possible from technical implementation details and thereby making it more accessible to the broader scientific community, unlocking untapped potential for innovation in biotechnology and therapeutics.

1 Introduction

Over the past decade, advances in protein structure prediction - most notably deep learning-based models such as AlphaFold2 [1], RosettaFold [2] and ESMFold [3] - have transformed the field of protein design [4]. The ability to predict the fold of (almost any) arbitrary sequence with relatively high accuracy enables researchers to effectively explore the vast protein sequence space and find those satisfying a set of desired structural and functional properties, increasing the potential for breakthroughs in therapeutic design, biosensing, and industrial biocatalysis [4, 5, 6]. Building on top of these developments, a new generation of computational protein design frameworks has emerged [7]. While successful, many remain tightly coupled to specific multi-step workflows inherited from the "inverse folding paradigm" [8], developed prior to the advent of accurate deep learning-based protein-folding algorithms.

The inverse folding paradigm follows the philosophy *backbone design* $\rightarrow$ *sequence painting* $\rightarrow$ *folding*. In such a workflow, different algorithms (e.g., hallucination [9] or RFDiffusion [10]) are used to determine specific backbone shapes satisfying a given constraint, for example, formation of an interface with a specific epitope on a target protein. Once the fold of these backbones is determined, an inverse-folding algorithm is applied (e.g., the ProteinMPNN-based

models [11, 12, 13]) to find the sequences most likely to fold into the backbone shapes. Finally, a protein-folding algorithm is used to check if such sequences indeed fold to the correct structure. This last validation step is necessary because, while a candidate sequence obtained as a result of the second step is the most likely to fold into the target backbone, the latter is not necessarily the most stable structure for that sequence. As a result, many designed sequences are rejected and the process is repeated until one is found that successfully folds into the target backbone. Alternatively to the inverse folding paradigm, one can train a single deep generative model on the joint sequence-structure distribution so that, when conditioned on a desired functional tag, it proposes a complete sequence-structure design in a one-shot generation setting [14, 15, 16]; the resulting design also being filtered with a folding predictor. This second approach often leads to poorly interpretable pipelines, with most of the design process becoming abstracted away inside a large neural network.

In both of the previous approaches, conditioning the protein generation on specific functional properties of interest is not particularly straightforward, or even possible. For example, preventing hydrophobic patches to be located on the surface, requesting the presence of certain folds, or enforcing a given symmetry cannot be easily and explicitly implemented. As a result, many different candidates must be generated, and subsequently screened downstream for both validated folding, and wanted properties. Recently, the authors of the hallucination-based pipeline BindCraft [7] recognized that re-folding validation is the critical step for obtaining successful designs. BindCraft's philosophy, therefore, is to maximize the success rate in this validation stage by using AlphaFold2-Multimer as an oracle to co-design sequence-structure pairs directly during backbone generation - rather than following the traditional multi-step pipeline. Such hallucination-based approach, however, requires all the design properties of the system to be fully differentiable, hindering its generalizability to arbitrary design constraints.

To solve these problems, we introduce BAGEL[1], a general-purpose Python package for programmable protein engineering. Our framework enables users to formalize protein design tasks as the optimization of an energy function (a loss) over sequence space. In practice, BAGEL provides a set of tools to seamlessly build arbitrary sampling and optimization problems by combining energy terms derived from any metric extractable from a sequence. For instance, these can be derived from structure and folding metrics computed by a deep learning structure predictor, vector embeddings from a protein language model (PLM), or metrics based on sequence alone. More specifically, these energy functions include geometric constraints, similarity to user-defined templates, physical properties, and custom objectives based on structural or functional hypotheses, as well as bulk and local confidence scores, e.g., local pLDDT, pTM or interface PAE (iPAE). While various energy terms have already been implemented in this initial release, BAGEL's modular structure also easily allows a user to define any term that could be calculated from knowledge of a (group of) sequence(s). The flexibility of this abstraction allows our framework to support a wide variety of classical protein engineering problems: from templating specific scaffold motifs for *de novo* enzymes, through building proteins with desired geometry, to designing protein binders. Crucially, to go beyond the capability of current approaches, we also allow to define loss functions that consider multiple states, which is particularly valuable, for example, in engineering proteins that binds to a specific target, while explicitly maintaining a weak interaction with other specific, non-target proteins.

Our package is inspired by - and builds upon - the seminal work of the former FAIR team at Facebook (now Meta), who developed the suite of ESM projects [17, 3], and in particular the paper titled *A high-level programming language for generative protein design* by Hie et al. [18]. In that work, the authors use Monte Carlo (MC) optimization to scan the sequence space in search for optimal sequences solving a prescribed protein engineering problem. In contrast to their work, we change the representation of a protein sequence and its associated loss function by shifting away from a graph-based representation in favour of one based on groups of residues, which we believe allows for a more flexible and user-friendly design interface. Besides this change in representation, among other differences that will be addressed later, our fully modular design framework provides:

- **A Multi-State Formalism.** Native support to specify multi-state problems, highly relevant for complex design tasks of, among others, selective binding, or binder design over multiple target variants (cross-reactivity).

- **Enhanced Energy Formalism.** A unified formalism for expressing arbitrary design constraints as energy penalties, following the convention from molecular simulations where total energy is expressed as a sum of simple, fully customizable N-body terms.

- **A Unified Approach to Optimization and Generative Problems.** By looking at protein design as the exploration of an energy landscape, optimization and candidate generations can be seen as two related variations of the same problem: finding the energy minimum (optimization) as opposed to exploring the sequence space around a certain energy value (generation).

[1] Biomolecular Algorithm for Guidance in Energy Landscapes

- **Multiple Optimization and Sampling Algorithms.** An extensive and easy-to-extend sampling and optimization backend supporting different types of advanced MC algorithms to accelerate exploration of the sequence space including simulated tempering, grand-canonical MC, and custom annealing schedules.
- **Flexible Oracle Choice.** A generalized and modular implementation of an oracle, which is defined as any algorithm that provides useful information for protein engineering tasks. This includes, for example, protein-folding models that return structures, protein language models that generate embeddings, or other algorithms (or combinations thereof) that compute relevant physicochemical properties for one or more proteins.
- **Accessibility.** A user-friendly interface that can be seamlessly integrated with other Python workflows, including a serverless implementation of deep learning-based protein models (i.e., oracles) through a standalone package, `boileroom` [19], abstracting away tedious dependency management.

Compared to other, problem-specific end-to-end solutions, BAGEL has been built from scratch with generality and modularity in mind, enabling continuous extension of the package with new minimization strategies beyond MC and new energy terms describing protein engineering objectives. Moreover, the package is fully model-agnostic, enabling us to leverage the continous advances in deep learning on proteins in terms of both quality and speed.

In this report, we describe the functionality and design philosophy of BAGEL through its algorithm formalism and core class abstractions. We illustrate its utility with four examples: i) simple binder design, ii) design of binders targeting intrinsically disordered epitopes, iii) design of selective binders via multi-state optimization, and iv) enzyme variant generation. Figure 1 summarizes the design workflow and the components of the package, including the applications.

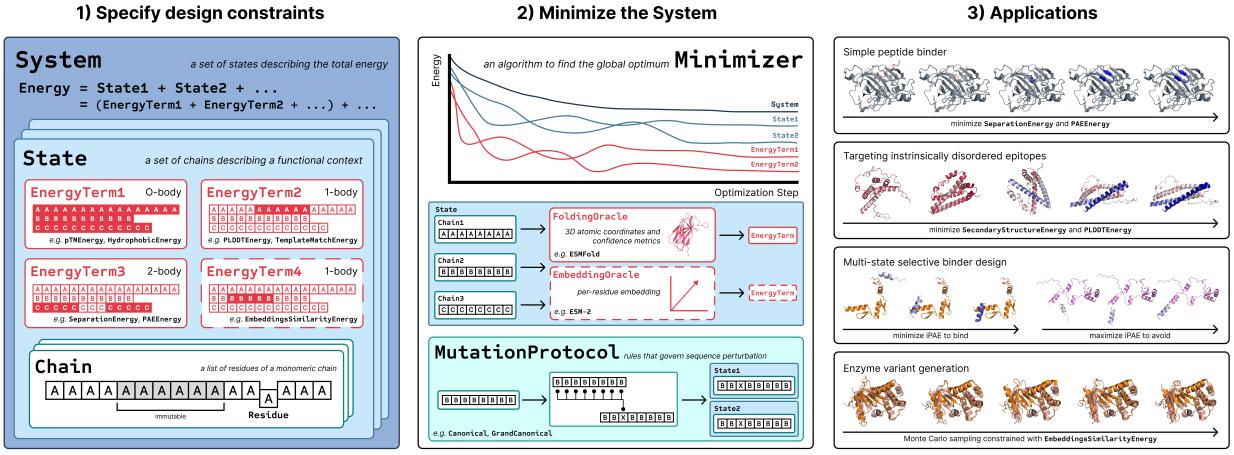

Figure 1: Conceptual schematic of the BAGEL protein design package. Modular setup of the `System` includes multiple `States`, each of which has its own set of `EnergyTerms` defining a specific design objective. Each `State` consists of `Chains`, that include either mutable or immutable `Residues`. `EnergyTerms` come in various flavours; affecting *residue groups* in a 0-body, 1-body, or 2-body fashion, and requiring the ouput of specialized `Oracles`, such as a `FoldingOracle` or an `EmbeddingOracle`. The latter is symbolized with the dashed outline. A `Minimizer` is used for the case of optimization, shown in this schematic, while another `MonteCarloMinimizer` is used for generative sampling. In this case, the `System`'s energy is optimized with the `Minimizer` through successive sequence perturbations using the `MutationProtocol`. Given the modularity, the package can be applied to a variety of applications, including the four highlighted in the figure detailed below in the report.

2 Algorithm

The energy landscape we aim to explore is defined by a system Ω and its associated energy E_Ω. Each system itself is composed of a set of states $\{S\}$, each with its own associated energy function E_S dependent on the set of chains $\{C\}_S$. A specific chain C_i, indicated with the subscript i, is a sequence of amino acid residues $\vec{\sigma}_i$, i.e., a monomeric protein, of length L_i. In mathematical form,

$$\Omega = (\{C\}, \{S\}) \tag{1}$$

$$S = (\{C\}_S \subset \{C\}, E_S);\ E_S = f(\{C\}_S) \tag{2}$$

$$C_i = \vec{\sigma}_i \in \mathcal{A}^{L_i}, \tag{3}$$

where $\mathcal{A}$ is the amino acid alphabet. We formalize the protein design problem as the sampling of the energy landscape defined by:

$$E_\Omega \ = \ \sum_{S \in \Omega} E_S$$

where a state energy function $E_S = E_S(\{C\}_S)$ is a weighted sum over N_S individual energy terms specific to each state:

$$E_S \ = \ \sum_{j}^{N_S} w_{js}\, \epsilon_{js}(\{C\}_S)$$

The energy terms ϵ_{js} can be chosen from a plethora of options, shown later in Section 3.2, but effectively impose a constraint on the design task on a particular state S, weighted by w_{js}; where j indexes over energy terms, and s over states, as the energy terms are specified for each S independently.

Sequences C_i may be *shared* or *unique* among different states. Let us make an example. Consider the case we want to design a protein C_0 that binds to both a protein C_1 but also separately to another protein C_2. The system Ω in this case can be composed of two states, let us call them A and B. Chain C_0 is present in both states, while only C_1 is part of state A and only C_2 is part of state B. We can then optimize C_0 by finding a sequence that *at the same time* would be predicted to form a stable multimer for the two independent states A and state B.

To calculate the energy terms ϵ, we introduce the general notion of an *oracle*, a black-box predictive computation (i.e., a function) mapping a set of chains $\{C\}_S$ to a useful representation y_s. These informative representations y_s are used as inputs to ϵ, thus ϵ being a function of y_s or $\{C\}_S$ interchangeably. Two important but non-exhaustive examples of classes of oracles are *folding oracles* and *embedding oracles*. Folding oracles are typically deep learning-based structure predictors,

$$f_{\text{fold}} : \{C\}_S \mapsto (\mathbf{X}, \Phi)$$

where $\mathbf{X} \in \mathbb{R}^{\sum_i 3 \times L_i}$ are three-dimensional atomic coordinates and Φ encodes model confidence metrics. Embedding oracles, in contrast, are language models providing learned high-dimensional features capturing biochemical or evolutionary context:

$$f_{\text{embed}} : \{C\}_S \mapsto (\vec{z})$$

where $\vec{z}$ is an embedding vector of dimensionality $d = (\sum_i L_i) \times N_{\text{dim}}$, that is, one embedding vector of dimensionality N_{dim} per residue in state S. Another simpler but general class of an oracle could be a function that maps a (set of) sequence(s) to a single scalar value $r \in \mathbb{R}$:

$$f_{\text{property}} : \{C\}_S \mapsto r$$

For instance, such oracle could calculate the number of polar residues in a state, or predict the unfolding temperature of a chain.

With the previous definitions, we can define protein engineering as a sampling problem, where the goal is to discover a tuple of sequences among all possible sets $\{C\}$ defining the system that samples a region of the energy landscape for which $E_\Omega(\{C\}_{\text{designed}}) = E_{\text{target}}$. When this target energy corresponds to the minimum value of the energy function E_Ω, the sampling problem becomes what one usually refers to as *optimization*.

To solve this problem, we employ the well-established Markov Chain Monte Carlo (MCMC) method [20]. At each iteration n, a proposal is generated by mutating the sequence of a chain $C_i^n \to C_i^{n+1}$ using a mutation operator $\mathcal{M}$. The total energy change is thus $\Delta E = E_\Omega(\{C\}^{n+1}) - E_\Omega(\{C\}^n)$, leading to an acceptance with the standard Metropolis criterion:

$$p_{\text{accept}} = \min\left(1, \exp\left(-\frac{\Delta E}{T}\right)\right)$$

where $T \geq 0$ is an effective temperature schedule controlling the probability of accepting proposals when $\Delta E > 0$. The sampling properties of MCMC guarantee that, for a large number of transitions, the system will relax to an equilibrium distribution that is the solution of the sampling problem. Moreover, if we replace T with an appropriate temperature schedule T_n, e.g., slowly bringing temperature from some finite initial value to zero, the MCMC method becomes an optimization method able to demonstrably find the global minimum of the energy function (although possibly for an extremely large number of steps).

