Antibody Library Design by Seeding Linear Programming with Inverse Folding and Protein Language Models

Conor F. Hayes, Steven A. Magana-Zook, Andre Gonçalves, Ahmet Can Solak, Daniel Faissol, Mikel Landajuela*
Lawrence Livermore National Laboratory
Livermore, CA, 94550, USA

Abstract

We propose a novel approach for antibody library design that combines deep learning and multi-objective linear programming with diversity constraints. Our method leverages recent advances in sequence and structure-based deep learning for protein engineering to predict the effects of mutations on antibody properties. These predictions are then used to seed a cascade of constrained integer linear programming problems, the solutions of which yield a diverse and high-performing antibody library. Operating in a *cold-start* setting, our approach creates designs without iterative feedback from wet laboratory experiments or computational simulations. We demonstrate the effectiveness of our method by designing antibody libraries for Trastuzumab in complex with the HER2 receptor, showing that it outperforms existing techniques in overall quality and diversity of the generated libraries. Our source code is available at https://github.com/LLNL/protlib-designer.

1 Introduction

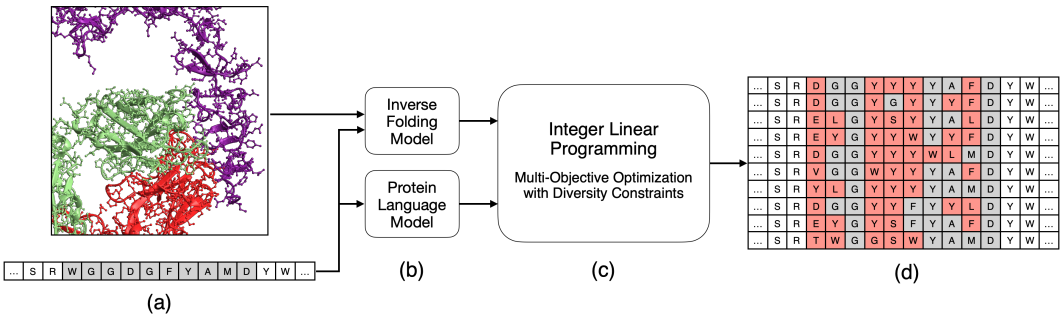


Figure 1: Overview of the proposed method for antibody library design. (a) The input to the method is an antibody-antigen complex and a target antibody sequence. (b) We generate *in silico* deep mutational scanning data using protein language and inverse folding models. (c) The result is fed into a multi-objective linear programming solver. (d) The solver generates a library of antibodies that are co-optimized for the *in silico* scores while satisfying diversity constraints.

Antibody-based therapeutics have revolutionized the treatment of a wide range of diseases, including cancer, autoimmune disorders, and infectious diseases [1]. To develop these drugs, researchers

*Corresponding author: landajuelala1@llnl.gov

Preprint. Under review.

often rely on directed evolution, a process that involves experimentally screening large libraries of antibodies to identify candidates with desirable properties [2]. In the early stages of directed evolution for antibody drug discovery, a key challenge is to design a diverse library of potential antibodies for further experimental screening [3, 4]. This challenge involves identifying promising "leads" that exhibit high affinity to target antigens and possess favorable developability characteristics [5, 6]. Effective library design is also central to many Bayesian optimization methods used in antibody discovery [3, 7], a problem known as batch active learning [8].

Traditionally, the initial step to bootstrap the directed evolution process has relied on random mutagenesis, often informed by deep mutational scanning data [9, 10, 11, 12], or through biomolecular simulations of binding free energies [13, 7]. These methods have been successfully applied to generate libraries that are enriched with high-affinity antibodies [14, 11]. However, these methods required a large amount of experimental data and/or computational resources, which can be prohibitively costly and time-consuming. Additionally, these approaches often overlook the need to maintain diversity within the antibody library, which is crucial for exploring the vast sequence space and overcoming potential failure modes or systematic biases in prediction tools [15, 16, 17].

Recent advancements in deep learning applied to biological sequences [18, 19, 20] and structures [21, 22], or a combination of both [23], have shown great promise as *in silico* screening tools for antibody drug discovery. These methods leverage the power of machine learning to learn from evolutionary scale data and predict the effects of mutations on antibody properties, such as binding affinity, stability, and developability [24, 25].

In this paper, we propose a novel approach that combines recent advances in deep learning for protein engineering with integer linear programming (ILP) to design diverse and high-quality antibody libraries. We are interested in a *cold-start* setting, where the objective is to design effective starting libraries without the need for experimental or computational fitness data. This setting is relevant for rapid response design scenarios against escape variants or new targets [7], where the availability of experimental data is limited or non-existent, and for seeding the directed evolution process with diverse and high-quality candidates [3]. We summarize our contributions as follows:

- We introduce a novel method for antibody library design that leverages constrained integer linear programming to generate high-quality libraries with explicit control over diversity parameters.
- We apply the method to the problem of *cold-start* antibody library design using *in silico* deep mutational scanning data from inverse folding and protein language models.
- We evaluate the performance of our approach to design antibody libraries for the Trastuzumab antibody in complex with the HER2 receptor.

2 Related work

Below we discuss relevant work related to antibody library design.

Antibody library design Typically when designing antibody libraries, experimental deep mutational scanning data or simulation data is utilized. For example, in [12] the authors start with experimental single-site deep mutational scanning to create a combinatorial library of antibody with respect to the Trastuzumab antibody. The library containing over $31,000$ designs is validated via *in vitro* experiments to determine binding quality. Additionally, in [7], an antibody that had previously lost potency due to mutations on the SARS-COV-2 antigen, is computationally redesigned to restore potency by producing an antibody library for experimental validation using a sequence generator informed by simulated data obtained after a large scale simulation campaign. In contrast, our proposed approach utilizes relatively inexpensive machine learning models to generate libraries in a *cold-start* setting without the need for experimental or computational fitness data.

Antibody library design with diversity Recently, a number of methods have been proposed to explicitly include diversity when generating antibody libraries. In [26], the authors use a quality-diversity optimization approach called MAP-elites [27] to produce a library of high performing and diverse antibodies. Similarly to SPEA2, the approach in [26], struggles to generate libraries of a pre-defined size and requires an additional down-selection step to generate a final library.

In [28], a constrained Bayesian optimization approach is proposed to strike a balance between binding affinity and thermostability while maintaining sequence diversity. This method optimizes a latent

space representation learned by a variational autoencoder [29] to produce a fit and diverse batch of designs. To ensure diversity, "black-box" constraints are utilized with an additional Levenstein distance constraint. In contrast, our ILP formulation optimizes directly on the additive objective values produced from the ML driven models, and enforces diversity directly in sequence-space with respect to the defined constraints.

In the larger context of protein design, [16, 17] recently proposed a differentiable generative approach to jointly optimize for the expected score and diversity of the generated library. By relaxing the discrete optimization problem to a continuous one, the authors are able to utilize gradient-based optimization methods to generate libraries. However, the diversity is represented by the entropy of the generator, which may not be directly related to the desired sequence diversity.

Integer linear programming for antibody optimization Antibody optimization has been formulated as integer linear programming in [14] and is used in an optimization pipeline to design a library with broadly binding and stable antibodies with respect to 180 divergent HIV viral strains. While the proposed approach is successful, the optimization pipeline requires extensive simulation data to train an ML binding predictor. The proposed ILP is formulated to optimize the respective binding scores of the trained ML predictor. We note the ML predictors may struggle to generalize to targets outside of the problem under consideration. Therefore, to apply this formulation to a new target viral strain, new data must be generated and the ML model retrained. Additionally, the ILP defined in [14] does not include diversity as a constraint during optimization.

Machine learning for antibody engineering Masked language models, trained on protein sequences, have been used to predict mutations in [20] and [19], where the scores produced by the models convey the predictive effects of mutations on protein function. For example, in [20], the authors use the likelihood of a mutation of a wild-type antibody under an ensemble of masked language models to select mutations on a number of antibodies. The models select evolutionarily plausible mutations without any context about the target antigen.

The majority of research to date has focused on using scores generated from masked language models without any context about the target antigen. However, the approach proposed in this work utilizes scores from antibody inverse folding models that can interpret provided antigen information. In [22], the trained inverse folding model achieves strong correlations when predicting antibody-antigen binding affinity. As machine learning methods improve, we assume the ability of these models to convey the predictive effects of mutations on protein function will also improve.