We summarize the algorithm as:

- **Setup.** Specify the set of states $\{S\}$ in terms of their energy terms ϵ_{js}, weights w_{js} and the subset of the system chains $\{C\}_S$ belonging to them.
- **Input.** Initialize the initial collection of chains $\{C\}^0$.
- **Sampling loop (MCMC).** For iterations $n = 0, \ldots, N$:
  1. Propose $\{C\}^{n+1} = \mathcal{M}(\{C\}^n)$ by mutating one or more chains.
  2. Evaluate $\Delta E = E_\Omega(\{C\}^{n+1}) - E_\Omega(\{C\}^n)$.
     (a) Obtain oracle representations y_s.
     (b) Compute state energies $E_S(\{C\}_S) = \sum_j w_{js}\, \epsilon_{js}(\{y_s\})$.
     (c) Sum over states to get system energy $E_\Omega = \sum_S E_S(\{C\}_S)$.
  3. Accept the proposal with probability $p_{\text{accept}} = \min\{1, \exp(-\Delta E/T_n)\}$; otherwise retain $\{C\}^n$.
- **Output.** Return either the state $\{C\}^*_S$ corresponding to the lowest-energy encountered (optimization) or a collection of states (sampling) once the system energy has reached its equilibrium value. In this latter case, equilibrium is obtained when the system's energy reaches a plateau, and then randomly oscillates around it.

3 Modular Blocks

To support flexible, extensible, and composable protein design workflows, BAGEL is built around a set of modular, object-oriented abstractions that cleanly separate concerns such as sequence representation, mutation logic, structural prediction, and optimization/sampling. This modularity enables users to define custom protein engineering problems by composing high-level components, while also making it easy to integrate new methods and models as they emerge.

At its core, the energy is always evaluated at the `System` level, which corresponds to the total energy function E_Ω. A `System` consists of multiple `States`, each associated with its own state energy E_S, defined as a weighted sum over a set of `EnergyTerms` ϵ_{js}. A `State` can have one or more `Chains`, each representing a sequence of `Residues`, i.e. C_i. A `Residue` can be either mutable (i.e., allowed to mutate via a `MutationProtocol`) or immutable, depending on the design goal. Since not all `Oracles` support all `EnergyTerms`, and different `Oracles` may be used to compute the same type of `EnergyTerm`, each `EnergyTerm` must be explicitly associated with a specific `Oracle`. For instance, structure-based `EnergyTerms` require a `FoldingOracle`, while terms like `EmbeddingsSimilarityEnergy` require an `EmbeddingOracle`. Based on the total energy of the `System`, a `Minimizer` algorithm explores the sequence space using proposed perturbations from `MutationProtocols`.

3.1 Oracles

An `Oracle` is any algorithm supplying data required by `EnergyTerms`. For this initial release, we implement one `FoldingOracle` based on ESMFold and one `EmbeddingOracle` based on ESM-2, both using models from HuggingFace's `transformers` library [21]. To support modularity and extensibility, we developed a standalone Python package, `boileroom` [19], which provides a unified API for the inference of deep learning protein models. This abstraction enables seamless swap-in/swap-out of different models within a design pipeline, making it straightforward to benchmark alternatives in terms of both predictive accuracy and computational speed. `boileroom` integrates with the serverless GPU platform Modal, enabling scalable inference either locally or on-demand, without burdening the user with hardware or dependency management.

`ESMFold` [3]**.** Provides 3D coordinates of all heavy atoms $\mathbf{X}$ and associated confidence metrics Φ, specifically local pLDDT, pTM and PAE, which are directly consumed by structure-based `EnergyTerms` and are described in their respective paragraphs in Section 3.2. While ESMFold produces per-atom pLDDT scores, we default to using the C_α-based pLDDT as a coarse proxy for local confidence. As ESMFold does not explicitly support multimer complex prediction, one has to *trick* the model by passing a single sequence, which can, however, somehow recapitulate a multimer structure. First, one can add a linker inbetween chains, usually composed of glycines [22], which adds a relative flexibility between the individual monomer chains. Second, one can introduce a gap in the positional

embeddings inbetween the monomers, thus informing the model that the monomers should act almost independently, as they would effectively form a single chain, but would be far away in the primary structure (i.e., the sequence) The latter automatically ensures the backbones' of the monomers are not connected via a peptide bond. `boileroom` gives users the ability to both specify the exact sequence of the linker [23], as well as the skip in the positional encoding indices. We default to no glycine linker, and a positional encoding skip in the positional integers of 512. When using a glycine linker, the attention in the folding trunk is masked for all linker residues.

ESM-2 [3]. Produces an informative high-dimensional embedding vector $\vec{z}$ that can be used for downstream tasks within `EnergyTerms`. For multimers, we use the same approach as for ESMFold. We default to the 650M parameter version of the model, but leave the user the possibility to employ any of the different ESM-2 models.

3.2 Energy Terms

Below we list the implemented `EnergyTerms`. For clarity, we categorize them into sequence-only terms, structure-based terms, folding-metric terms, and vector-embedding terms. We define a *residue group* G as a set of residues that contribute to an `EnergyTerm`. All residues in a group must belong to the same `Chain`, but they are not required to be contiguous in sequence. Conceptually, we adopt an N-body formalism similar to that used in molecular simulations: zero-body terms apply globally to all residues in a system, single-body terms act on a single residue group, and two-body terms operate on a pair of residue groups. Although this classification is not explicitly enforced in the code, it serves as a helpful abstraction when constructing and reasoning about `EnergyTerms`. This N-body energy formulation and its modular implementation constitute a key difference compared to the original framework presented in Hie et al. [18], where defining certain energy terms was extremely cumbersome, especially when this definition required specifying a large number of non-contiguous regions along a single chain. More generally, their reliance on a tree graph-based representation of the system made input files increasingly difficult to construct and interpret for large systems.

3.2.1 Sequence Terms

`HydrophobicEnergy`. Measures the fraction of hydrophobic residues in a residue group G. Optionally, the magnitude of this energy can be scaled by the mean normalized solvent-accessible surface area (SASA) [24], described in detail in `SurfaceAreaEnergy` in Section 3.2.3. In this mode, hydrophobic residues that are buried (low SASA) contribute less, while surface-exposed hydrophobes contribute more, effectively focusing the penalty on exposed hydrophobic patches. Therefore, this term can be used to penalize hydrophobicity on the surface, acting more as a structural constraint, though we keep it here for simplicity.

$$E_{\text{hydrophobic}} = \frac{N_{\text{hydrophobic}}}{N_G}$$

where N_G is the number of residues in the residue group and $N_{\text{hydrophobic}}$ is the count of hydrophobic residues in the group. Hydrophobic residues are defined as valine, isoleucine, leucine, phenylalanine, methionine, and tryptophan.

`ChemicalPotentialEnergy`. Adds an energy proportional to the deviation of the number of residues from a target size, optionally raised to a power. This mimics a chemical potential term in the free energy, allowing to run an optimization with a freely floating number of residues, i.e., allowing for insertions and deletions, equivalent to sampling within the grand-canonical ensemble.

$$E_{\text{chem}} = \mu \left| N_{\text{total}} - N_{\text{target}} \right|^p$$

where μ is the chemical potential, N_{total} is the total number of residues in the `System`, N_{target} is the target number of residues, and p is an exponent parameter.

3.2.2 Folding Metric Terms

`PTMEnergy`. Penalizes low predicted Template Modeling (pTM) scores, conceived in the original AlphaFold2 paper [1]. pTM reflects the global confidence of the folding model in the predicted structure, thus it is a single scalar value over the entire `State`. The metric is based on the original TM-score [25]. This is used to achieve global structures the `FoldingOracle` is confident in.

$$E_{\text{pTM}} = -\text{pTM}$$

`PLDDTEnergy`. Penalizes low predicted Local Distance Difference Test (pLDDT) scores, which indicate local structural confidence. This confidence metric, originally also proposed in AlphaFold2 [1] is based on the Local Distance Difference Test [26]. The energy is the negative mean pLDDT over the residue group G. This term is used to promote model's confidence in a local structure.

$$E_{\text{pLDDT}} = -\frac{1}{N_G} \sum_{\alpha \in G} \text{pLDDT}_\alpha$$

where N_G is the number of residues in the group, and pLDDT_α is the pLDDT of residue α. To apply a global (0-body) constraint across the entire protein, the user can instead use `OverallPLDDTEnergy`, which applies the same formulation over all residues in the `System`.

`PAEEnergy`. Measures the uncertainty in the predicted distances between two residue groups G_1 and G_2, using the mean Predicted Alignment Error (PAE). This term encourages confident, well-defined relative positioning of residue groups, in practice, encouraging groups of residues to behave as a single rigid body, where all positions are perfectly correlated. This term is used to create protein-protein interfaces, for instance for peptide binders, and is sometimes referred to as interface PAE (iPAE), or cross PAE. It can optionally be used on a single residue group, when the pairs of residues considered are thus with the same group itself.

$$E_{\mathrm{PAE}} = \frac{1}{N_{\mathrm{pairs}}} \sum_{(\alpha,\beta)\in(G_1,G_2)} \frac{\mathrm{PAE}_{\alpha\beta}}{\mathrm{PAE}_{\mathrm{max}}}$$

where $\mathrm{PAE}_{\alpha\beta}$ is the error between residues α and β, $\mathrm{PAE}_{\mathrm{max}} \approx 30$ Å is the maximum predicted alignment error between any two residues, and N_{pairs} is the total number of residue pairs considered.

3.2.3 Structural Terms

`SurfaceAreaEnergy`. Penalizes exposure of a residue group G by computing its residues' mean solvent-accessible surface area (SASA), normalized by a reference maximum value. This energy encourages buried or compact structures and can be applied globally (0-body) or to specific residue groups (1-body). The SASA is calculated using the Shrake–Rupley "rolling probe" algorithm [24], as implemented in `Biotite` [27], with an adjustable probe radius.

$$E_{\mathrm{SASA}} = \frac{1}{N_{\mathrm{atoms,G}}} \sum_{a\in G} \frac{\mathrm{SASA}_a}{\mathrm{SASA}_{\mathrm{max}}}$$

where SASA_a is the computed SASA of atom a, $\mathrm{SASA}_{\mathrm{max}}$ is a normalization constant (by default, the full surface area of a sulfur atom) in Å units, and $N_{\mathrm{atoms,G}}$ is the number of atoms in the residue group G considered.

`SeparationEnergy`. Controls the spatial separation between two residue groups G_1 and G_2 by penalizing the Euclidean distance, in units of Å, between their backbone atoms' centroids.

$$E_{\mathrm{sep}} = \|\mathbf{c}_1 - \mathbf{c}_2\|$$

where $\mathbf{c}_1$ and $\mathbf{c}_2$ are centroids of the two residue groups respectively.

`GlobularEnergy`. Favors globular structures by minimizing the variance of distances from backbone atoms to the centroid, promoting compact and spherical distributions. It is effectively proportional to the moment of inertia of the structure.

$$E_{\mathrm{glob}} = \mathrm{std}\left(\|\mathbf{x}_a - \mathbf{c}\|\right)_G$$

where $\mathbf{x}_a$ is the position of backbone atom a, $\mathbf{c}$ is the centroid of all backbone atoms in the residue group G, and $\mathrm{std}(\cdot)_G$ represents the standard deviation over all backbone atoms a in residue group G.

`TemplateMatchEnergy`. Drives the structure of residue group G to match a provided structural template by minimizing the root mean square deviation (RMSD) between corresponding residues. RMSD is computed after optimal superposition. By default we use all the heavy atoms to obtain the alignment and the RMSD, but one can optionally choose to only use backbone atoms.

$$E_{\mathrm{template}} = \sqrt{\frac{1}{N_{\mathrm{atoms},G}} \sum_{\beta\in G} \sum_{a\in\beta} \|\mathbf{x}_{a,\beta} - \hat{\mathbf{x}}_{a,\beta}\|^2}$$

where $\mathbf{x}_{a,\beta}$ is the position of atom a in residue β , $\hat{\mathbf{x}}_{a,\beta}$ is the corresponding position in the template after optimal alignment, and $N_{\mathrm{atoms,G}}$ is the total number of atoms in the group G. The energy can also be computed using pairwise distance matrices (distograms) instead of direct atom positions.

$$E^{\mathrm{dist}}_{\mathrm{template}} = \sqrt{\frac{1}{N_{\mathrm{atoms},G}(N_{\mathrm{atoms},G}-1)/2} \sum_{\beta\in G} \sum_{\substack{a,c\in\beta \\ a<c}} \left(\|\mathbf{x}_{a,\beta} - \mathbf{x}_{c,\beta}\| - \|\hat{\mathbf{x}}_{a,\beta} - \hat{\mathbf{x}}_{c,\beta}\|\right)^2}$$

where $\|\mathbf{x}_{a,\beta} - \mathbf{x}_{c,\beta}\|$ and $\|\hat{\mathbf{x}}_{a,\beta} - \hat{\mathbf{x}}_{c,\beta}\|$ are the pairwise distances between atoms a and c of residue β in the current structure and template, respectively.

SecondaryStructureEnergy. Encourages residues in the specified group to adopt a target secondary structure (alpha-helix, beta-sheet, or coil) by penalizing deviations from the desired assignment. The secondary structure of each residue is determined using the P-SEA algorithm [28] implemented in `Biotite` [27]. The energy is defined as the fraction of residues whose assigned secondary structure does not match the target.

$$E_{\mathrm{SS}} = \frac{1}{N_G} \sum_{\alpha \in G} \mathbb{I}\left[\mathrm{SS}_\alpha \neq \mathrm{SS}_{\mathrm{target}}\right]$$

where SS_α is the assigned secondary structure of residue α (as a categorical label), $\mathrm{SS}_{\mathrm{target}}$ is the desired type (e.g., alpha-helix), $\mathbb{I}$ is the indicator function, and N_G is the number of residues in the group.

RingSymmetryEnergy. This energy term enforces N-fold symmetry, where N is the number of independent residue groups G specified as input. Symmetry among residue groups G is encouraged by minimizing the variance in distances between the groups' centroids - either for all pairs of G or only their direct neighbors. We define direct neighbours as consecutive groups G specified in the residue group list in the input. The first and last element of the list are also considered direct neighbors.

$$E_{\mathrm{sym}} = \mathrm{std}\left(d_{G_\alpha G_\beta}\right)_G$$

where $d_{G_\alpha G_\beta} = ||\mathbf{c}_{G_\alpha} - \mathbf{c}_{G_\beta}||$ is the distance between centroids of the pairs of groups G_α and G_β, computed by backbone atoms only, and $\mathrm{std}(\cdot)_G$ represents the standard deviation over all the group pairs considered. Referring back to the N-body notation, notice that this term is an arbitrary N-body term, given its flexibility to include an arbitrary amount of groups G, achieving an arbitrary N-fold symmetry.