3 Method

The antibody reengineering problem starts with a wild-type antibody sequence $\mathbf{w} = (w_1, \ldots, w_L)$ of length L, where each w_i takes a value in the set of $M = 20$ amino acids $A = \{a_1, \ldots a_M\}$. The wild-type antibody might present weak binding to an epitope $\mathbf{g}$ (showed in purple in Figure 1(a)). The antibody interface is a set of $N \leq L$ position indices $\mathbf{r} = \{r_1, \ldots, r_N\}$ where $1 \leq r_i \leq L$ (gray positions in Figure 1(a)). Typically, residues at positions $\mathbf{r}$ are in contact with the antigen $\mathbf{g}$ or have been deemed important for binding. In the following, we use the notation $\mathcal{I}_A(a)$ to denote the index of the amino acid a in the set A.

In a library design problem, the objective is to identify a set of mutants derived from the wild-type antibody $\mathbf{w}$ that exhibit enhanced binding to the antigen $\mathbf{g}$ while preserving developability properties [30, 31, 11]. Each mutant is represented by $\mathbf{x} = (x_1, \ldots, x_N)$, where $x_i \in A$. Additionally, the set of mutants should be diverse to encompass a broad spectrum of potential antibodies and reduce the risk of experimental failure [32]. Consider a batch of K mutants $B = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$. Formally, we can write the problem as the following constrained multi-objective optimization problem:

$$\text{Minimize}_B \quad \mathbf{F}(B), \tag{1}$$

$$\text{subject to} \quad \text{div}(B) \geq \delta, \tag{2}$$

where $\mathbf{F}(B)$ is a vector of aggregated scores for the batch B, $\text{div}(B)$ is the diversity of the batch B, and δ is a given tolerance. Different scoring functions $\mathbf{F}(B)$ with different fidelity levels can be used to evaluate the mutants in the batch B. For instance, $\mathbf{F}(B)$ could involve experimental data, computational models, or a combination of both. In this work, we focus on scoring functions based on deep learning models.

3.1 *In silico* deep mutational scanning with deep learning

Experimental deep mutational scanning (DMS) is a powerful technique to guide protein design by systematically mutating position indices $\mathbf{r}$ to all amino acid identities in the set A and measuring the effects of these mutations on protein function [9, 10, 11]. The result is a matrix of scores s_{ij}, where s_{ij} is the effect of mutating the amino acid at position i to amino acid j. DMS has been successfully used to design libraries enriched with high-affinity antibodies [14, 11]. However, DMS can be prohibitively costly and time-consuming, and it may not be feasible for all proteins of interest. For that reason, we propose to use *in silico* DMS, obtained via deep learning methods, to predict the effects of mutations on protein properties. We use two recent types of deep learning models to predict the effects of mutations: sequence-based models and structure-based models.

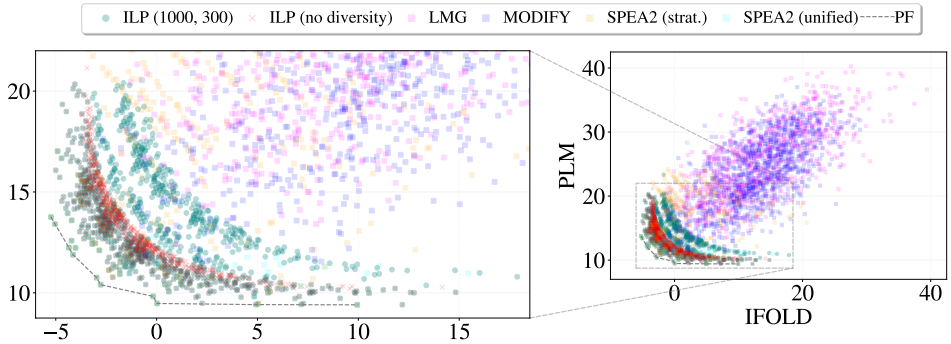


Figure 2: (right) Objective values of mutants generated by the ILP and baseline methods for Trastuzumab in complex with the HER2 receptor. Each point represents a 5 to 8-point mutant. (left) Zoomed-in perspective focusing on the objective values of the Pareto front and mutants generated by the ILP with, and without, diversity constraints.

***Intrinsic fitness* score from protein language models** Protein language models (PLM) [18, 19, 20], trained on evolutionary-scale datasets of protein sequences, can be used to capture the evolutionary rules that govern protein sequences. PLMs capture the *intrinsic fitness* of a protein sequence, which is related to its folding stability, thermostability, developability, and evolutionary plausibility [20]. In this work, we use a protein masked language model [20] and compute the score as

$$s_{ij}^{\text{PLM}} = -\log\left(\frac{p(x_i = a_j|\mathbf{w})}{p(x_i = w_i|\mathbf{w})}\right) = -\log(p(x_i = a_j|\mathbf{w})) + \log(p(x_i = w_i|\mathbf{w})). \tag{3}$$

The score function (3) has been shown to be predictive of mutation effects [18, 19, 20]. In [19] they refer to (3) as the *wild-type marginal probability*, and they show extensive experiments on the predictive power of this score. Note that (3) is a measure of the likelihood of observing a mutation to amino acid a_j at position i over the wild type sequence $\mathbf{w}$. As such, it is formally related to the free energy difference between the wild type and the mutant state, $\Delta G_{w_i \to a_j} = -\kappa T \log\left({}^{p(a_j)}/_{p(w_i)}\right)$, by the Boltzmann distribution in statistical mechanics, where κ is the Boltzmann constant and T is the temperature.

***Extrinsic fitness* score from inverse folding models** Several inverse-folding (IFOLD) methods have been proposed to design protein sequences that fold into a given target structure [33, 21, 34]. It has been shown that, given an epitope as context, these methods can capture the *extrinsic fitness* of an antibody sequence, e.g., the specific selection pressure for binding to the epitope [33]. These models are conditioned on structural data of the co-complex structure $\text{struct}(\mathbf{w}, \mathbf{g})$. We define the score

$$s_{ij}^{\text{IFOLD}} = -\log(p(x_i = a_j|\mathbf{w}_{<i}, \text{struct}(\mathbf{w}, \mathbf{g}))) + \log(p(x_i = w_i|\mathbf{w}_{<i}, \text{struct}(\mathbf{w}, \mathbf{g}))). \tag{4}$$

In this work, we consider Antifold [22], an IFOLD method based on ESM-IF1 [33], fine-tuned on a large dataset of antibody data [35]. In the case of Antifold, $\text{struct}(\mathbf{w}, \mathbf{g})$ is given by the spatial coordinates of the backbone atoms (N, C_α and C). Geometric deep learning methods [24, 25] trained as regressors of antibody-antigen binding affinity could also be used to capture extrinsic fitness.

3.2 Multi-objective integer linear programming for antibody optimization

In this section, we consider the problem of finding a mutant of the wild-type antibody that minimizes a set Q of objectives. A mutant $\mathbf{x} = (x_1, \ldots, x_N)$ is represented by a matrix $\mathbf{z} = z_{ij} \in \{0, 1\}, \forall 1 \leq i \leq N, 1 \leq j \leq M$, where $z_{ij} = 1$ if the amino acid j is present in the position i of the mutant, and $z_{ij} = 0$ otherwise. We assume that for each objective $q \in Q$, we have a score $s^q_{ij} \in \mathbb{R}$ associated to the contribution of the amino acid j in position i to the objective q.

We define the ILP multi-objective problem as follows:

$$\text{Minimize} \quad \left\{ \sum_{i=1}^{N} \sum_{j=1}^{M} s^q_{ij} z_{ij} \right\}_{q \in Q} \tag{5}$$

$$\text{subject to} \quad 0 \leq \sum_{j=1}^{M} z_{ij} \leq 1, \quad \forall i \in \{1, \ldots, N\} \tag{6}$$

$$n_{\text{min}} \leq \sum_{i=1}^{N} \sum_{j=1}^{M} z_{ij} \leq n_{\text{max}} \tag{7}$$

$$\sum_{i=1}^{N} z_{i\mathcal{I}_A(w_{r_i})} = 0 \tag{8}$$

$$z_{ij} \in \{0, 1\}, \quad \forall i \in \{1, \ldots, N\}, j \in \{1, \ldots, M\} \tag{9}$$

where (6) constraints the solution to one mutation per position, (7) constraints the number of mutations to be between n_{min} and n_{max}, (8) constraints the current solution to be different from the wild-type at the positions $\mathbf{r}$, and (9) constraints the solution to be binary.