3.2.4 Embedding Terms

EmbeddingsSimilarityEnergy. Measures the cosine similarity between learned embeddings of the current sequence and a group of reference embeddings, encouraging functional or structural similarity at the embedding level.

$$E_{\mathrm{embedding}} = 1 - \frac{1}{N_G} \sum_{\alpha \in G} \cos\theta_\alpha$$

where θ_α is the angle between the high-dimensional embeddings of residue α in the current sequence and reference sequences respectively as e_α and e_α^{ref}. $\cos\theta_\alpha = e_\alpha \cdot e_\alpha^{\mathrm{ref}}$ is then the cosine similarity and N_G is the number of residues in the group. Note that this energy term is defined so that its minimum is exactly zero when all embeddings are equal to the reference values.

3.3 Mutation Protocols

`MutationProtocols` define how to perturb sequences during optimization. Each protocol implements a strategy for proposing mutations to the sequence, such as point substitutions, insertions, or deletions. These proposals form the basis of the sampling procedure used by the `Minimizer`. All protocols operate on `Chains` within a `System`, ensuring consistency across all `States` sharing the same `Chain` object. In other words, when a chain is mutated, all states containing said chain are automatically updated.

Canonical. Implements fixed-length point-mutation moves. At each optimization step, we:

(i) pick a `Chain` with probability proportional to its number of mutable `Residues`;

(ii) draw n_{mut} `Residues` uniformly from that `Chain`;

(iii) resample their amino acid identity from a user-defined categorical distribution p_{mut}, which specifies the mutation probability for each amino acid type.

We default to p_{mut} that avoids mutating to cysteines to prevent downstream effects, such as disulfide bond formation.

GrandCanonical. Extends the canonical scheme with length-changing moves, thereby sampling a grand-canonical ensemble in which sequence length is conjugate to an effective chemical potential μ, described in the `ChemicalEnergyTerm` above. For each of the n_{mut} attempted moves, we:

(i) select a `Chain`;

(ii) draw a move type $m_{\mathrm{move}} \in \{\mathrm{substitution}, \mathrm{addition}, \mathrm{removal}\}$ with user-defined probabilities p_{type};

(iii) execute one of the following:

- substitution: identical to `Canonical`.

- addition: pick an insertion index $\alpha \in \{0, \ldots, L\}$ from $L+1$ indices, sample an amino acid type with p_{mut}, and insert a new Residue at index α.
- removal: choose a mutable Residue at index α uniformly and delete it, provided the Chain retains length greater than one (i.e., is not deleted completely from the State).

Upon addition of a new Residue, we determine whether it should *inherit* the influence of EnergyTerms from its neighbouring residues. If both neighbours (or one, at Chain boundaries) are associated with the same EnergyTerm, the new residue inherits it. If the neighbours differ in their EnergyTerm assignments, the new residue inherits from either neighbour with equal probability. As a result, each EnergyTerm must specify whether it is *inheritable*. Most terms are inheritable by default. However, some, such as TemplateMatchEnergy, are not since including newly inserted Residues in a comparison with a fixed-length reference structure would be ill-defined. Note that GrandCanonical can be used without a ChemicalEnergyTerm, in which case it implicitly runs with a zero chemical potential.

3.4 Minimizers

Minimizers define the sampling/optimization strategy used to explore sequence space (and minimize the energy function) defined over a System. Each Minimizer implements a protocol for proposing and accepting mutations over a series of steps. Below, we describe the currently available Minimizers.

MonteCarloMinimizer. Implements a general Monte Carlo sampling protocol with pluggable acceptance criteria. By default, it uses the standard Metropolis criterion $p_{n\to n+1} = \min\left(1, \exp\left(-\frac{\Delta E}{T}\right)\right)$. Users have full control over the temperature schedule, which may be constant, annealing, or arbitrarily defined. While called a *minimizer*, this class naturally generalizes to sampling applications, depending on the choice of temperature schedule, shown in the enzyme variant generation example in Section 4.4. To encourage retention of promising candidates, the best system found so far can be periodically reinstated as the current state by specifying an interval $n_{\text{best system}}$.

SimulatedAnnealing. A special case of MonteCarloMinimizer that employs a linearly decreasing temperature schedule from an initial high temperature to a final low temperature.

SimulatedTempering. A specialized variant of the MonteCarloMinimizer, this protocol employs a cyclical temperature schedule alternating between fixed low and high temperatures, T_{low} and T_{high}, across n_{cycles} cycles. Each cycle comprises $n_{\text{low,steps}}$ steps at low temperature followed by $n_{\text{high,steps}}$ at high temperature, always beginning in the low-temperature regime. To encourage retention of promising candidates, we set $n_{\text{best system}}$ to the total number of steps in one full cycle. This ensures the preservation step always occurs at the end of the high-temperature phase. Anecdotally, this strategy has led to improved convergence toward viable protein candidates.

3.4.1 Analyzers

We provide an abstract class Analyzer, which serves as the starting point to develop reliable data analysis tools for the output files. We currently support outputting the current System and the best System found so far, including their FASTA sequences, CIF structure files, and ESMFold-related metrics like pLDDT and PAE matrices, stored as a NumPy [29] text file. The exact outputs are as follows.

config.csv: A CSV file listing each State, EnergyTerm name, and its numerical weight. In practice, this file contains all the data to reproduce (at least in a statistical sense), any given experiment, thereby helping reproducibility. Nevertheless, we recommend saving the original Python script to ensure reproducibility, until a more reliable solution - such as standardized json config files - is implemented.

current/energies.csv and **best/energies.csv**: For both the running ("current") and optimal ("best") Systems, a CSV file where each row has individual energy contributions of each State at a particular step (e.g., stateA:HydrophobicEnergy).

current/state.fasta and **best/state.fasta**: FASTA files for each State, where each entry - row - represents a specific minimization step, and multiple Chains are concatenated and separated with ":".

current/structures/ and **best/structures/**: Directories containing State's structure as a CIF file per minimization step, named <State>_<Oracle>_step.cif, together with any Oracle-specific outputs (e.g., PAE or pLDDT arrays saved as .pae and .plddt files in a NumPy text format respectively).

optimization.log: A CSV file with one row per MC step, containing information from the Minimizer, including the temperature, and whether the step was accepted or rejected.

4 Examples

To illustrate the utility of BAGEL, we present design tasks that serve as reference workflows and starting points for new applications. These examples span common protein engineering use cases and demonstrate various framework features. All are reproducible, modifiable, and provided as ready-to-run templates in the GitHub repository. Simulation details, along with an input file, are included in the Appendix A.1. While not experimentally validated, these designs are valid under the assumption that ESMFold and ESM-2 yield accurate predictions. Assessing their accuracy - and its implications for real-world translation - remains an open question for future investigation.

4.1 Simple Peptide Binder

A primary application of BAGEL is designing small proteins that can bind a protein target of interest. Figure 2 shows binders to three clinically relevant targets: carbonic anhydrase IV (CA4, UniProt P22748), a protein associated to hypertension and cardiovascular disease [30], epidermal growth-factor receptor (EGFR, UniProt P00533), an important target for cancer therapeutics [31], and the dust-mite allergen Der f 7 (DERF7, UniProt Q26456), studied for the neutralisation and alleviation of allergic reactions [32]. All were designed to minimize SeparationEnergy and PAEEnergy (interface) in respect to the binder and the hotspot on the target. Taken together, these examples highlight BAGEL's ability to explore helical, beta-rich, and coil-dominated solutions from the same generic energy function.

4.2 Targeting Intrinsically Disordered Epitopes

In our earlier work [33], we showed how BAGEL's energy functions can lead to peptide binders design that induce order in an otherwise intrinsically disordered region (IDR) of a protein. We designed several such peptides for four clinically relevant targets - α-synuclein (ASYN, UniProt P37840), T-cell co-receptor CD28 (UniProt P10747), tumour suppressor p53 (P53, UniProt P04637) and Small Ubiquitin-like Modifier 1 (SUMO, UniProt P63165). Traditional drug discovery approaches require docking of molecules into rigid pockets, thus having the ability to bind disordered regions opens up plethora of applications for therapeutic modulation in disease [34], ranging from neurodegenerative diseases (ASYN) [35] through immune system regulation (CD28) [36] to cancer (P53, SUMO) [37]. Binding to disordered epitopes is achieved by enforcing high pLDDT values on the IDR with PLDDTEnergy, and optionally also promoting secondary structure with SecondaryStructureEnergy. In our original work, we also validated these designs and their binding affinities with free energy calculations. Figure 3 illustrates the key effect: the presence of the binder sharply increases the pLDDT confidence - either across the full target (ASYN), or locally at the targeted IDRs (CD28, P53, SUMO) - highlighting BAGEL's ability to stabilize otherwise disordered segments.

4.3 Multi-State Selective Peptide Binder

A key strength of BAGEL is its ability to optimize a single binder sequence against multiple structural states simultaneously. Here, we demonstrate a two-state design in which the binder must engage the mouse Zif268 zinc-finger domain (EGR1, UniProt P08046), while avoiding binding to the closely related human zinc-finger protein ZNF593 (ZNF, UniProt O00488). Species-selective binders like this could be useful in xenograft transplant settings, where tissue-specific or species-selective gene knock-out is desired [38]: for example, when humanizing the mouse immune system. We achieve this selective binding by minimizing the interface PAEEnergy for the mouse target (encouraging interaction), while simultaneously maximizing the same metric for the human off-target (discouraging binding). The latter is implemented using a negative weight w_{js}. Figure 4 summarizes the results. Multi-state optimization in BAGEL can be extended beyond this example - for instance, by designing cross-species (also known as cross-reactive) binders that target both mouse and human variants simultaneously. This flexibility is critical for therapeutic development, where candidates must succeed in both animal models and human trials. BAGEL's modular framework supports similar multi-functional design problems with ease.

4.4 Enzyme Variants with Conserved Active Site

We previously introduced a generative algorithm to design enzyme variants while preserving local function in Wu et al. [39], purely relying on sequence embeddings and without any structural information. Given the modular versatility of BAGEL, this method is now also implemented in this package. In the example, we demonstrate this with oxidoreductase (UniProt P0AEG4), where the active site (Cys30 and Cys33) is held immutable in sequence, while the remainder of the protein is allowed to mutate freely. To guide sampling, we use ESM-2 and EmbeddingsSimilarityEnergy, encouraging similarity to the reference protein in embedding space in the active site, while permitting extensive sequence drift elsewhere. In contrast to the previous applications and converging to a single optimum, we use the MonteCarloMinimizer to collect a diverse ensemble of sequences that satisfy the design

constraints at constant temperature. Figure 5 summarizes the results of this simulation. Despite large differences in the sequence and mutable structure, the catalytic site is preserved both geometrically and chemically, demonstrating the ability of embedding-guided sampling to produce viable functional diversity. As described in the original work [39], this *in silico* sampling can effectively *seed* rounds of directed evolution in the lab, allowing experimentalists to focus experimental screening on useful mutations, and avoiding deleterious mutations that would destabilize or unfold the active site altogether.

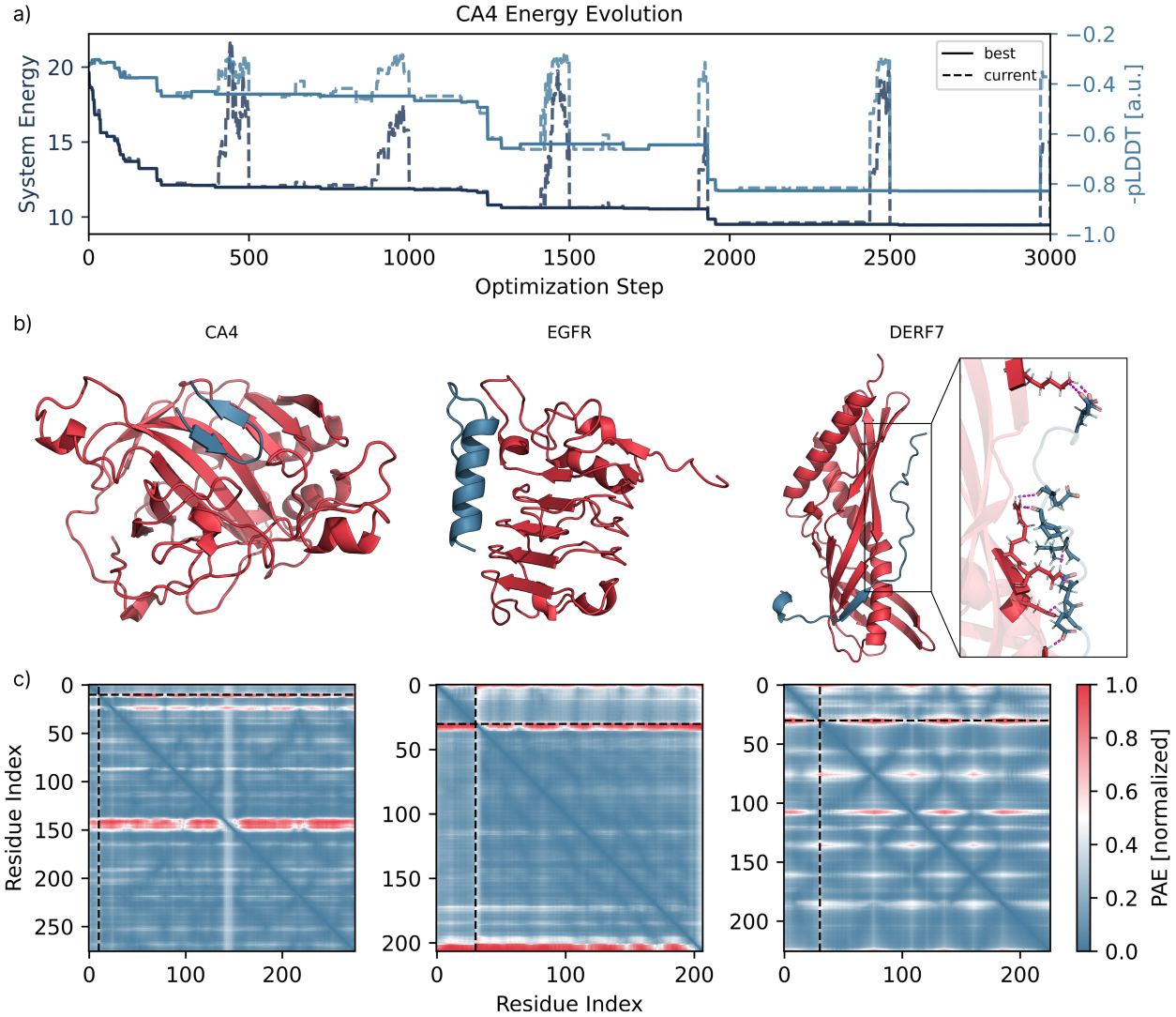

Figure 2: **Peptide binder design for three protein targets using BAGEL. a)** System energy (dark, left axis) and binder −pLDDT (light, right axis) traces during CA4 binder optimization. The optimal sequence emerged within the first ∼2,000 of 20,000 steps. Increases in energy and −pLDDT correspond to the high-temperature regime. Solid lines show the best system found; dashed lines, the current system. **b)** Final binder-target complexes for carbonic anhydrase IV (CA4), epidermal growth factor receptor (EGFR), and dust-mite allergen Der f 7 (DERF7). Binders in blue, targets in red. CA4 minimization yielded beta-sheet-rich binders despite no secondary structure restraints. For EGFR, we model only Cys329–Val506, which includes a known epitope. For DERF7, no hotspot constraints were used; minimization was over the entire target. Though the DERF7 binder appears loosely structured due to its coil-like geometry, equilibrated structures (10 ns of molecular dynamics at 310K) show plausible interfaces mediated by hydrogen bonds (magenta dashed lines, close-up). The binder wraps around the target as minimizing SeparationEnergy without a hotspot often collapses centroids - illustrating a potential failure mode that may require more careful tuning of energy terms and weights. **c)** PAE matrices for each binder-target pair, normalized to 1. Black dashed lines delineate binder (upper-left blocks) from target. All interfaces show low iPAE values: 0.242, 0.255, and 0.235, respectively.

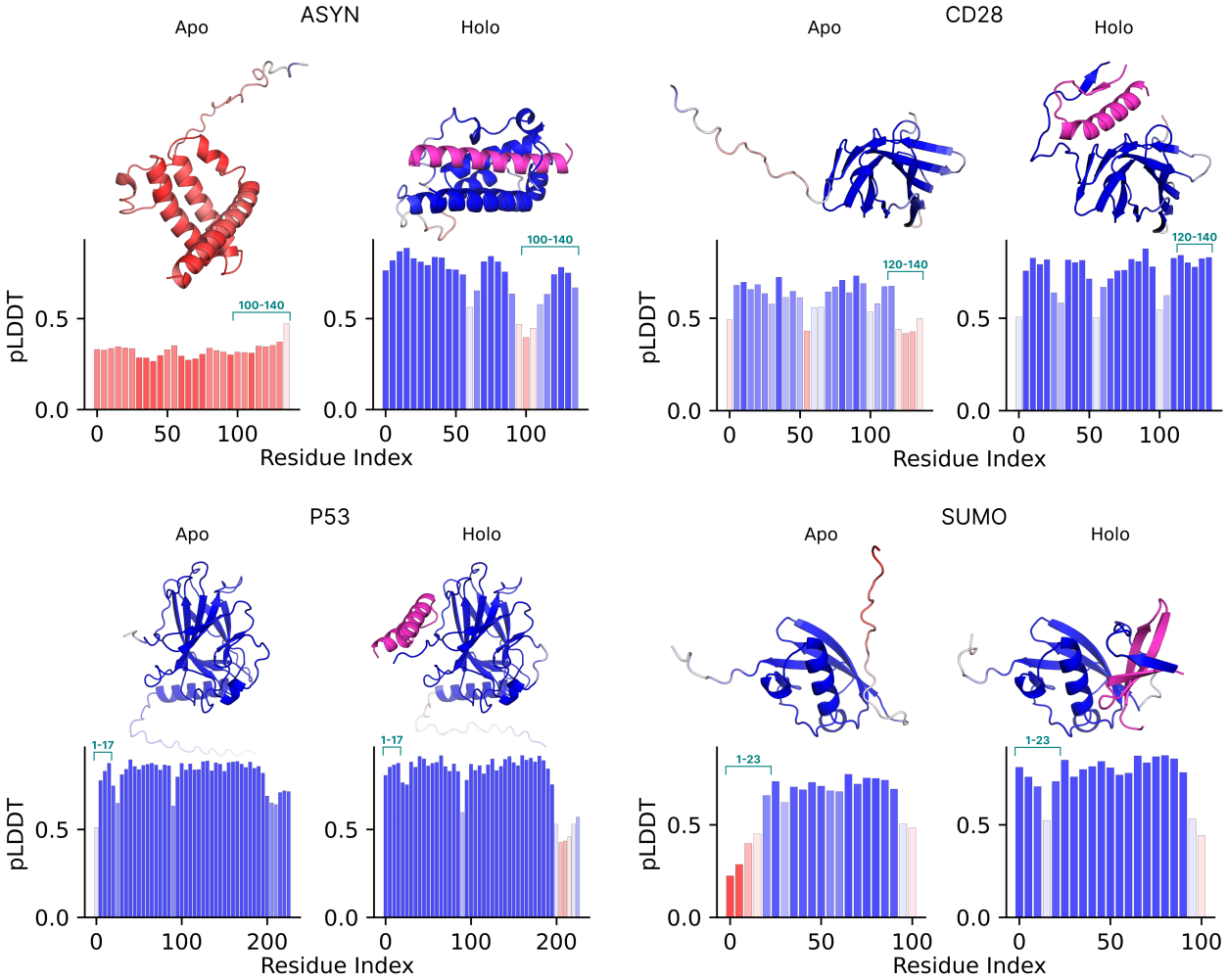

Figure 3: **Targeting of intrinsically disordered epitopes with designed binders using BAGEL.** For each of the four targets in the quadrants - α-synuclein (ASYN), CD28, p53 (P53) and SUMO-1 (SUMO) - the left sub-panel shows the predicted *apo* structure, while the right sub-panel (*holo*) includes the magenta peptide binder. Backbones are colored by per-residue pLDDT confidence (red → white→ blue corresponds to 0.0 to 1.0 transition). The bar plots beneath each structure display the same pLDDT values averaged over consecutive 5-residue windows, illustrating the local gain in structural confidence upon the presence of the binder targeting the IDR epitopes highlighted in cyan. Note that the horizontal axis is 0-indexed, while the epitope residue numbers in cyan are 1-indexed. ASYN shows a broad increase in pLDDT across the whole target, while P53 shows a significant reduction in pLDDT on the opposite end of the sequence (beyond residue 200) from the targeted epitope. CD28 and SUMO show the expected behavior, with increased pLDDT localized to the epitope region. Figure adapted from Lála et al. [33].

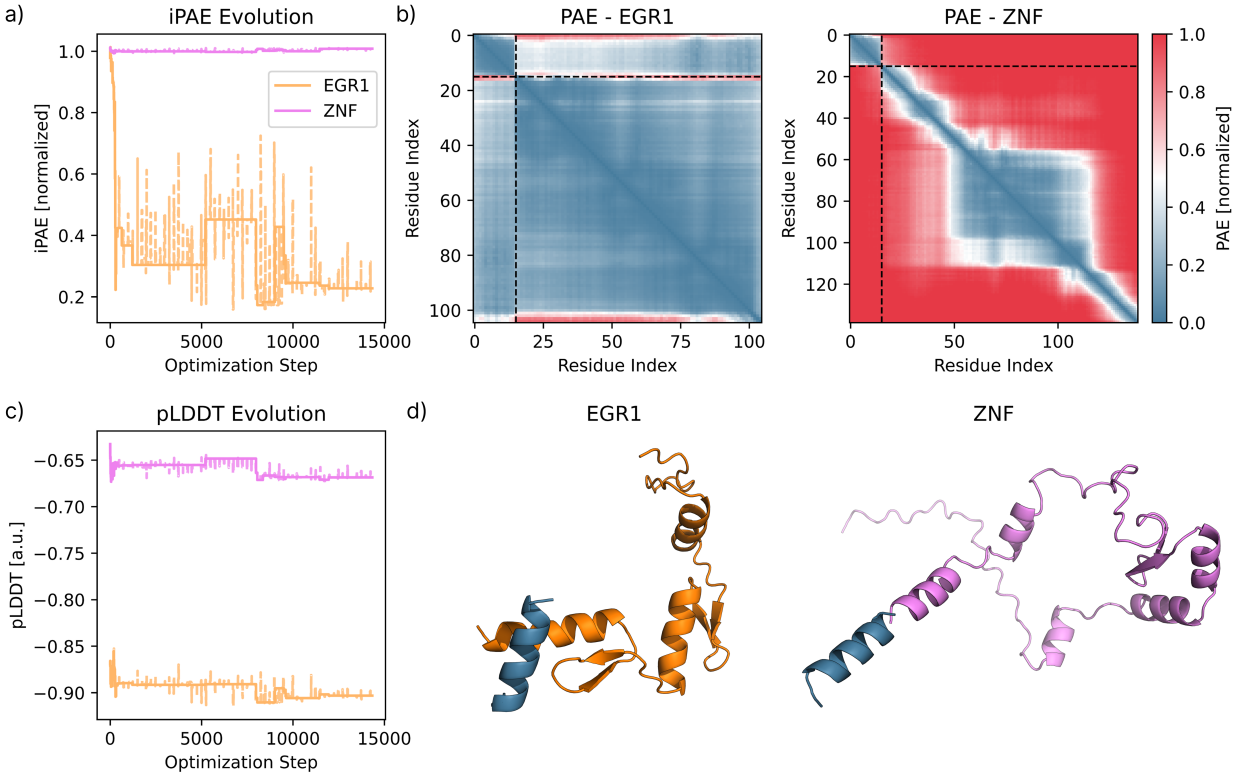

Figure 4: **Selective binder design against targets, avoiding off-targets using BAGEL. a)** Evolution of the mean interface (iPAE) for the binding (orange) and non-binding (violet) states between the binder and the target. Solid lines are the best energies, and dashed lines are the current energies from the MC trajectory. **b)** Final PAE matrix for the binding (EGR1) and non-binding (ZNF) states; black dashed lines separate binder (upper-left block) from target residues. The final mean iPAE of the binding and non-binding states are 0.23 and 1.0 respectively (normalized units), observable in the PAE matrices. **c)** Evolution from the MC optimization of the binder's mean pLDDT in each state, colors are the same as in a). **d)** Structure of the binding and non-binding complex (dark blue = binder, orange = target, violet = target to avoid). From further visual inspection not shown here, only the binding pair shows a well-packed contact surface, while the non-binding state renders an implausible, sterically clashing structure. The pLDDT of the ZNF binder is relative low (0.66 as shown in a)), while the iPAE is relatively high (shown in b)). All of these hint at a low chance of the binder interacting with the off-target ZNF.

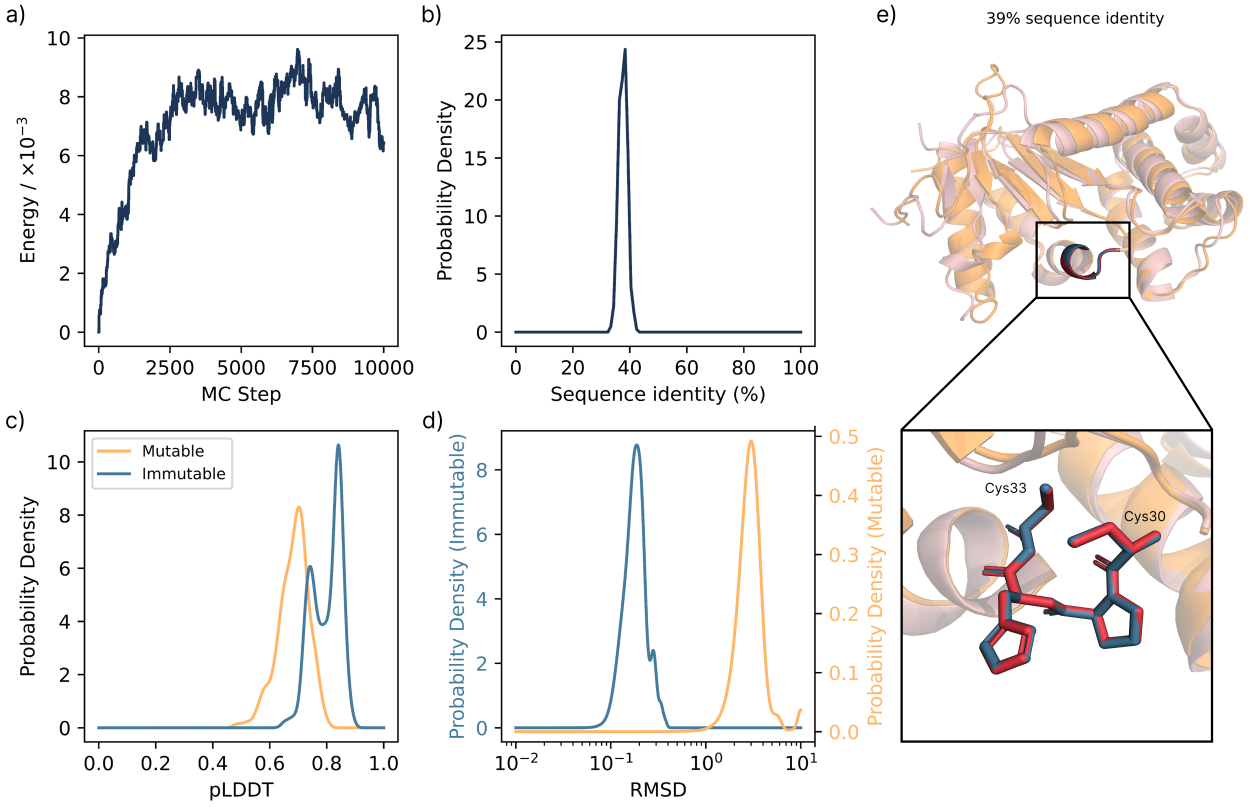

Figure 5: **Enzyme (oxidoreductase) variant generation with BAGEL. a)** Energy evolution of the MC trajectory to sample new enzyme variants. We define the first 3,000 steps as the *transient* region and analyze the subsequent *equilibrium* sampling from steps 3,000 to 10,000, collecting 1,286 unique variants as a result. **b)** Probability density of the sequence identity to the original sequence. **c)** Probability density of pLDDT values comparing the residues within the mutable and immutable groups. Immutable residues, i.e., the active site, show significantly higher pLDDT values. **d)** Probability density of RMSD to the original sequence. All RMSD values are computed using folded structures with ESMFold, including the original sequence. Mutable residues' RMSD is computed on the backbone atoms, immutable residues' RMSD is computed on all the heavy atoms of the active site, i.e., Cys-30 and Cys-33. Structures are superimposed to minimize the RMSD of the active site. Despite the relatively low sequence identity, the RMSD of the active site remains low, and the pLDDT sufficiently high. **e)** Example of a variant (blue = original, red = variant), including a close-up of residues Cys-30 to Cys-33. For this variant, the RMSD of the immutable and mutable regions are 0.11 Å and 2.7 Å respectively.