If $|Q| = 1$, problem (5)–(9) can be solved by presenting it as an integer linear program. Specifically, problem (5)–(9) is a relaxed version of an assignment problem [36], where the bijectivity constraint is removed from (6). The convex optimization problem can be solved globally and efficiently using any available ILP solver. In this work, we use the COIN-OR Branch and Cut solver (CBC) [37]. If $|Q| > 1$, the problem is a multi-objective optimization problem that can be solved in a Pareto optimal sense using the weighted sum method [38], i.e., a single objective problem is solved by weighting the vector (5) with weights $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{|Q|})$, where $\lambda_q \geq 0$ and $\sum_{q=1}^{|Q|} \lambda_q = 1$. In the following, we consider $Q = \{\text{PLM}, \text{IFOLD}\}$ with the corresponding score matrices introduced in (3) and (4).

Note that the additive model in (5) has been extensively used in the literature to score multipoint mutants. In [39], the authors show that (5) correlates well with experimentally measured binding affinities when used with different generative models, including PLM and IFOLD methods. For $q = \text{PLM}$, the additive model in (5) has been used in [19] for scoring general protein mutants.

3.3 *Solve-and-remove* algorithm for library design with diversity constraints

In this section, we present a novel algorithm to solve the antibody library design problem with diversity constraints (1)–(2). We use the notation $\text{ILP}(n_{\text{min}}, n_{\text{max}}, \mathbf{z}, Q, \boldsymbol{\lambda})$ to represent the problem defined in equations (5)–(9) with minimum and maximum number of mutations per mutant given by n_{min} and n_{max}, respectively. Given a budget of $K > 0$ mutants, we aim to find the top K solutions to the problem, ensuring diversity among the selected mutants.

The proposed *solve-and-remove* algorithm is presented in Algorithm 1. Additionally, we visualize the *solve-and-remove* process using a step by step illustration in Figure 3. The basic idea is to solve a sequence of K problems (10), where each problem is supplemented with constraints that depend on the solutions found in previous steps (11)–(13). The *ball constraint* (11) ensures that a set of balls of radius ϵ around the solutions found in previous steps are removed from the feasible region of the next step. The *position constraint* (12) and the *mutation constraint* (13) ensure that the final antibody library presents at most δ_1 mutations in the same position and at most δ_2 mutants with the same mutation, respectively.

Furthermore, in the experiments section, we refer to this algorithm as simply the ILP algorithm.

Algorithm 1 *Solve-and-remove* algorithm for antibody library design with diversity constraints

Input: $\mathbf{w}$ (wild-type sequence), $K \in \mathbb{N}$ (budget), $\epsilon \in \mathbb{R}$ (ball radius), $\delta_1 \in \mathbb{N}$ (max number of mutations in the same position), $\delta_2 \in \mathbb{N}$ (max number of mutants with same mutation), and, $\text{ILP}(n_{\text{min}}, n_{\text{max}}, \mathbf{z})$ (integer linear programming solver for 5-9).

Output: $\{\mathbf{z}^{(1)}, \dots \mathbf{z}^{(K)}\}$ (library of mutants)

Initialize: Compute s^{PLM} and s^{IFOLD} score matrices, define $z_{ij}^{(0)} = 0, \forall i, j$.

for $k = 1$ to K **do**

Sample $\boldsymbol{\lambda}^{(k)} \sim \text{Dirichlet}(1, \dots, \alpha_{|Q|+1})$ and normalize $\boldsymbol{\lambda}^{(k)} = \boldsymbol{\lambda}^{(k)} / \sum_{q=1}^{|Q|} \lambda_q^{(k)}$.

Solve problem:

$$\begin{cases} \text{ILP}(n_{\text{min}}, n_{\text{max}}, \mathbf{z}^{(k)}, \{\text{PLM}, \text{IFOLD}\}, \boldsymbol{\lambda}^{(k)}) & (10) \\ \sum_{i=1}^{N}\sum_{j=1}^{M} z_{ij}^{(k)} - 2\sum_{i=1}^{N}\sum_{j=1}^{M} z_{ij}^{(l)} \geq 1 + \epsilon - \sum_{i=1}^{N}\sum_{j=1}^{M} z_{ij}^{(l)}, \quad \forall l = 1, \dots, k-1, & (11) \\ \sum_{j=1}^{M} z_{ij}^{(k)} = 0, \quad \forall i \text{ s.t. } \sum_{l=1}^{k-1}\sum_{j=1}^{M} z_{ij}^{(l)} \geq \delta_1, & (12) \\ z_{ij}^{(k)} = 0, \quad \forall i, j \text{ s.t. } \sum_{l=1}^{k-1} z_{ij}^{(l)} \geq \delta_2 & (13) \end{cases}$$

end for

4 Experiments

We show the flexibility of our library design method using the Trastuzumab antibody in complex with the antigen human epidermal growth factor receptor 2 (HER2) [12]. Due to space limitations, we refer the reader to Appendix A.4 for extensive ablations and discussion.

Experiment configuration To evaluate the different batches of mutated sequences generated by the ILP, we use a binding prediction surrogate to evaluate each sequence (see Appendix A.4). We also compare the performance of the ILP against a LMG algorithm designed for antibody library design [7].

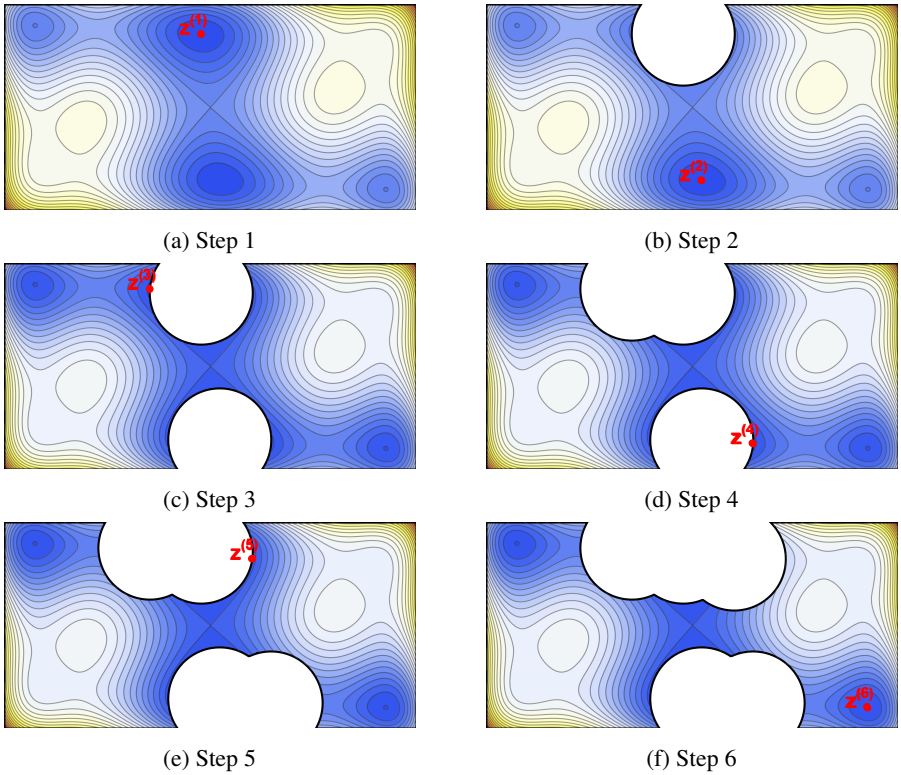
We mutate the Trastuzumab antibody sequence on the CDR3 region of the heavy chain. The set of mutable positions N contains the following positions: H99, H100, H101, H102, H103, H104, H105, H106, H107, H108. The set of amino acids M contains all amino acids except for wild-type, resulting in a set of 19 possible amino acids from which to choose.