5 Future Outlook

We built BAGEL with the clear goal not only to withstand the test of time, but to actually mature and improve as time passes, as more accurate and faster algorithms evaluating properties of proteins become available, or with advancements in computing hardware. Nevertheless, the molecular community's skepticism - summarized in the aphorism "garbage in, garbage out" - remains valid, and BAGEL's prediction can only be as good as the currently available tools allow. In this regard, we recognize several current limitations: some internal - directly stemming from our choices and omissions for this initial release - and some external - dependent on the accuracy of current models, especially when considering truly *de novo* protein sequences and multimers, and on their ability to provide accurate predictions, e.g., of the binding interactions between different proteins. The ongoing development of BAGEL is envisioned to address these limitations to ensure continuous relevance and adaptability in the field of protein engineering.

Improved `Oracle` Models. Future versions will integrate additional state-of-the-art `FoldingOracles` - a non-exhaustive list to benchmark includes RGN2 [40], AlphaFold2/3 [1, 41], Chai-1 [42], and Boltz-1/2 [43, 44]. In this regard, unifying the inference access with `boileroom` will enable direct comparison of the models' accuracy and computational speed. Given the sequential nature of MC, computationally faster models such as MiniFold [45] also need to be considered, even at the expense of marginal inaccuracy. Moreover, one could only calculate the change in mutated embeddings instead of recomputing their value from scratch (an interpolation of a Car-Parrinello-like approach to embeddings [46]), potentially leading to increased inference speed. We also point out that different models excel in different scenarios: for example, RGN2 has been shown to outperform other models for orphan proteins structure prediction [40]. Therefore, one approach is to employ an ensemble of `FoldingOracles` with a weighted consensus; although it remains unclear whether such consensus improves design outcomes [47]. Once additional `FoldingOracles` are implemented inside BAGEL, such an approach can be readily implemented within the current codebase. Regarding `EmbeddingOracles`, we plan to explore models including AMPLIFY [48] and ProtGen [49], as well as protein language models trained explicitly to provide improved descriptions of protein-protein interactions such as the recently released MINT [50]. In general, the modular nature of BAGEL will help us leverage the continuous improvements of models and hardware in the upcoming months and years; and given that we do not rely on gradient-based optimization, the implementation is more flexible to swap out the underlying models, enabling rapid and convenient experimentation with different models.

Expanded `Oracle` Suite. The modular design philosophy also motivates us to introduce new types of oracles, helping to provide more physically accurate information on interactions and dynamics. For instance, the promise of emerging sequence-to-ensemble predictors, such as BioEmu [51], AlphaFlow [52] or CryoBoltz [53], could be used to compute `EnergyTerms` over ensembles as opposed to static structures, with the goal of better capturing the *true* thermodynamic behavior of biomolecules. A more physically-grounded description will come from introducing molecular dynamics (MD)-based `Oracles`. Simple energy minimization with a physical force field, such as AMBER [54], would filter out unstable, or sterically clashing structures. Furthermore, a short MD simulation measuring the RMSD between the initial and equilibrated structures could be used to estimate the stability of the complex. Finally, the most computationally demanding approach would involve techniques such as free energy perturbation [55] to provide the most correlated relative binding affinity prediction as an `EnergyTerm` to guide the MC trajectory in search of the optimal design.

Advanced `Minimizer` Algorithms. Although currently we only implemented different forms of Monte Carlo sampling with point mutations to explore the energy landscape, see Section 2, arbitrary sampling/optimization algorithms can be employed. In the near future, we plan to benchmark genetic algorithms (GAs) and carefully evaluate their potential to improve sequence space exploration efficiently. We expect that this endeavour will require careful consideration of the crossover mutation algorithms used, especially for shorter sequences. Moreover, we plan to leverage the scalability of `boileroom` and implement parallel tempering, hence truly taking advantage of parallelization, which may help alleviate the inherent limitations of sequential, single-replica MC. Based on benchmarking the accuracy and speed of different `FoldingOracles`, a low-hanging fruit that could also be rapidly implemented within the existing codebase is a multi-resolution sampling scheme [56, 57]. First, a cheaper model, such as RGN2 or MiniFold, is initially used to generate a long Markov Chain of n steps to move further and more rapidly in the energy landscape. Second, a more expensive (and, in principle, more accurate) model, such as AlphaFold3, is used to accept or reject the whole chain. This would be especially useful for systems with very long decorrelation times, when many potential iterations of the MC algorithm are wastefully spent sampling very similar candidates corresponding to the same basin of the energy landscape.

New `EnergyTerms`. Solving different design problems requires the use of `EnergyTerms` to drive the system in the correct region of sequence space. While in the current implementation of BAGEL we implemented useful terms to tackle various classical protein design tasks such as motif templating and binder design, there is still much work to be done. Two different but complementary directions are the introduction of terms that a) quantitively improve desired correlations for already implemented design objectives, or b) qualitatively widen the range of potential design tasks. In the first group, the highest-priority advancement includes implementation of new `EnergyTerms` that more directly correlate

with the interaction between two groups of residues, possibly calculated using Oracles specialized in reproducing different aspects of protein-protein interactions, such as the previously discussed MINT [50], or PESTO [58]. In the second group, the choice is only limited by our creativity and the design problem we want to solve. For the next iteration of BAGEL, we plan to implement terms that can be used to enforce the positioning of specific amino acids in user-defined positions, e.g., to create addressable protein scaffolds. Combining such terms with excluded volume terms that create void regions, for example, could be used to create pockets to co-localise drugs or catalysts in a specific protein patch, which could be useful to design enzymes for pro-drugs activation [59]. Similarly, EnergyTerms that measure the propensity for a given protein surface to interact with other class of molecules, such as nucleic acids, could be used to design transcription factors with known binding patches. In this endeavour, we make an open call to the community to contribute new EnergyTerms relevant to their research interests, thus increasing the breadth of problems addressable with BAGEL.

Choice of EnergyTerms. One current limitation is the need for the user to correctly specify the energy terms and their associated weights. Although this provides a tremendous flexibility, allowing a wide array of designs, it leads to a non-trivial process for the user input. Follow-up publications and the examples above hopefully illustrate the exemplary applications, serving as effective tutorials for novels users. Moreover, if one is able to identify the most important final metric to optimize (e.g., iPAE in some cases), we believe employing methods such as bayesian optimization, or reinforcement learning combined with large language model-based agents could potentially simplify the user experience for the protein engineer, while also aiding in the efficient exploration of the hyperparameter space.

Future of boileroom. Managing dependencies across diverse deep learning models often results in complex and fragile dependency graphs. We introduced boileroom to alleviate the dependency management, running each model in an isolated, containerized environment on Modal. Nevertheless, this introduces internet connection requirements and network latency, which we hope to tackle by providing better support for local execution of these models. We can achieve this by employing similar logic to Modal, where containers are spun up through conda, Docker or Apptainer, and thus run these models in isolated environments. We are also considering integrating with NVIDIA's BioNemo and NIM frameworks, which are, however, still in their infancy in terms of the breadth of bio-related models provided. Lastly, boileroom's API should be unified, abstracting a consistent Protein class to ensure consistent and predictable outputs from the interface.

Biomolecular Modality. Long-term additions might include the introduction of non-canonical amino acids, e.g., with RareFold [60], or nucleotides, if models supporting nucleic acids (NAs) - such as Boltz-2 and AlphaFold3 - reach a sufficiently translatable accuracy in predicting protein-NA interactions. Non-canonical-based peptides have been growing in popularity thanks to their stability and bioavailability [61], while targeting nucleic acids could lead to novel gene-editing [62] or cell reprogramming tools [63]. This would require a more detailed refactoring of the Chain and Residue logic, however, the overall codebase's philosophy and components would stay the same. At the same time, we highlight that FoldingOracles or EmbeddingOracles trained for systems made exclusively of DNA or RNA [64] could be directly implemented and used to design nucleotide-based structures within BAGEL with very minimal changes. Although they would probably require implementation of *ad hoc* EnergyTerms, such an endeavour would immediately expand BAGELs capability to design a new class of systems.

Community Engagement and Contribution. We aim to cultivate a vibrant community around BAGEL by encouraging contributions via GitHub pull requests. Specifically, we welcome implementations of new Oracles into boileroom, new Minimizer algorithms or adaptations of existing ones, and detailed benchmarking studies using the suite of available components. Our principal ambition ultimately remains grounded in experimental validation. We thus aim to rigorously test the practical efficacy of BAGEL-designed proteins in laboratory conditions. This will help establish BAGEL not only as a computational tool, but as a reliable platform for practical biological innovation. Accordingly, we extend an open call to experimentalists seeking *in silico* designed peptides: the provided templates on GitHub may be readily adapted for specific applications, and we welcome direct contact to discuss potential collaborations.

6 Conclusion

In this report, we introduced BAGEL, an open-source, modular framework for protein engineering through energy landscape exploration. By formalizing protein design as the sampling of a customizable energy function and leveraging powerful deep learning model oracles, BAGEL addresses the limitations of existing pipelines and expands the scope of computational protein design. The package supports the definition of arbitrary design objectives without requiring differentiability, while also enabling multi-objective optimization across multiple design states. We demonstrated its versatility through diverse applications, ranging from simple peptide design to multi-state selective binder design and enzyme variant generation. Looking ahead, our immediate goal is to experimentally validate BAGEL-designed proteins within the coming year. We encourage the broader scientific community to engage actively: whether by

contributing novel functionalities, conducting rigorous benchmarking, or pursuing experimental validation. Thanks to its model-agnostic philosophy and a collaborative development approach, we anticipate that BAGEL will continue to benefit from rapid advances in deep learning and protein modeling, offering a powerful alternative to existing design pipelines and accelerating the transition from computation to real-world biological innovation.

7 Author Contributions

S.A.-U. conceived the project. J.L., A.A.-S. and S.A.-U. all contributed to the BAGEL codebase. J.L. wrote the boileroom backend package. J.L. and S.A.-U. wrote and edited the manuscript.

8 Acknowledgements

We thank Shanil Panara, Dr Daniele Visco, Arnav Cheruku and Harsh Agrawal for fruitful conversations discussing the implementation of the codebase and design pipeline. J.L. acknowledges the President's PhD scholarship at Imperial College London for funding. We acknowledge computational resources and support provided by the Imperial College Research Computing Service (http://doi.org/10.14469/hpc/2232). We are grateful to the UK Materials and Molecular Modelling Hub for computational resources, which is partially funded by EPSRC (EP/T022213/1, EP/W032260/1 and EP/P020194/1). This research was also supported by grants from NVIDIA and utilized NVIDIA's GPU access through the Academic Grant for Simulation and Modeling.

9 Code availability

The initial public release of BAGEL v0.1 is available at https://github.com/softnanolab/bagel, including all the templates used in this manuscript to reproduce the results. The exact v0.1.0 release with the compatible templates is available on Zenodo [65]. The accompanying package boileroom is also publicly available at https://github.com/softnanolab/boileroom.

References

[1] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, July 2021. ISSN 1476-4687. doi:10.1038/s41586-021-03819-2. URL http://dx.doi.org/10.1038/s41586-021-03819-2.

[2] Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N. Kinch, R. Dustin Schaeffer, Claudia Millán, Hahnbeom Park, Carson Adams, Caleb R. Glassman, Andy DeGiovanni, Jose H. Pereira, Andria V. Rodrigues, Alberdina A. van Dijk, Ana C. Ebrecht, Diederik J. Opperman, Theo Sagmeister, Christoph Buhlheller, Tea Pavkov-Keller, Manoj K. Rathinaswamy, Udit Dalwadi, Calvin K. Yip, John E. Burke, K. Christopher Garcia, Nick V. Grishin, Paul D. Adams, Randy J. Read, and David Baker. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, August 2021. ISSN 1095-9203. doi:10.1126/science.abj8754. URL http://dx.doi.org/10.1126/science.abj8754.

[3] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, March 2023. ISSN 1095-9203. doi:10.1126/science.ade2574. URL http://dx.doi.org/10.1126/science.ade2574.

[4] Zhenyu Yang, Xiaoxi Zeng, Yi Zhao, and Runsheng Chen. Alphafold2 and its applications in the fields of biology and medicine. *Signal Transduction and Targeted Therapy*, 8(1), March 2023. ISSN 2059-3635. doi:10.1038/s41392-023-01381-z. URL http://dx.doi.org/10.1038/s41392-023-01381-z.

[5] Alfredo Quijano-Rubio, Hsien-Wei Yeh, Jooyoung Park, Hansol Lee, Robert A. Langan, Scott E. Boyken, Marc J. Lajoie, Longxing Cao, Cameron M. Chow, Marcos C. Miranda, Jimin Wi, Hyo Jeong Hong, Lance Stewart,

Byung-Ha Oh, and David Baker. De novo design of modular and tunable protein biosensors. *Nature*, 591(7850): 482–487, January 2021. ISSN 1476-4687. doi:10.1038/s41586-021-03258-z. URL http://dx.doi.org/10.1038/s41586-021-03258-z.

[6] Nathan M. Ennist, Steven E. Stayrook, P. Leslie Dutton, and Christopher C. Moser. Rational design of photosynthetic reaction center protein maquettes. *Frontiers in Molecular Biosciences*, 9, September 2022. ISSN 2296-889X. doi:10.3389/fmolb.2022.997295. URL http://dx.doi.org/10.3389/fmolb.2022.997295.

[7] Martin Pacesa, Lennart Nickel, Christian Schellhaas, Joseph Schmidt, Ekaterina Pyatova, Lucas Kissling, Patrick Barendse, Jagrity Choudhury, Srajan Kapoor, Ana Alcaraz-Serna, Yehlin Cho, Kourosh H. Ghamary, Laura Vinué, Brahm J. Yachnin, Andrew M. Wollacott, Stephen Buckley, Adrie H. Westphal, Simon Lindhoud, Sandrine Georgeon, Casper A. Goverde, Georgios N. Hatzopoulos, Pierre Gönczy, Yannick D. Muller, Gerald Schwank, Daan C. Swarts, Alex J. Vecchio, Bernard L. Schneider, Sergey Ovchinnikov, and Bruno E. Correia. Bindcraft: one-shot design of functional protein binders. October 2024. doi:10.1101/2024.09.30.615802. URL http://dx.doi.org/10.1101/2024.09.30.615802.

[8] James U. Bowie, Roland Lüthy, and David Eisenberg. A method to identify protein sequences that fold into a known three-dimensional structure. *Science*, 253(5016):164–170, July 1991. ISSN 1095-9203. doi:10.1126/science.1853201. URL http://dx.doi.org/10.1126/science.1853201.

[9] Ivan Anishchenko, Samuel J. Pellock, Tamuka M. Chidyausiku, Theresa A. Ramelot, Sergey Ovchinnikov, Jingzhou Hao, Khushboo Bafna, Christoffer Norn, Alex Kang, Asim K. Bera, Frank DiMaio, Lauren Carter, Cameron M. Chow, Gaetano T. Montelione, and David Baker. De novo protein design by deep network hallucination. *Nature*, 600(7889):547–552, December 2021. ISSN 1476-4687. doi:10.1038/s41586-021-04184-w. URL http://dx.doi.org/10.1038/s41586-021-04184-w.

[10] Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I. M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, July 2023. ISSN 1476-4687. doi:10.1038/s41586-023-06415-8. URL http://dx.doi.org/10.1038/s41586-023-06415-8.

[11] J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet, R. J. de Haas, N. Bethel, P. J. Y. Leung, T. F. Huddy, S. Pellock, D. Tischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, A. K. Bera, N. P. King, and D. Baker. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, October 2022. ISSN 1095-9203. doi:10.1126/science.add2187. URL http://dx.doi.org/10.1126/science.add2187.

[12] Casper A. Goverde, Martin Pacesa, Nicolas Goldbach, Lars J. Dornfeld, Petra E. M. Balbi, Sandrine Georgeon, Stéphane Rosset, Srajan Kapoor, Jagrity Choudhury, Justas Dauparas, Christian Schellhaas, Simon Kozlov, David Baker, Sergey Ovchinnikov, Alex J. Vecchio, and Bruno E. Correia. Computational design of soluble and functional membrane protein analogues. *Nature*, 631(8020):449–458, June 2024. ISSN 1476-4687. doi:10.1038/s41586-024-07601-y. URL http://dx.doi.org/10.1038/s41586-024-07601-y.

[13] Justas Dauparas, Gyu Rie Lee, Robert Pecoraro, Linna An, Ivan Anishchenko, Cameron Glasscock, and David Baker. Atomic context-conditioned protein sequence design using ligandmpnn. *Nature Methods*, 22 (4):717–723, March 2025. ISSN 1548-7105. doi:10.1038/s41592-025-02626-1. URL http://dx.doi.org/10.1038/s41592-025-02626-1.

[14] John B. Ingraham, Max Baranov, Zak Costello, Karl W. Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana M. Lord, Christopher Ng-Thow-Hing, Erik R. Van Vlack, Shan Tie, Vincent Xue, Sarah C. Cowles, Alan Leung, João V. Rodrigues, Claudio L. Morales-Perez, Alex M. Ayoub, Robin Green, Katherine Puentes, Frank Oplinger, Nishant V. Panwar, Fritz Obermeyer, Adam R. Root, Andrew L. Beam, Frank J. Poelwijk, and Gevorg Grigoryan. Illuminating protein space with a programmable generative model. *Nature*, 623(7989):1070–1078, November 2023. ISSN 1476-4687. doi:10.1038/s41586-023-06728-8. URL http://dx.doi.org/10.1038/s41586-023-06728-8.

[15] Sidney Lyayuga Lisanza, Jacob Merle Gershon, Samuel W. K. Tipps, Jeremiah Nelson Sims, Lucas Arnoldt, Samuel J. Hendel, Miriam K. Simma, Ge Liu, Muna Yase, Hongwei Wu, Claire D. Tharp, Xinting Li, Alex Kang, Evans Brackenbrough, Asim K. Bera, Stacey Gerben, Bruce J. Wittmann, Andrew C. McShan, and David Baker. Multistate and functional protein design using rosettafold sequence space diffusion. *Nature Biotechnology*, September 2024. ISSN 1546-1696. doi:10.1038/s41587-024-02395-w. URL http://dx.doi.org/10.1038/s41587-024-02395-w.

[16] Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J. Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q. Tran, Jonathan Deaton, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul S. Molina, Neil Thomas, Yousuf A. Khan, Chetan Mishra, Carolyn Kim, Liam J. Bartie, Matthew Nemeth, Patrick D. Hsu, Tom Sercu, Salvatore Candido, and Alexander Rives. Simulating 500 million years of evolution with a language model. *Science*, 387(6736):850–858, February 2025. ISSN 1095-9203. doi:10.1126/science.ads0018. URL http://dx.doi.org/10.1126/science.ads0018.

[17] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15), April 2021. ISSN 1091-6490. doi:10.1073/pnas.2016239118. URL http://dx.doi.org/10.1073/pnas.2016239118.

[18] Brian Hie, Salvatore Candido, Zeming Lin, Ori Kabeli, Roshan Rao, Nikita Smetanin, Tom Sercu, and Alexander Rives. A high-level programming language for generative protein design. December 2022. doi:10.1101/2022.12.21.521526. URL http://dx.doi.org/10.1101/2022.12.21.521526.

[19] Jakub Lála. boileroom: serverless protein prediction models, 2025. URL https://github.com/jakublala/boileroom.

[20] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953. ISSN 1089-7690. doi:10.1063/1.1699114. URL http://dx.doi.org/10.1063/1.1699114.

[21] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[22] Richard Evans, Michael O'Neill, Alexander Pritzel, Natasha Antropova, Andrew Senior, Tim Green, Augustin Žídek, Russ Bates, Sam Blackwell, Jason Yim, Olaf Ronneberger, Sebastian Bodenstein, Michal Zielinski, Alex Bridgland, Anna Potapenko, Andrew Cowie, Kathryn Tunyasuvunakool, Rishub Jain, Ellen Clancy, Pushmeet Kohli, John Jumper, and Demis Hassabis. Protein complex prediction with alphafold-multimer. October 2021. doi:10.1101/2021.10.04.463034. URL http://dx.doi.org/10.1101/2021.10.04.463034.

[23] Shuxian Zou, Hui Li, Shentong Mo, Xingyi Cheng, Eric Xing, and Le Song. Linker-tuning: Optimizing continuous prompts for heterodimeric protein prediction, 2023. URL https://arxiv.org/abs/2312.01186.

[24] A. Shrake and J.A. Rupley. Environment and exposure to solvent of protein atoms. lysozyme and insulin. *Journal of Molecular Biology*, 79(2):351–371, September 1973. ISSN 0022-2836. doi:10.1016/0022-2836(73)90011-9. URL http://dx.doi.org/10.1016/0022-2836(73)90011-9.

[25] Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, October 2004. ISSN 1097-0134. doi:10.1002/prot.20264. URL http://dx.doi.org/10.1002/prot.20264.

[26] Valerio Mariani, Marco Biasini, Alessandro Barbato, and Torsten Schwede. lddt: a local superposition-free score for comparing protein structures and models using distance difference tests. *Bioinformatics*, 29(21):2722–2728, August 2013. ISSN 1367-4811. doi:10.1093/bioinformatics/btt473. URL http://dx.doi.org/10.1093/bioinformatics/btt473.

[27] Patrick Kunzmann and Kay Hamacher. Biotite: a unifying open source computational biology framework in python. *BMC Bioinformatics*, 19(1), October 2018. ISSN 1471-2105. doi:10.1186/s12859-018-2367-z. URL http://dx.doi.org/10.1186/s12859-018-2367-z.

[28] G. Labesse, N. Colloc'h, J. Pothier, and J.-P. Mornon. P-sea: a new efficient assignment of secondary structure from cα trace of proteins. *Bioinformatics*, 13(3):291–295, 1997. ISSN 1460-2059. doi:10.1093/bioinformatics/13.3.291. URL http://dx.doi.org/10.1093/bioinformatics/13.3.291.

[29] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with numpy. *Nature*, 585(7825):357–362, September 2020. ISSN 1476-4687. doi:10.1038/s41586-020-2649-2. URL http://dx.doi.org/10.1038/s41586-020-2649-2.

[30] Fabrizio Carta and Claudiu T Supuran. Diuretics with carbonic anhydrase inhibitory action: a patent and literature review (2005 – 2013). *Expert Opinion on Therapeutic Patents*, 23(6):681–691, March 2013. ISSN 1744-7674. doi:10.1517/13543776.2013.780598. URL http://dx.doi.org/10.1517/13543776.2013.780598.

[31] Salvador Guardiola, Monica Varese, Macarena Sánchez-Navarro, and Ernest Giralt. A third shot at egfr: New opportunities in cancer therapy. *Trends in Pharmacological Sciences*, 40(12):941–955, December 2019. ISSN 0165-6147. doi:10.1016/j.tips.2019.10.004. URL http://dx.doi.org/10.1016/j.tips.2019.10.004.

[32] Mirela Curin, Huey-Jy Huang, Tetiana Garmatiuk, Sandra Gutfreund, Yvonne Resch-Marat, Kuan-Wei Chen, Kerstin Fauland, Walter Keller, Petra Zieglmayer, René Zieglmayer, Patrick Lemell, Friedrich Horak, Wolfgang Hemmer, Margarete Focke-Tejkl, Sabine Flicker, Susanne Vrtala, and Rudolf Valenta. Ige epitopes of the house dust mite allergen der p 7 are mainly discontinuous and conformational. *Frontiers in Immunology*, 12, June 2021. ISSN 1664-3224. doi:10.3389/fimmu.2021.687294. URL http://dx.doi.org/10.3389/fimmu.2021.687294.

[33] Jakub Lála, Daniele Visco, and Stefano Angioletti-Uberti. Programming co-folding to design binders for intrinsically disordered epitopes. In *ICLR 2025 Workshop on Generative and Experimental Perspectives for Biomolecular Design*, 2025.

[34] M Madan Babu, Robin van der Lee, Natalia Sanchez de Groot, and Jörg Gsponer. Intrinsically disordered proteins: regulation and disease. *Current Opinion in Structural Biology*, 21(3):432–440, June 2011. ISSN 0959-440X. doi:10.1016/j.sbi.2011.03.011. URL http://dx.doi.org/10.1016/j.sbi.2011.03.011.

[35] David Sulzer and Robert H. Edwards. The physiological role of α-synuclein and its relationship to parkinson's disease. *Journal of Neurochemistry*, 150(5):475–486, July 2019. ISSN 1471-4159. doi:10.1111/jnc.14810. URL http://dx.doi.org/10.1111/jnc.14810.

[36] Sijing Xia, Qin Chen, and Bing Niu. Cd28: A new drug target for immune disease. *Current Drug Targets*, 21 (6):589–598, April 2020. ISSN 1389-4501. doi:10.2174/1389450120666191114102830. URL http://dx.doi.org/10.2174/1389450120666191114102830.

[37] Antti Kukkula, Veera K. Ojala, Lourdes M. Mendez, Lea Sistonen, Klaus Elenius, and Maria Sundvall. Therapeutic potential of targeting the sumo pathway in cancer. *Cancers*, 13(17):4402, August 2021. ISSN 2072-6694. doi:10.3390/cancers13174402. URL http://dx.doi.org/10.3390/cancers13174402.

[38] Jianxuan Zou, Richard J. Jones, Hua Wang, Isere Kuiatse, Fazal Shirazi, Elisabet E. Manasanch, Hans C. Lee, Robert Sullivan, Leah Fung, Normand Richard, Paul Erdman, Eduardo Torres, David Hecht, Imelda Lam, Brooke McElwee, Aparajita H. Chourasia, Kyle W. H. Chan, Frank Mercurio, David I. Stirling, and Robert Z. Orlowski. The novel protein homeostatic modulator btx306 is active in myeloma and overcomes bortezomib and lenalidomide resistance. *Journal of Molecular Medicine*, 98(8):1161–1173, July 2020. ISSN 1432-1440. doi:10.1007/s00109-020-01943-6. URL http://dx.doi.org/10.1007/s00109-020-01943-6.

[39] Adam Wu, Quentin Trolliet, Abhinav Rajendran, Jakub Lála, and Stefano Angioletti-Uberti. Mind the gap: An embedding guide to safely travel in sequence space. *bioRxiv*, 2025. doi:10.1101/2025.06.19.660524. URL https://www.biorxiv.org/content/10.1101/2025.06.19.660524v1.

[40] Ratul Chowdhury, Nazim Bouatta, Surojit Biswas, Christina Floristean, Anant Kharkar, Koushik Roy, Charlotte Rochereau, Gustaf Ahdritz, Joanna Zhang, George M. Church, Peter K. Sorger, and Mohammed AlQuraishi. Single-sequence protein structure prediction using a language model and deep learning. *Nature Biotechnology*, 40(11):1617–1623, October 2022. ISSN 1546-1696. doi:10.1038/s41587-022-01432-w. URL http://dx.doi.org/10.1038/s41587-022-01432-w.

[41] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, Olaf Ronneberger, Lindsay Willmore, Andrew J. Ballard, Joshua Bambrick, Sebastian W. Bodenstein, David A. Evans, Chia-Chun Hung, Michael O'Neill, David Reiman, Kathryn Tunyasuvunakool, Zachary Wu, Akvilė Žemgulytė, Eirini Arvaniti, Charles Beattie, Ottavia Bertolli, Alex Bridgland, Alexey Cherepanov, Miles Congreve, Alexander I. Cowen-Rivers, Andrew Cowie, Michael Figurnov, Fabian B. Fuchs, Hannah Gladman, Rishub Jain, Yousuf A. Khan, Caroline M. R. Low, Kuba Perlin, Anna Potapenko, Pascal Savy, Sukhdeep Singh, Adrian Stecula, Ashok Thillaisundaram, Catherine Tong, Sergei Yakneen, Ellen D. Zhong, Michal Zielinski, Augustin Žídek, Victor Bapst, Pushmeet Kohli, Max Jaderberg, Demis Hassabis, and John M. Jumper. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, May 2024. ISSN 1476-4687. doi:10.1038/s41586-024-07487-w. URL http://dx.doi.org/10.1038/s41586-024-07487-w.