We generate a batch, K, of $1,000$ mutated sequences from wild-type, the minimum number of mutations, n_{min}, is 5 and the maximum of number of mutations, n_{max}, is 8.

Experimental setup As previously outlined, the proposed method can be used to design a diverse antibody library whereby a given wild-type is mutated with respect to pre-defined optimization objective(s). For our experiments, we use scores s_{ij}^{IFOLD} from Antifold [22], and, s_{ij}^{PLM} from ProtBERT [40], as optimization objectives for the ILP problem (5).

Our method can be used to design new antibody candidates, where mutations are required on specific, or various, regions across the heavy on light chains of the wild-type antibody. During experimentation, we use the Trastuzumab antibody, and consider only heavy chain mutations on the CDR3 region of the wild-type.

To generate a batch of diverse sequences for each experiment, we apply constraints to the number of solutions containing a given position (12) and to the solutions containing a given mutation per position (13). These constraints ensure any one mutation or position is not overly represented in the final batch. We enforce a maximum, n_{max} and minimum, n_{min}, number of mutations from wild-type, to ensure a library of variation with respect to mutation length.



(a) Step 1 (b) Step 2

(c) Step 3 (d) Step 4

(e) Step 5 (f) Step 6

Figure 3: Illustration of the *solve-and-remove* strategy for diversification. The algorithm finds the best solution to the problem and then removes a neighborhood of solutions to ensure diversity (a ball of radius ϵ around the solution). The algorithm proceeds by solving new problems with reduced search spaces until the desired number of solutions is reached.

4.1 Results

For evaluation, we compare the ILP with the *Linear Mutant Generator* (LMG) algorithm [7], the *Strength Pareto Evolutionary Algorithm 2* (SPEA2) [41], and the *ML-optimized library design with improved fitness and diversity* (MODIFY) method [17]. See Appendix A.2 for extensive details on baselines, including implementation details. Figure 2 shows the libraries generated by each algorithm projected onto the objective space and Table 1 presents the compiled evaluation metrics.

In Figure 2 we observe that the ILP libraries are concentrated in the bottom-left corner of the objective space, near the Pareto front (PF). The mutants obtained by the ILP with diversity constraints are organized in *stratified layers* separated by the effect of diversity constraints, while the ILP without diversity constraints presents a single layer of mutants. In contrast, the LMG and MODIFY libraries distributions resemble the trace of a *shotgun* across the objective space. The SPEA2 libraries exhibit intermediate performance but face a significant limitation in generating the required library size, as shown in Table 1. This is due to the presence of non-unique sequences in the final population (targeted at 1,000 individuals), resulting in less diverse libraries. For the problem of designing a $1,000$-mutant library, the ILP variants outperform all other methods in multi-objective metrics BEU and HV (see Appendix A.3 for definitions) while maintaining entropy values larger than 3. A trade-off between entropy and BEU values is observed in the ILP libraries with and without diversity constraints.

To further illustrate each algorithms ability to optimize the objective values, we present the Pareto front for the ILP and each baseline algorithm in Figure 4. The volume of the computed Pareto front with respect to a reference vector ($\mathbf{V}_{ref} = [50, 50]$) corresponds to the HV score in Table 1. The ILP and SPEA2 algorithms compute the Pareto front that captures the best objective values compared to the other baselines. The ILP and SPEA2 Pareto fronts span greater ranges of the objective space. Furthermore, both the ILP and SPEA2 algorithms compute the approximated Pareto front in Table 1, resulting in the ILP, SPEA2, and Pareto front having the same HV score. As a result, the ILP

Method	# of unique sequences ↑	Residue entropy ↑	BEU ↓	HV ↑	Average humanness ↑	Oracle fitness (%) ↑
ILP (1000, 300)	1,000	3.22	4.615	2231.90	-1296.73	58.2
ILP (no diversity)	1,000	3.11	4.30	2231.90	-1293.68	61.8
LMG [7]	1,000	4.75	10.70	1937.70	-1309.75	17.1
MODIFY [17]	1,000	3.97	10.14	2012.43	-1308.27	18.6
SPEA2 (*strat.*)*	263	2.74	6.24	2231.90	-1290.01	38.02
SPEA2 (*unified*)*	69	2.60	4.08	2231.90	-1282.57	68.12
Pareto Front†	10	2.21	3.55	2231.90	-1282.31	80.0

Table 1: Diversity and fitness of libraries generated by each algorithm. *SPEA2 methods [41] were unable to generate libraries of the required size, making direct comparison with other algorithms difficult, though we include their results for reference. †We approximate the Pareto front by combining the Pareto front of each algorithm's solutions set and removing the Pareto dominated solutions.

and SPEA2 algorithms are both able to optimize the respective objective values while maintaining diversity.

The last two columns in Table 1 are used as external evaluation metrics for the library design problem. Following [42], we consider the log-likelihood of the mutants under ProtGPT2 [43] as a proxy for developability (called average humanness). The oracle fitness is the percentage of mutants predicted to bind to the target antigen by a ML-classifier trained on experimental data (see Appendix A.3 for details). We observe $\simeq 3.5$-fold increase in predicted oracle fitness of the ILP mutants with higher average humanness compared to the LMG and MODIFY mutants. A similar trade-off between entropy and oracle fitness as described above is observed within the ILP libraries. The ILP libraries present a comparable predicted oracle fitness to the SPEA2 libraries, while involving a 5 to 14 times larger libraries. Note that the derived PF has an oracle fitness of 80%, justifying the choice of scoring functions. Overall, the ILP method provides the best libraries under the multi-objective metrics BEU and HV, oracle fitness, and average humanness, while providing library size control and flexibility to adjust the trade-off between diversity and fitness.

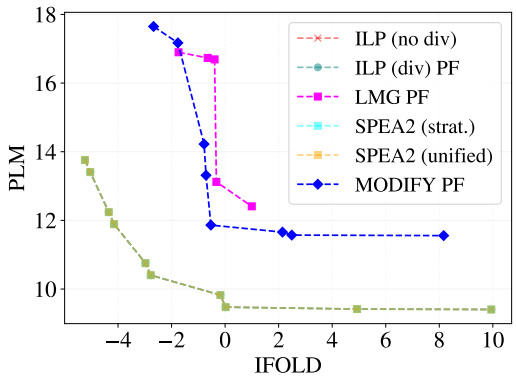


Figure 4: The individually computed Pareto front of the batches generated by the ILP and all baseline algorithms.

5 Conclusion and future work

In this work, we proposed a novel method for antibody library design that combines multi-objective optimization with diversity constraints. Our method leverages recent advances in sequence and structure-based machine learning models to compute *in silico* deep mutational scanning data that is fed into an integer linear programming solver to generate diverse and high-performing antibody libraries. In an extensive evaluation involving the Trastuzumab antibody, we showed that our method provides the best libraries in terms of multi-objective metrics, oracle fitness, and average humanness, while providing library size control and diversity-fitness trade-off flexibility.

As limitations, it is important to note that our method requires the structure of the antibody-antigen complex (which may not always be available) and that the quality of the generated libraries is affected by the quality of the scores predicted by the deep learning models. It is also important to note that the method might become computationally expensive for very large libraries, as the number of constraints in the ILP formulation grows linearly with the number of selected mutants.

In future work, we plan to extend our method to consider the breadth optimization problem, where the goal is to design antibodies that are effective against a set of divergent viral strains. We also plan to investigate the use of a quadratic assignment formulation to model the pairwise interactions between amino acids in the antibody-antigen complex.

Code availability

The code associated with this manuscript has been open-sourced to facilitate reproducibility and further research in the field. The repository contains the implementation of the *Solve-and-remove* Algorithm 1. It also includes scripts for data preprocessing and evaluation.

The code is available at the following URL: `https://github.com/LLNL/protlib-designer`. Please refer to the repository's README file for setup instructions, dependencies, and usage guidelines.

For any questions or issues regarding the code, please contact the corresponding author.

Acknowledgments and disclosure of funding

The GUIDE program is executed by the Joint Program Executive Office for Chemical, Biological, Radiological and Nuclear Defense (JPEO-CBRND) Joint Project Lead for Enabling Biotechnologies (JPL CBRND EB) on behalf of the Department of Defense's Chemical and Biological Defense Program. The views expressed in this publication reflect the views of the authors and do not necessarily reflect the position of the Department of the Army, Department of Defense, nor the United States Government. References to non-federal entities do not constitute or imply Department of Defense or Army endorsement of any company or organization. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC. LLNL-CONF-868784.