[42] Jacques Boitreaud, Jack Dent, Matthew McPartlon, Joshua Meier, Vinicius Reis, Alex Rogozhnikov, and Kevin Wu. Chai-1: Decoding the molecular interactions of life. October 2024. doi:10.1101/2024.10.10.615955. URL http://dx.doi.org/10.1101/2024.10.10.615955.

[43] Jeremy Wohlwend, Gabriele Corso, Saro Passaro, Noah Getz, Mateo Reveiz, Ken Leidal, Wojtek Swiderski, Liam Atkinson, Tally Portnoi, Itamar Chinn, Jacob Silterra, Tommi Jaakkola, and Regina Barzilay. Boltz-1

democratizing biomolecular interaction modeling. November 2024. doi:10.1101/2024.11.19.624167. URL http://dx.doi.org/10.1101/2024.11.19.624167.

[44] Saro Passaro, Gabriele Corso, Jeremy Wohlwend, Mateo Reveiz, Stephan Thaler, Vignesh Ram Somnath, Noah Getz, Tally Portnoi, Julien Roy, Hannes Stark, David Kwabi-Addo, Dominique Beaini, Tommi Jaakkola, and Regina Barzilay. Boltz-2: Towards accurate and efficient binding affinity prediction. June 2025. doi:10.1101/2025.06.14.659707. URL http://dx.doi.org/10.1101/2025.06.14.659707.

[45] Jeremy Wohlwend, Mateo Reveiz, Axel Feldmann, Wengong Jin, and Regina Barzilay. Minifold: Simple, fast and accurate protein structure prediction. In *International Conference on Learning Representations (ICLR) – 2024*, 2024. URL https://openreview.net/forum?id=SjgfWbamtN. ICLR 2024 submission on OpenReview; no DOI assigned.

[46] R. Car and M. Parrinello. Unified approach for molecular dynamics and density-functional theory. *Physical Review Letters*, 55(22):2471–2474, November 1985. ISSN 0031-9007. doi:10.1103/physrevlett.55.2471. URL http://dx.doi.org/10.1103/PhysRevLett.55.2471.

[47] Tudor-Stefan Cotet, Igor Krawczuk, Filippo Stocco, Noelia Ferruz, Anthony Gitter, Yoichi Kurumida, Lucas de Almeida Machado, Francesco Paesani, Cianna N. Calia, Chance A. Challacombe, Nikhil Haas, Ahmad Qamar, Bruno E. Correia, Martin Pacesa, Lennart Nickel, Kartic Subr, Leonardo V. Castorina, Maxwell J. Campbell, Constance Ferragu, Patrick Kidger, Logan Hallee, Christopher W. Wood, Michael J. Stam, Tadas Kluonis, Süleyman Mert Ünal, Elian Belot, and Alexander Naka. Crowdsourced protein design: Lessons from the adaptyv egfr binder competition. April 2025. doi:10.1101/2025.04.17.648362. URL http://dx.doi.org/10.1101/2025.04.17.648362.

[48] Quentin Fournier, Robert M. Vernon, Almer van der Sloot, Benjamin Schulz, Sarath Chandar, and Christopher James Langmead. Protein language models: Is scaling necessary? September 2024. doi:10.1101/2024.09.23.614603. URL http://dx.doi.org/10.1101/2024.09.23.614603.

[49] Ali Madani, Ben Krause, Eric R. Greene, Subu Subramanian, Benjamin P. Mohr, James M. Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z. Sun, Richard Socher, James S. Fraser, and Nikhil Naik. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8):1099–1106, January 2023. ISSN 1546-1696. doi:10.1038/s41587-022-01618-2. URL http://dx.doi.org/10.1038/s41587-022-01618-2.

[50] Varun Ullanat, Bowen Jing, Samuel Sledzieski, and Bonnie Berger. Learning the language of protein-protein interactions. *bioRxiv*, 2025. doi:10.1101/2025.03.09.642188. URL https://www.biorxiv.org/content/early/2025/03/10/2025.03.09.642188.

[51] Sarah Lewis, Tim Hempel, José Jiménez Luna, Michael Gastegger, Yu Xie, Andrew Y. K. Foong, Victor García Satorras, Osama Abdin, Bastiaan S. Veeling, Iryna Zaporozhets, Yaoyi Chen, Soojung Yang, Arne Schneuing, Jigyasa Nigam, Federico Barbero, Vincent Stimper, Andrew Campbell, Jason Yim, Marten Lienen, Yu Shi, Shuxin Zheng, Hannes Schulz, Usman Munir, Ryota Tomioka, Cecilia Clementi, and Frank Noé. Scalable emulation of protein equilibrium ensembles with generative deep learning. December 2024. doi:10.1101/2024.12.05.626885. URL http://dx.doi.org/10.1101/2024.12.05.626885.

[52] Bowen Jing, Bonnie Berger, and Tommi Jaakkola. Alphafold meets flow matching for generating protein ensembles, 2024. URL https://arxiv.org/abs/2402.04845.

[53] Rishwanth Raghu, Axel Levy, Gordon Wetzstein, and Ellen D. Zhong. Multiscale guidance of alphafold3 with heterogeneous cryo-em data, 2025. URL https://arxiv.org/abs/2506.04490.

[54] Romelia Salomon-Ferrer, David A. Case, and Ross C. Walker. An overview of the amber biomolecular simulation package. *WIREs Computational Molecular Science*, 3(2):198–210, September 2012. ISSN 1759-0884. doi:10.1002/wcms.1121. URL http://dx.doi.org/10.1002/wcms.1121.

[55] Michael R. Shirts and Vijay S. Pande. Comparison of efficiency and bias of free energies computed by exponential averaging, the bennett acceptance ratio, and thermodynamic integration. *The Journal of Chemical Physics*, 122 (14), April 2005. ISSN 1089-7690. doi:10.1063/1.1873592. URL http://dx.doi.org/10.1063/1.1873592.

[56] Cameron F. Abrams. Concurrent dual-resolution monte carlo simulation of liquid methane. *The Journal of Chemical Physics*, 123(23), December 2005. ISSN 1089-7690. doi:10.1063/1.2136884. URL http://dx.doi.org/10.1063/1.2136884.

[57] Artem B. Mamonov, Steven Lettieri, Ying Ding, Jessica L. Sarver, Rohith Palli, Timothy F. Cunningham, Sunil Saxena, and Daniel M. Zuckerman. Tunable, mixed-resolution modeling using library-based monte carlo and graphics processing units. *Journal of Chemical Theory and Computation*, 8(8):2921–2929, July 2012. ISSN 1549-9626. doi:10.1021/ct300263z. URL http://dx.doi.org/10.1021/ct300263z.

[58] Lucien F. Krapp, Luciano A. Abriata, Fabio Cortés Rodriguez, and Matteo Dal Peraro. Pesto: parameter-free geometric deep learning for accurate prediction of protein binding interfaces. *Nature Communications*, 14 (1), April 2023. ISSN 2041-1723. doi:10.1038/s41467-023-37701-8. URL http://dx.doi.org/10.1038/s41467-023-37701-8.

[59] Xinyu Li, Fangjun Huo, Yongbin Zhang, Fangqin Cheng, and Caixia Yin. Enzyme-activated prodrugs and their release mechanisms for the treatment of cancer. *Journal of Materials Chemistry B*, 10(29):5504–5519, 2022. ISSN 2050-7518. doi:10.1039/d2tb00922f. URL http://dx.doi.org/10.1039/D2TB00922F.

[60] Qiuzhen Li, Diandra Daumiller, and Patrick Bryant. Rarefold: Structure prediction and design of proteins with noncanonical amino acids. May 2025. doi:10.1101/2025.05.19.654846. URL http://dx.doi.org/10.1101/2025.05.19.654846.

[61] Jennifer L. Hickey, Dan Sindhikara, Susan L. Zultanski, and Danielle M. Schultz. Beyond 20 in the 21st century: Prospects and challenges of non-canonical amino acids in peptide drug discovery. *ACS Medicinal Chemistry Letters*, 14(5):557–565, April 2023. ISSN 1948-5875. doi:10.1021/acsmedchemlett.3c00037. URL http://dx.doi.org/10.1021/acsmedchemlett.3c00037.

[62] Rachel A. Silverstein, Nahye Kim, Ann-Sophie Kroell, Russell T. Walton, Justin Delano, Rossano M. Butcher, Martin Pacesa, Blaire K. Smith, Kathleen A. Christie, Leillani L. Ha, Ronald J. Meis, Aaron B. Clark, Aviv D. Spinner, Cicera R. Lazzarotto, Yichao Li, Azusa Matsubara, Elizabeth O. Urbina, Gary A. Dahl, Bruno E. Correia, Debora S. Marks, Shengdar Q. Tsai, Luca Pinello, Suk See De Ravin, Qin Liu, and Benjamin P. Kleinstiver. Custom crispr–cas9 pam variants via scalable engineering and machine learning. *Nature*, April 2025. ISSN 1476-4687. doi:10.1038/s41586-025-09021-y. URL http://dx.doi.org/10.1038/s41586-025-09021-y.

[63] Sager J. Gosai, Rodrigo I. Castro, Natalia Fuentes, John C. Butts, Kousuke Mouri, Michael Alasoadura, Susan Kales, Thanh Thanh L. Nguyen, Ramil R. Noche, Arya S. Rao, Mary T. Joy, Pardis C. Sabeti, Steven K. Reilly, and Ryan Tewhey. Machine-guided design of cell-type-targeting cis-regulatory elements. *Nature*, 634(8036): 1211–1220, October 2024. ISSN 1476-4687. doi:10.1038/s41586-024-08070-z. URL http://dx.doi.org/10.1038/s41586-024-08070-z.

[64] Ning Wang, Jiang Bian, Yuchen Li, Xuhong Li, Shahid Mumtaz, Linghe Kong, and Haoyi Xiong. Multi-purpose rna language modelling with motif-aware pretraining and type-guided fine-tuning. *Nature Machine Intelligence*, 6 (5):548–557, May 2024. ISSN 2522-5839. doi:10.1038/s42256-024-00836-4. URL http://dx.doi.org/10.1038/s42256-024-00836-4.

[65] Stefano Angioletti-Uberti, Jakub Lála, and Arnav Cheruku. softnanolab/bagel: v0.1.0 - first public release, July 2025. URL https://doi.org/10.5281/zenodo.15808839.

A Appendix

A.1 Experimental Details

Here, we detail the exact energy terms and their parameters used for all designs presented in the main text. While the Monte Carlo sampling procedure is inherently stochastic, using these settings with BAGEL v0.1 will yield qualitatively similar results across repeated runs. We do not specify the number of tempering cycles (n_{cycles}) used in SimulatedTempering, as the optimal number of cycles depends strongly on the initial random candidate sequence and the specifics of the design task. In practice, n_{cycles} should be tuned to ensure sufficient exploration. An example input file for the simple peptide binder case is also provided.

A.1.1 Parameters - Simple Peptide Binder

Table 1: Design parameters for the CA4 peptide binder. Groups are G_{target} (therapeutic target), G_{hotspot} (binding interface), and G_{binder} (designable peptide). All EnergyTerms use ESMFold as the Oracle.

Protein	**Description**	**Residue Group**	**Value / Definition**
Target	UniProt ID	G_{target}	P22748 (CAH4_HUMAN), all residues
	Hot-spot residues	G_{hotspot}	Glu2–Val9; Val65–Asn78; Ala93–His99; Glu120–His125; Ile145–Val147; Leu206–Val216
Binder	Length	G_{binder}	30 aa (all mutable)
State	**EnergyTerm**	**Affected Group**	**Details**
bound	PTMEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	OverallPLDDTEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	PLDDTEnergy	G_{binder}	$w = 2.0$
	HydrophobicEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$, all atoms considered (surface_only=False)
	PAEEnergy	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 6.0$, interface PAE
	SeparationEnergy	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 1.0$
Minimizer	**Parameter**	**Value**	
SimulatedTempering	MutationProtocol	Canonical	
	n_{mut}	1	
	T_{low}	0.1	
	T_{high}	1.0	
	$n_{\text{low, steps}}$	400	
	$n_{\text{high, steps}}$	100	
	$n_{\text{best system}}$	500	

Table 2: Design parameters for the EGFR peptide binder. Groups are G_{target} (therapeutic target), G_{hotspot} (binding interface), and G_{binder} (designable peptide). All EnergyTerms use ESMFold as the Oracle.

Protein	**Description**	**Residue Group**	**Value / Definition**
Target	UniProt ID	G_{target}	P00533 (EGFR), only residues Cys305 to Val482
	Hot-spot residues	G_{hotspot}	Gly354–Thr358; Gly435–Val437; Ile439–Gly441
Binder	Length	G_{binder}	30 aa (all mutable)
State	**EnergyTerm**	**Affected Group**	**Details**
bound	PTMEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	PLDDTEnergy	G_{binder}	$w = 2.0$
	HydrophobicEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$, all atoms considered (surface_only=False)
	PAEEnergy	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 4.0$, interface PAE
	SeparationEnergy	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 0.2$
Minimizer	**Parameter**	**Value**	
SimulatedTempering	MutationProtocol	Canonical	
	n_{mut}	1	
	T_{low}	0.001	
	T_{high}	1.0	
	$n_{\text{low, steps}}$	400	
	$n_{\text{high, steps}}$	100	
	$n_{\text{best system}}$	500	

Table 3: Design parameters for the DERF7 peptide binder. Groups are G_{target} (therapeutic target), G_{hotspot} (binding interface), and G_{binder} (designable peptide). All EnergyTerms use ESMFold as the Oracle.