References

[1] P. Carter and G. A. Lazar, "Next generation antibody drugs: pursuit of the 'high-hanging fruit'," *Nature Reviews Drug Discovery*, vol. 17, pp. 197–223, 2017.

[2] R. Lu, Y.-C. Hwang, I. Liu, C.-C. Lee, H. zen Tsai, H.-J. Li, and H. Wu, "Development of therapeutic antibodies for the treatment of diseases," *Journal of Biomedical Science*, vol. 27, 2020.

[3] M. S. Morrison, C. J. Podracky, and D. R. Liu, "The developing toolkit of continuous directed evolution," *Nature chemical biology*, vol. 16, no. 6, pp. 610–619, 2020.

[4] S. Paul, D. Mytelka, C. Dunwiddie, C. C. Persinger, B. Munos, S. Lindborg, and A. Schacht, "How to improve rd productivity: the pharmaceutical industry's grand challenge," *Nature Reviews Drug Discovery*, vol. 9, pp. 203–214, 2010.

[5] R. J. Fox, A. K. Roy, S. Govindarajan, J. Minshull, C. Gustafsson, J. T. Jones, and R. Emig, "Optimizing the search algorithm for protein engineering by directed evolution.," *Protein engineering*, vol. 16 8, pp. 589–97, 2003.

[6] V. K. Sharma, T. Patapoff, B. Kabakoff, S. Pai, E. Hilario, B. Zhang, C. Li, O. Borisov, R. Kelley, I. Chorny, J. Zhou, K. Dill, and T. Swartz, "In silico selection of therapeutic antibodies for development: Viscosity, clearance, and chemical stability," *Proceedings of the National Academy of Sciences*, vol. 111, pp. 18601 – 18606, 2014.

[7] T. A. Desautels, K. T. Arrildt, A. T. Zemla, E. Y. Lau, F. Zhu, D. Ricci, S. Cronin, S. J. Zost, E. Binshtein, S. M. Scheaffer, *et al.*, "Computationally restoring the potency of a clinical antibody against omicron," *Nature*, pp. 1–8, 2024.

[8] G. Citovsky, G. DeSalvo, C. Gentile, L. Karydas, A. Rajagopalan, A. Rostamizadeh, and S. Kumar, "Batch active learning at scale," *Advances in Neural Information Processing Systems*, vol. 34, pp. 11933–11944, 2021.

[9] T. A. Whitehead, A. Chevalier, Y. Song, C. Dreyfus, S. Fleishman, C. D. Mattos, C. A. Myers, H. Kamisetty, P. Blair, I. Wilson, and D. Baker, "Optimization of affinity, specificity and function of designed influenza inhibitors using deep sequencing," *Nature Biotechnology*, vol. 30, pp. 543–548, 2012.

[10] P. Koenig, C. V. Lee, B. T. Walters, V. Janakiraman, J. Stinson, T. Patapoff, and G. Fuh, "Mutational landscape of antibody variable domains reveals a switch modulating the interdomain conformational dynamics and antigen binding," *Proceedings of the National Academy of Sciences*, vol. 114, pp. E486 – E495, 2017.

[11] S. Warszawski, A. Katz, R. Lipsh, L. Khmelnitsky, G. B. Nissan, G. Javitt, O. Dym, T. Unger, O. Knop, S. Albeck, R. Diskin, D. Fass, M. Sharon, and S. Fleishman, "Optimizing antibody affinity and stability by the automated design of the variable light-heavy chain interfaces," *PLoS Computational Biology*, vol. 15, 2019.

[12] D. M. Mason, S. Friedensohn, C. R. Weber, C. Jordi, B. Wagner, S. M. Meng, R. Ehling, L. Bonati, J. Dahinden, P. Gainza, B. Correia, and S. Reddy, "Optimization of therapeutic antibodies by predicting antigen specificity from antibody sequence via deep learning," *Nature Biomedical Engineering*, vol. 5, pp. 600 – 612, 2021.

[13] K. A. Barlow, S. Conchúir, S. Thompson, P. Suresh, J. E. Lucas, M. Heinonen, and T. Kortemme, "Flex ddg: Rosetta ensemble-based estimation of changes in protein-protein binding affinity upon mutation," *bioRxiv*, 2017.

[14] A. M. Sevy, S. Panda, J. E. Crowe Jr, J. Meiler, and Y. Vorobeychik, "Integrating linear optimization with structural modeling to increase hiv neutralization breadth," *PLoS computational biology*, vol. 14, no. 2, p. e1005999, 2018.

[15] F. E. Agamah, G. Mazandu, R. Hassan, C. Bope, N. E. Thomford, A. Ghansah, and E. Chimusa, "Computational/in silico methods in drug target and lead prediction," *Briefings in bioinformatics*, 2019.

[16] D. Zhu, D. H. Brookes, A. Busia, A. Carneiro, C. Fannjiang, G. Popova, D. Shin, K. C. Donohue, E. Chang, T. Nowakowski, J. Listgarten, and D. Schaffer, "Optimal trade-off control in machine learning–based library design, with application to adeno-associated virus (aav) for gene therapy," *Science Advances*, vol. 10, 2021.

[17] K. Ding, M. Chin, Y. Zhao, W. Huang, B. K. Mai, H. Wang, P. Liu, Y. Yang, and Y. Luo, "Machine learning-guided co-optimization of fitness and diversity facilitates combinatorial library design in enzyme engineering," *Nature Communications*, vol. 15, 2024.

[18] A. J. Riesselman, J. Ingraham, and D. Marks, "Deep generative models of genetic variation capture the effects of mutations," *Nature Methods*, vol. 15, pp. 816 – 822, 2018.

[19] J. Meier, R. Rao, R. Verkuil, J. Liu, T. Sercu, and A. Rives, "Language models enable zero-shot prediction of the effects of mutations on protein function," *bioRxiv*, 2021.

[20] B. L. Hie, V. R. Shanker, D. Xu, T. U. Bruun, P. A. Weidenbacher, S. Tang, W. Wu, J. E. Pak, and P. S. Kim, "Efficient evolution of human antibodies from general protein language models," *Nature Biotechnology*, 2023.

[21] J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. Ragotte, L. Milles, B. Wicky, A. Courbet, R. D. de Haas, N. Bethel, P. J. Leung, T. Huddy, S. Pellock, D. Tischer, F. Chan, B. Koepnick, H. Nguyen, A. Kang, B. Sankaran, A. Bera, N. King, and D. Baker, "Robust deep learning based protein sequence design using proteinmpnn," *Science (New York, N.Y.)*, vol. 378, pp. 49 – 56, 2022.

[22] M. H. Høie, A. M. Hummer, T. H. Olsen, B. Aguilar-Sanjuan, M. Nielsen, and C. M. Deane, "Antifold: Improved antibody structure-based design using inverse folding," 2024.

[23] F. Wu, Y. Tao, D. Radev, and J. Xu, "When geometric deep learning meets pretrained protein language models," 2022.

[24] S. Shan, S. Luo, Z. Yang, J. Hong, Y. Su, F. Ding, L. Fu, C. Li, P. Chen, J. Ma, X. Shi, Q. Zhang, B. Berger, L. Zhang, and J. Peng, "Deep learning guided optimization of human antibody against sars-cov-2 variants with broad neutralization," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 119, 2022.

[25] A. M. Hummer, C. Schneider, L. Chinery, and C. Deane, "Investigating the volume and diversity of data needed for generalizable antibody-antigen g prediction," *bioRxiv*, 2023.

[26] D. Jérémie, A. Flajolet, A. Marginean, A. Cully, and T. Pierrot, "Quality-diversity for one-shot biological sequence design," in *ICML'24 Workshop ML for Life and Material Science: From Theory to Industry Applications*.

[27] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arXiv preprint arXiv:1504.04909*, 2015.

[28] Y. Zeng, H. Elliott, P. Maffettone, P. Greenside, O. Bastani, and J. R. Gardner, "Antibody design with constrained bayesian optimization," in *ICLR 2024 Workshop on Generative and Experimental Perspectives for Biomolecular Design*.