Protein	**Description**	**Residue Group**	**Value / Definition**
Target	UniProt ID	G_{target}	Q26456 (ALL7_DERFA)
Binder	Length	G_{binder}	30 aa (all mutable)
State	**EnergyTerm**	**Affected Group**	**Details**
bound	PTMEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	OverallPLDDTEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	PLDDTEnergy	G_{binder}	$w = 2.0$, confident binder
	HydrophobicEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$, all atoms considered (surface_only=False)
	PAEEnergy	$G_{\text{target}} \leftrightarrow G_{\text{binder}}$	$w = 6.0$, interface PAE
	SeparationEnergy	$G_{\text{target}} \leftrightarrow G_{\text{binder}}$	$w = 1.0$
Minimizer	**Parameter**	**Value**	
SimulatedTempering	Mutator	Canonical	
	n_{mut}	1	
	T_{low}	0.1	
	T_{high}	1.0	
	$n_{\text{low, steps}}$	400	
	$n_{\text{high, steps}}$	100	
	$n_{\text{best system}}$	500	

A.1.2 Parameters - Target Intrinsically Disordered Epitopes

Table 4: Design parameters for the ASYN peptide binder. Groups are G_{target} (therapeutic target), G_{hotspot} (targeted disordered epitope), and G_{binder} (designable peptide). All `EnergyTerms` use `ESMFold` as the `Oracle`.

Protein	**Description**	**Residue Group**	**Value / Definition**
Target	UniProt ID	G_{target}	P37840 (SYUA_HUMAN)
	Hot-spot residues	G_{hotspot}	Leu100–Ala140
Binder	Length	G_{binder}	50 aa (all mutable)
`State`	**`EnergyTerm`**	**Affected Group**	**Details**
bound	PTMEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	OverallPLDDTEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	PLDDTEnergy	G_{binder}	$w = 4.0$, confident binder
	PLDDTEnergy	G_{hotspot}	$w = 4.0$, induce order on epitope
	HydrophobicEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 2.0$, all atoms considered (`surface_only=False`)
	PAEEnergy	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 4.0$, interface PAE
	SeparationEnergy	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 0.2$
`Minimizer`	**Parameter**	**Value**	
`SimulatedTempering`	`Mutator`	`Canonical`	
	n_{mut}	1	
	T_{low}	0.2	
	T_{high}	2.0	
	$n_{\text{low, steps}}$	50	
	$n_{\text{high, steps}}$	50	
	$n_{\text{best system}}$	100	

Table 5: Design parameters for the CD28 peptide binder. Groups are G_{target} (therapeutic target), G_{hotspot} (targeted disordered epitope), and G_{binder} (designable peptide). All `EnergyTerms` use `ESMFold` as the `Oracle`.

Protein	**Description**	**Residue Group**	**Value / Definition**
Target	UniProt ID	G_{target}	P10747 (CD28_HUMAN)
	Hot-spot residues	G_{hotspot}	Gly121–Arg140
Binder	Length	G_{binder}	30 aa (all mutable)
`State`	**`EnergyTerm`**	**Affected Group**	**Details**
bound	PTMEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	OverallPLDDTEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	PLDDTEnergy	G_{binder}	$w = 4.0$, confident binder
	PLDDTEnergy	G_{hotspot}	$w = 4.0$, induce order on epitope
	HydrophobicEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 2.0$, all atoms considered (`surface_only=False`)
	PAEEnergy	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 4.0$
	SeparationEnergy	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 0.2$
	SecondaryStructureEnergy	G_{hotspot}	$w = 8.0$, alpha-helix
`Minimizer`	**Parameter**	**Value**	
`SimulatedTempering`	`Mutator`	`Canonical`	
	n_{mut}	1	
	T_{low}	0.3	
	T_{high}	2.0	
	$n_{\text{low, steps}}$	100	
	$n_{\text{high, steps}}$	100	
	$n_{\text{best system}}$	200	

Table 6: Design parameters for the P53 peptide binder. Groups are G_{target} (therapeutic target), G_{hotspot} (targeted disordered epitope), and G_{binder} (designable peptide). All `EnergyTerms` use `ESMFold` as the `Oracle`.

Protein	**Description**	**Residue Group**	**Value / Definition**
Target	UniProt ID	G_{target}	P04637 (P53_HUMAN)
	Hot-spot residues	G_{hotspot}	Pro1-Phe18
Binder	Length	G_{binder}	30 aa (all mutable)
`State`	**`EnergyTerm`**	**Affected Group**	**Details**
bound	PTMEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	OverallPLDDTEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	PLDDTEnergy	G_{binder}	$w = 4.0$, confident binder
	PLDDTEnergy	G_{hotspot}	$w = 4.0$, induce order on epitope
	HydrophobicEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 2.0$, all atoms considered (`surface_only=False`)
	PAEEnergy	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 5.0$, interface PAE
	SeparationEnergy	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 0.2$
`Minimizer`	**Parameter**	**Value**	
`SimulatedTempering`	`Mutator`	`Canonical`	
	n_{mut}	1	
	T_{low}	0.1	
	T_{high}	1.0	
	$n_{\text{low, steps}}$	200	
	$n_{\text{high, steps}}$	200	
	$n_{\text{best system}}$	400	

Table 7: Design parameters for the SUMO1 peptide binder. Groups are G_{target} (therapeutic target), G_{hotspot} (targeted disordered epitope), and G_{binder} (designable peptide). All `EnergyTerms` use `ESMFold` as the `Oracle`.

Protein	**Description**	**Residue Group**	**Value / Definition**
Target	UniProt ID	G_{target}	P63165 (SUMO1_HUMAN)
	Hot-spot residues	G_{hotspot}	Gly1–Gly21
Binder	Length	G_{binder}	30 aa (all mutable)
`State`	**`EnergyTerm`**	**Affected Group**	**Details**
bound	PTMEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	OverallPLDDTEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	PLDDTEnergy	G_{binder}	$w = 4.0$, confident binder
	PLDDTEnergy	G_{hotspot}	$w = 4.0$, induce order on epitope
	HydrophobicEnergy	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 2.0$, all atoms considered (`surface_only=False`)
	PAEEnergy	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 5.0$, interface PAE
	SeparationEnergy	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 0.2$
`Minimizer`	**Parameter**	**Value**	
`SimulatedTempering`	`Mutator`	`Canonical`	
	n_{mut}	1	
	T_{low}	0.1	
	T_{high}	1.0	
	$n_{\text{low, steps}}$	200	
	$n_{\text{high, steps}}$	200	
	$n_{\text{best system}}$	400	

A.1.3 Parameters - Multi-State Selective Peptide Binder

Table 8: Design parameters for the selective zinc finger binder. Groups are G_{target} (target protein), $G_{\text{non-target}}$ (off-target), G_{hotspot} (target binding interface on target protein), and G_{binder} (designable peptide). All `EnergyTerms` use `ESMFold` as the `Oracle`.

Protein	**Description**	**Residue Group**	**Value / Definition**
Target	UniProt ID	G_{target}	P08046 (*Mus musculus*, i.e., mouse variant)
	Hot-spot residues	G_{hotspot}	His81–Asp90, i.e., last 10 residues
Non-target	UniProt ID	$G_{\text{non-target}}$	O00488 (*Homo sapiens*, i.e., human variant)
Binder	Length	G_{binder}	15 aa (all mutable)
`State`	**`EnergyTerm`**	**Affected Group**	**Details**
target (binding)	`PTMEnergy`	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	`PLDDTEnergy`	$G_{\text{target}} \cup G_{\text{binder}}$	$w = 1.0$
	`PAEEnergy`	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 5.0$
	`SeparationEnergy`	$G_{\text{hotspot}} \leftrightarrow G_{\text{binder}}$	$w = 1.0$
non-target (avoiding)	`PTMEnergy`	$G_{\text{non-target}} \cup G_{\text{binder}}$	$w = 1.0$, remain confident in prediction
	`PLDDTEnergy`	$G_{\text{non-target}} \cup G_{\text{binder}}$	$w = 1.0$, remain confident in prediction
	`PAEEnergy`	$G_{\text{non-target}} \leftrightarrow G_{\text{binder}}$	$w = -5.0$, discourage binding
	`SeparationEnergy`	$G_{\text{non-target}} \leftrightarrow G_{\text{binder}}$	$w = -1.0$, discourage binding
`Minimizer`	**Parameter**	**Value**	
`SimulatedTempering`	`Mutator`	`Canonical`	
	n_{mut}	1	
	T_{low}	0.1	
	T_{high}	1.0	
	$n_{\text{low, steps}}$	200	
	$n_{\text{high, steps}}$	50	
	$n_{\text{best system}}$	250	

A.1.4 Parameters - Enzyme Variants with Conserved Active Site

Table 9: Design parameters for the oxidoreductase variant generation. Groups are G_{all} (full enzyme), $G_{\text{conserved}}$ (immutable/conserved residues). The `EmbeddingsSimilarityEnergy` uses the 650M parameter version of `ESM2` as the `Oracle`.

Protein	**Description**	**Residue Group**	**Value / Definition**
Target	UniProt ID	G_{all}	P0AEG4 (wild-type oxidoreductase)
	Active Site	$G_{\text{conserved}}$	Cys30, Cys33 (immutable)
`State`	**`EnergyTerm`**	**Affected Group**	**Details**
variant	`EmbeddingsSimilarityEnergy`	$G_{\text{conserved}}$	$w = 1.0$, reference: ESM2 embeddings of Cys30 and Cys33 using the original, wild-type sequence
`Minimizer`	**Parameter**	**Value**	
`MonteCarloMinimizer`	`Mutator`	`Canonical`	
	n_{mut}	1	
	Temperature	10^{-4}	
	n_{steps}	10,000	

A.1.5 Example Input File

The following Python script gives an example of peptide binder design against CA4 using BAGEL.

```python
import os
import random
import pathlib as pl
import bagel as bg

def main(
    use_modal: bool = False,
    binder_sequence: str = None,
    optimization_params: dict = None,
    output_dir: str = 'data/CA4-binder'
):
    # PART 1: Define the target protein
    # Carbonic anhydrase 4, CAH4_HUMAN, Gene: CA4
    # PDB ID: 1ZNC; chain A; UniProt ID: P22748
    target_sequence = (
        "AESHWCYEVQAESSNYPCLVPVKWGGNCQKDRQSPINIVTTKAKVDKKLGRFFFSGYDKKQTWTVQ"
        "NNGHSVMMLLENKASISGGGLPAPYQAKQLHLHWSDLPYKGSEHSLDGEHFAMEMHIVHEKEKGTS"
        "RNVKEAQDPEDEIAVLAFLVEAGTQVNEGFQPLVEALSNIPKPEMSTTMAESSLLDLLPKEEKLRH"
        "YFRYLGSLTTPTCDEKVVWTVFREPIQLHREQILAFSQKLYYDKEQTVSMKDNVRPLQQLGQRTVIKS"
    )

    # Define the mutability of the residues, all immutable in this case since this is the target sequence
    mutability = [False for _ in range(len(target_sequence))]

    # Define the chain
    residues_target = [
        bg.Residue(name=aa, chain_ID='CA4H', index=i, mutable=mut)
        for i, (aa, mut) in enumerate(zip(target_sequence, mutability))
    ]

    # Now define residues in the hotspot where you want to bind.
    residue_ids = [
        [1, 2, 3, 4, 5, 6, 7, 8],
        [64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77],
        [92, 93, 94, 95, 96, 97, 98],
        [119, 120, 121, 122, 123, 124],
        [144, 145, 146],
        [205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215],
        ]
    residue_ids = [item for sublist in residue_ids for item in sublist]

    residues_hotspot = [residues_target[i] for i in residue_ids]
    target_chain = bg.Chain(residues=residues_target)

    # PART 2: Define the binder
    binder_length = 30
    if binder_sequence is None:
        # Start with a random sequence of amino acids selecting randomly from the 30 amino acids
        binder_sequence = ''.join([random.choice(list(bg.constants.aa_dict.keys())) for _ in range(
            binder_length)])
    else:
        # or restart from a previously designed sequence
        assert len(binder_sequence) == binder_length, 'Binder sequence must be of length 30'

    # Define the mutability of the residues, all mutable in this case since this is the design sequence
    mutability = [True for _ in range(len(binder_sequence))]
    # Define the chain
    residues_binder = [
        bg.Residue(name=aa, chain_ID='BIND', index=i, mutable=mut)
        for i, (aa, mut) in enumerate(zip(binder_sequence, mutability))
    ]
    binder_chain = bg.Chain(residues=residues_binder)

    # PART 3: Define the Oracles and EnergyTerms

    # Define the ESMFold Oracle
    config = {
        'output_pdb': False,
        'output_cif': False,
        'glycine_linker': 50 * "G",
        'position_ids_skip': 512,
    }

    esmfold = bg.oracles.ESMFold(
        use_modal=use_modal, config=config
    )
```

```python

    # Define the energy terms to be applied to the chain
    energy_terms = [
        bg.energies.PTMEnergy(
            oracle=esmfold,
            weight=1.0,
        ),
        bg.energies.OverallPLDDTEnergy(
            oracle=esmfold,
            weight=1.0,
        ),
        bg.energies.PLDDTEnergy(
            oracle=esmfold,
            residues=residues_binder,
            weight=2.0
        ),
        bg.energies.HydrophobicEnergy(
            oracle=esmfold,
            weight=1.0
            ),
        bg.energies.PAEEnergy(
            oracle=esmfold,
            residues=[residues_hotspot, residues_binder],
            weight=6.0,
        ),
        bg.energies.SeparationEnergy(
            oracle=esmfold,
            residues=[residues_hotspot, residues_binder],
            weight=1.0,
        ),
    ]

    # PART 4: Define the State
    state = bg.State(
        chains=[binder_chain, target_chain],
        energy_terms=energy_terms,
        name='state',
    )

    # PART 5: Define the System
    initial_system = bg.System(
        states=[state],
        name='CA4-binder'
    )

    # PART 6: Define the minimizer and run the optimization
    mutator = bg.mutation.Canonical()

    # Use optimization parameters if provided, otherwise use defaults
    if optimization_params is None:
        optimization_params = {
            'high_temperature': 1.0,
            'low_temperature': 0.1,
            'n_steps_high': 100,
            'n_steps_low': 400,
            'n_cycles': 100,
        }

    current_dir = os.path.dirname(os.path.abspath(__file__))
    minimizer = bg.minimizer.SimulatedTempering(
        mutator=mutator,
        high_temperature=optimization_params['high_temperature'],
        low_temperature=optimization_params['low_temperature'],
        n_steps_high=optimization_params['n_steps_high'],
        n_steps_low=optimization_params['n_steps_low'],
        n_cycles=optimization_params['n_cycles'],
        preserve_best_system_every_n_steps=optimization_params['n_steps_high'] + optimization_params['
            n_steps_low'],
        log_frequency=1,
        log_path=pl.Path(os.path.join(current_dir, output_dir)),
    )

    # Return the best system
    best_system = minimizer.minimize_system(system=initial_system)
    return best_system
```