[29] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *stat*, vol. 1050, p. 1, 2014.

[30] V. Saxena, R. Panicucci, Y. Joshi, and S. Garad, "Developability assessment in pharmaceutical industry: An integrated group approach for selecting developable candidates.," *Journal of pharmaceutical sciences*, vol. 98 6, pp. 1962–79, 2009.

[31] A. Jarasch, H. Koll, J. Regula, M. Bader, A. Papadimitriou, and H. Kettenberger, "Developability assessment during the selection of novel therapeutic antibodies.," *Journal of pharmaceutical sciences*, vol. 104 6, pp. 1885–1898, 2015.

[32] H. Shim, "Synthetic approach to the generation of antibody diversity," *BMB Reports*, vol. 48, pp. 489 – 494, 2015.

[33] C. Hsu, R. Verkuil, J. Liu, Z. Lin, B. Hie, T. Sercu, A. Lerer, and A. Rives, "Learning inverse folding from millions of predicted structures," *bioRxiv*, 2022.

[34] V. R. Shanker, T. U. J. Bruun, B. L. Hie, and P. S. Kim, "Inverse folding of protein complexes with a structure-informed language model enables unsupervised antibody evolution," *bioRxiv*, 2023.

[35] A. Kovaltsuk, J. Leem, S. Kelm, J. Snowden, C. M. Deane, and K. Krawczyk, "Observed antibody space: a resource for data mining next-generation sequencing of antibody repertoires," *The Journal of Immunology*, vol. 201, no. 8, pp. 2502–2509, 2018.

[36] B. H. Korte, J. Vygen, B. Korte, and J. Vygen, *Combinatorial optimization*, vol. 1. Springer, 2011.

[37] J. Forrest and R. Lougee-Heimer, "Cbc user guide," in *Emerging theory, methods, and applications*, pp. 257–277, INFORMS, 2005.

[38] P. Fishburn, "Letter to the editor - additive utilities with incomplete product sets: Application to priorities and assignments," *Oper. Res.*, vol. 15, pp. 537–542, 1967.

[39] T. Uçar, C. Malherbe, and F. Gonzalez, "Exploring log-likelihood scores for ranking antibody sequence designs," *bioRxiv*, 2024.

[40] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rehawi, Y. Wang, L. Jones, T. Gibbs, T. Feher, C. Angerer, M. Steinegger, *et al.*, "Prottrans: Toward understanding the language of life through self-supervised learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 10, pp. 7112–7127, 2021.

[41] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," report, ETH Zurich, Computer Engineering and Networks Laboratory, Zurich, 2001-05.

[42] N. Gruver, S. Stanton, N. Frey, T. G. Rudner, I. Hotzel, J. Lafrance-Vanasse, A. Rajpal, K. Cho, and A. G. Wilson, "Protein design with guided discrete diffusion," *Advances in neural information processing systems*, vol. 36, 2024.

[43] N. Ferruz, S. Schmidt, and B. Höcker, "Protgpt2 is a deep unsupervised language model for protein design," *Nature communications*, vol. 13, no. 1, p. 4348, 2022.

[44] C. F. Hayes, R. Rădulescu, E. Bargiacchi, J. Källström, M. Macfarlane, M. Reymond, T. Verstraeten, L. M. Zintgraf, R. Dazeley, F. Heintz, *et al.*, "A practical guide to multi-objective reinforcement learning and planning," *Autonomous Agents and Multi-Agent Systems*, vol. 36, no. 1, p. 26, 2022.

[45] A. Abels, D. Roijers, T. Lenaerts, A. Nowé, and D. Steckelmacher, "Dynamic weights in multi-objective deep reinforcement learning," in *International conference on machine learning*, pp. 11–20, PMLR, 2019.

[46] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, "DEAP: Evolutionary algorithms made easy," *Journal of Machine Learning Research*, vol. 13, pp. 2171–2175, jul 2012.

[47] B. K. Petersen, M. Landajuela, T. N. Mundhenk, C. P. Santiago, S. K. Kim, and J. T. Kim, "Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients," *arXiv preprint arXiv:1912.04871*, 2019.

[48] M. Landajuela, C. S. Lee, J. Yang, R. Glatt, C. P. Santiago, I. Aravena, T. Mundhenk, G. Mulcahy, and B. K. Petersen, "A unified framework for deep symbolic regression," *Advances in Neural Information Processing Systems*, vol. 35, pp. 33985–33998, 2022.

[49] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[50] B. E. Suzek, H. Huang, P. McGarvey, R. Mazumder, and C. H. Wu, "Uniref: comprehensive and non-redundant uniprot reference clusters," *Bioinformatics*, vol. 23, no. 10, pp. 1282–1288, 2007.

[51] L. M. Zintgraf, T. V. Kanters, D. M. Roijers, F. A. Oliehoek, and P. Beau, "Quality assessment of morl algorithms: A utility-based approach," in *Benelearn 2015: proceedings of the 24th annual machine learning conference of Belgium and the Netherlands*, vol. 5, 2015.

A Appendix

A.1 Antibody library design as a multi-objective problem

Designing a library with respect to *extrinsic fitness*, e.g. binding quality to the antigen target, does not ensure experimental success. Safe, stable, and manufacturable antibody candidates contain key properties related to *intrinsic fitness* e.g thermostability, developability, and stability. Without explicitly considering the *intrinsic fitness* related properties during optimization a generated library may contain candidates that overfit to the biases of the optimized *in silico* tool increasing the risk of experimental failure. As a risk mitigation strategy, we propose to optimize for *intrinsic fitness* and *extrinsic fitness* simultaneously as separate objectives.

Approaches that frame antibody library design as a multi-objective problem typically compute a Pareto front of solutions, or optimize the library for a fixed weighting over the problem objectives [17]. As we will show, the Pareto front does not contain sufficiently diverse solutions from which an adequate antibody library can be derived. Moreover, in a *zero-shot* setting, optimizing a library with respect to a fixed weighting over the objectives increases the risk of experimental failure. Objective weightings are extremely difficult to tune [44] and, as a further complicating factor, it is infeasible to determine if any selected weighting is appropriate given the absence of experimental fitness data. We utilize a dynamic weighting approach [45], whereby for each iteration a random weighting over the objectives is sampled from the distribution over all possible weightings, and used to compute a feasible solution for the given problem. Sampling weights mitigates the risk of over optimizing for any individual weighting, ensuring diversity and coverage over the space of objective weights.

A.2 Baseline Algorithms

Below we outline the LMG, SPEA2 and MODIFY algorithms, used as baselines to compare against our proposed ILP method. We also describe how an approximation of the Pareto front is computed, which we also use as a comparison.

Linear Mutant Generator The Linear Mutant Generator (LMG) baseline is adapted from [7]. The LMG uses a hierarchical sampling process. First, the number of mutations is sampled from the range $[5, 8]$ uniformly at random. Then, mutations are sampled without replacement according to the following probabilities,

$$p_{ij} = \frac{\sum_{q=1}^{|Q|} \sigma(s_{ij}^q)}{\sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{q=1}^{|Q|} \sigma(s_{ij}^q)},$$

where $\sigma(s)$ is the generalized logistic function $\sigma(s) = \frac{1}{(1+a \cdot e^{-s \cdot b})^{1/c}}$, with $a = 1000$, $b = 5$, and $c = 8$. The LMG algorithm can be used to generate a batch of K mutants.

Strength Pareto Evolutionary Algorithm 2 As a baseline, we implemented the Strength Pareto Evolutionary Algorithm 2 (SPEA2) [41], enhanced with an island model strategy. SPEA2 was chosen for its balance between convergence and diversity, crucial in exploring the complex design space of antibody optimization. The island model employed 20 independent sub-populations, each with 50 individuals, totaling 1,000 individuals across the entire population. Each sub-population evolved independently, with no migration, ensuring isolated genetic pools and promoting diverse exploration of the solution space.

A restart strategy was introduced to maintain diversity. If a sub-population's Pareto Front (PF) was a subset of another sub-population's PF, the sub-population was restarted by replacing all individuals with random solutions. This mechanism helped prevent premature convergence within any sub-population, encouraging ongoing exploration. The restart mechanism was disabled for the last 10% of generations to prevent random solutions from appearing in the final population.

In our SPEA2 implementation, each solution (antibody mutant) is represented as $\mathbf{y} = (y_1, \ldots, y_N)$, where $y_i = \mathcal{I}_A(x_i)$ and $x_i \in A$. Here, y_i indicates the index of the amino acid selected for the i-th mutable position from the set of amino acids A, and N represents the number of antibody positions permitted to mutate.

The crossover operator was a modified version of one-point crossover that respected the constraint of 5 to 8 amino acid mutations per antibody sequence. After the traditional one-point crossover was

applied, if the offspring exceeded the upper mutation limit, random mutations were reverted to the wild-type sequence. If the number of mutations fell below the lower threshold, additional random amino acid mutations were introduced. The probability of crossover was set to 70%.

A custom mutation operator was also used, which randomly mutated an amino acid at a position that already had a mutation to a different amino acid. This ensures the total number of mutations remained constant and within the specified range. The probability of mutation was 30% at the individual level and 5% at the position level. The algorithm ran for 500 generations, with the final population comprising all unique individuals of the joint sub-populations, ensuring a broad and diverse set of solutions. We used the SPEA2 implementation from the DEAP [46] Python package.

In our experiments, we used two SPEA2 setups called *stratified* and *unified*. In the *stratified* setup, we ran four separate SPEA2 instances, each constrained to find solutions with an exact number of mutations (5, 6, 7, or 8). The final antibody library was composed of all unique solutions from the combined selections of these instances, ensuring diversity in the number of mutations. In the *unified* setup, a single SPEA2 instance was run across the entire mutation range [5, 8].

ML-optimized library design with improved fitness and diversity The MODIFY algorithm was introduced in [17], following the work in [16]. The method optimizes a tensor $\phi \in \mathbb{R}^{N \times M}$ and uses it to build probabilities $p_{ij} = \exp(\phi_{ij}) / \sum_{k=1}^{N} \exp(\phi_{ik})$ for each mutable position i. The probability of a mutant is given by

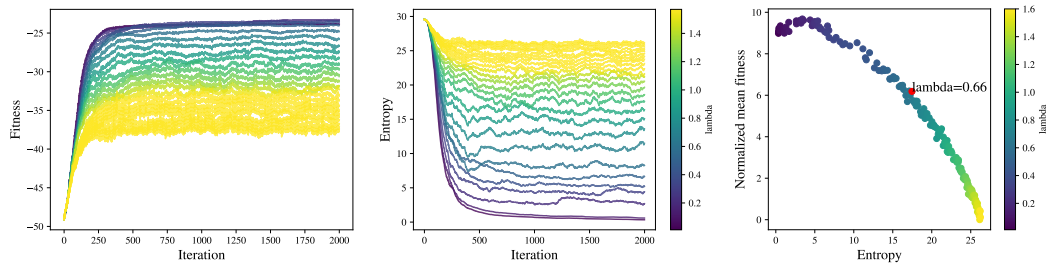
$$p(\mathbf{x}|\phi) = \prod_{i=1}^{N} \sum_{j=1}^{M} z_{ij} p_{ij}.$$

The optimization objective is,

$$J(\phi) = \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}|\phi)} [f(\mathbf{x})] + \lambda \sum_{i=1}^{N} \alpha_i \mathbb{H}(p_i), \tag{14}$$

where $f(\mathbf{x}) = \sum_{q=1}^{|Q|} w_q s^q(\mathbf{x})$ is the weighted sum of the scores of the $|Q|$ objectives, w_q is the weight of the q-th objective, $s^q(\mathbf{x})$ is the score of the q-th objective, and $\mathbb{H}(p_i)$ is the entropy of the marginal distribution p_i over amino acids at position i. The method sweeps through 200 λ values, optimizes the tensor ϕ for each λ value using policy gradient, and then selects the best λ value based on the area under the expected fitness and entropy curve. Objective (14) has been extensively used in other works for discrete optimization [47, 48].

Here, we adopt the MODIFY algorithm with $Q = \{\text{PLM}, \text{IFOLD}\}$, the scores defined in (3)-(4), and, as suggested in [17], $w_{\text{PLM}} = w_{\text{IFOLD}} = 0.5$. We train the models for $2,000$ iterations, with a batch size of $1,000$, and a learning rate of 10^{-1}. The final library of $1,000$ mutants was obtained by sampling from the optimal distribution $p(\mathbf{x}|\phi_{\lambda=0.66})$ (see Figure 5).



(a) Training curves for the MODIFY algorithm for different λ values.

(b) Optimal λ value.

Figure 5: (a) Evolution during training of the fitness and entropy of the MODIFY algorithm. (b) The optimal λ value maximizes the area under the curve of the expected fitness and entropy.

Approximating the Pareto front It is infeasible to exhaustively compute all valid solutions over the search space for $5 - 8$ point mutations. As a result, exactly computing the Pareto front is difficult without access to an exhaustive set of solutions. Therefore, we compute an approximation of the Pareto front by first computing the Convex hull of the Pareto front, then running Pareto optimality

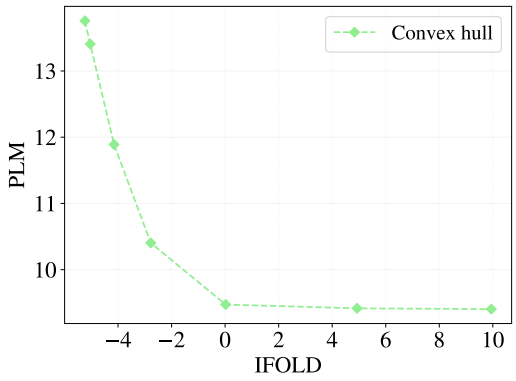


Figure 6: Convex hull computed using the ILP without the solve-and-remove algorithm. Generated using $1,000$ sampled linear weights.

filter on the combined sets of the Convex hull with the ILP, SPEA2, MODIFY and LMG runs. The resulting set of solutions is then used as the approximation of the Pareto front shown in Section 4.

The Convex hull is a subset of the Pareto front [44, 49]. The ILP is an optimal solver, meaning for any linear weighting the ILP can find the optimal solution that lies on the Convex hull. To compute the Convex hull, we run the ILP without the solve-and-remove algorithm, therefore the solution landscape remains static during optimization. To uncover the Convex hull, at each iteration we sample a different linear weighting and solve with respect to the sampled weighting. For our experiments, we sampled a total of 1,000 weightings. The resulting Convex hull is presented in Fig. 6. While the Convex hull contains many solutions that lie on the Pareto front, it does not contain solutions that lie on the concave regions of the Pareto front. Therefore, we initiate another step to approximate the remaining solutions in these missing regions.

To find the missing regions, we first compute the Pareto fronts of the solution set computed by each algorithm. In Fig. 4, it is clear that the solutions computed by the LMG and MODIFY algorithms are all Pareto dominated by the SPEA2 and ILP Pareto fronts. Both the ILP and SPEA2 runs compute identical Pareto fronts, which also contain all solutions on the Convex hull, and solutions in concave regions. As a result, we use the resulting Pareto front as our approximation presented in Section 4.

A.3 Evaluation metrics

To measure each individual batch produced by the ILP, MODIFY, LMG, and SPEA2 algorithm we utilized a number of metrics to compare the performance of each algorithm.

Evaluating extrinsic fitness To measure the extrinsic fitness of a batch of mutated sequences, we employ an oracle function to approximate the binding potential of a given mutated sequence to the target antigen. The oracle is a trained predictive classification model and will act as a surrogate for experimental validation. To train the oracle, we utilized extensive experimental binding data from [12][2]. The corresponding dataset contains $31,000$ experimentally validated and labelled mutated sequences, where $11,000$ bind to the target antigen (labelled as 1) and $20,000$ do not bind to the target antigen (labelled as 0). During training we use a random $70/10/20$ train/val/test split with binary cross-entropy loss for 20 epochs and a batch size of 16. The trained oracle network achieves 83% accuracy on the test set. To measure the extrinsic fitness of each batch, we report the percentage of sequences in the batch that are predicted by the oracle to bind to the target.

Evaluating intrinsic fitness Similarly to Gruver et al. [42], to measure the intrinsic fitness of a batch of mutated sequences we utilize the log-likelihood assigned by ProtGPT2 [43] (trained on Uniref50 [50]). The resulting score for each sequence represents how likely a given sequence is with respect to the model. The log-likelihood assigned by ProtGPT2 can be interpreted as a humanness score, where a lower score corresponds to a sequence being more "natural" or human-like. Therefore,

[2]All experimental data can be found here: https://github.com/dahjan/DMS_opt

we refer to this measure as humanness in Section 4. To calculate the humanness of a given batch, we compute the mean over the batch with respect to the humanness score.

Diversity metrics We use the residue entropy to measure the diversity of the batch. The residue entropy refers to the entropy of the empirical distribution over amino acids and positions of a batch, B. Consider the empirical distribution, $p_{ij} \propto \sum_{k=1}^{K} z_{ij}^{(k)}$, the residue entropy of the batch is defined as

$$\mathbb{H}_{\text{res}}(B) = -\sum_{i=1}^{N}\sum_{j=1}^{M} p_{ij} \log\left(p_{ij}\right).$$

As entropy increases a distribution becomes uniform. As a result, we aim to maximize entropy to compute a batch that is uniform over possible amino acids and mutable positions, therefore making the batch more diverse.

Multi-objective metrics To measure the ability of each algorithm to optimize each objective we utilize the *hypervolume* metric and a *based expected utility* metric.

The hypervolume (HV) metric measures the volume in vector space of a given Pareto front, which correlates to the spread of a given undominated set over the possible multi-objective solutions space. The HV is defined as

$$\text{HV}(\text{PF}_B, \mathbf{V}_{ref}) = \bigcup_{\mathbf{x}\in\text{PF}_B} \text{Volume}(\mathbf{V}_{ref}, \mathbf{x}),$$

where $\text{Volume}(\mathbf{V}_{ref}, \mathbf{x})$ is the volume of the hypercube spanned by the reference vector, $\mathbf{V}_{ref}$, and the solution vector $\mathbf{x}$ in the Pareto front of a given batch, PF_B [44]. To measure the HV in our experiments we set $\mathbf{V}_{ref} = [50, 50]$.

While the hypervolume metric has been used extensively in the literature, it has some limitations. Specifically, the hypervolume metric requires a reference point to compute the score. The choice of reference point is arbitrary and effects the resulting score.

Many multi-objective measures focus on evaluating an algorithms ability to compute a coverage set of non-dominated solutions e.g. Pareto front [49]. However, for antibody library design we are concerned with measuring the quality with respect to the objective values over a batch of solutions. To do so, we implement a *batch expected utility* metric (BEU) that leverages the expected utility metric (EUM) proposed by Zintgraf et al. [51]. The BEU metric produces a scalar representation of the expected utility over a given batch of solutions with respect to a distribution over utility functions, and can be defined as follows:

$$\text{BEU} = \mathbb{E}_{\mathbf{x}\sim B}[\mathbb{E}_{u\sim P(\cdot|\mathcal{U})}[u(\mathbf{x})]],$$

where u is a utility function drawn from a set of utility functions $\mathcal{U}$, and $\mathbf{x}$ is a mutant in the batch, B.

To compute the BEU metric, a distribution over utility functions must be known a priori. Therefore, we assume a distribution over linear utility functions and sample from a simplex over all possible convex combinations. In our experiments, we sample $10,000$ utility functions (linear scalarization weights) and compute the BEU metric for each algorithm in Table 1.

A.4 Ablations

In Table 2, we show results of ablations of ILP experiments with different diversity constraints and mutation shuffling configurations. In Table 2, fitness is measured as the percentage of the generated library predicted to bind the antigen HER2 target using the binding prediction surrogate. For experiment 1, we evaluate the ILP without any diversity constraints and without mutation shuffling enabled. In this case, the ILP is greedy and selects sequences only with respect to the weighted AntiFold and ProtBERT scores, and achieves a fitness of 61.8%. For evaluations 2-6, we introduce diversity constraints, and vary them for each experiment respectively. We focus on varying the mutational constraint, δ_2, from 500 to 100, to evaluate the effect of this constraint on the fitness and diversity of a given batch. For $\delta_2 = 500$ and $\delta_2 = 400$ a small change in the fitness and diversity of the batch is observed. However, as the δ_2 constraint is reduced further ($\delta_2 < 400$) the predicted fitness of each batch decreases. Simultaneously, as the δ_2 constraint is reduced, the entropy of each

Exp. id	Diversity constraints	δ_1	δ_2	Shuffle	Entropy	Pred. Fitness (%)
1	×	NA	NA	×	3.11	61.8
2	✓	1000	500	×	3.10	62.9
3	✓	1000	400	×	3.12	66.2
4	✓	1000	300	×	3.22	58.2
5	✓	1000	200	×	3.47	44.5
6	✓	1000	100	×	4.02	28.5
7	×	NA	NA	✓	2.96	40
8	✓	1000	500	✓	3.09	39.7
9	✓	1000	400	✓	3.21	37.4
10	✓	1000	300	✓	3.33	32.8
11	✓	1000	200	✓	3.66	22.9
12	✓	1000	100	✓	4.26	13.4

Table 2: Ablation of ILP experiments with various diversity parameters and the resulting predicted fitness and entropy scores.

batch increases resulting in more diverse batches. A trade-off between predicted fitness and diversity is observed, where generating batches of higher diversity results in a lower predicted fitness.

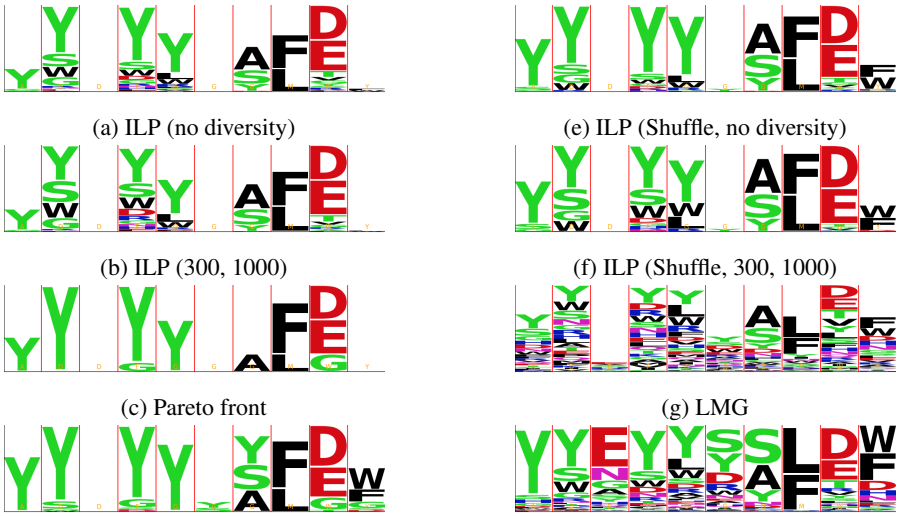
Fig. 8a visualizes the scores for the problem objectives for each mutated sequence computed by ILP experiments, where experiments 1 and 4 are shown in Fig. 8a from Table 2. Additionally, we compute the Pareto front, represented in Fig. 8 as a dashed line. Both ILP configurations find all sequence mutations on the Pareto front. Experiment 1 (red), the ILP instance without diversity constraints, greedily selects sequences close to, or on, the PF. It is clear, the solve-and-remove algorithm (Algorithm 1) enables some level of diversity even without diversity constraints being explicitly configured. Without the solve-and-remove algorithm, only solutions on the PF would be selected. In contrast, experiment 4 (teal), the ILP instance with diversity constraints, produces a batch of sequences where some solutions lie close to the PF, but many solutions span different regions of the objective space, resulting in more pronounced diversity. The diversity for experiment 4 is visualized as a sequence logo in Fig. 7b. The entropy of experiment 4 is 3.22, and is higher (i.e., more diverse) when compared to the entropy score of 3.11 for experiment 1.

Next, we investigate the impact of mutational shuffling on the batches generated by the ILP. For experiments 7-12, we introduce mutational shuffling with and without diversity constraints. Notably, the diversity of each batch with mutational shuffling enabled increases when compared to the corresponding experiments without mutational shuffling. Fig. 8b presents the objective scores for experiment 7 and 10, where the objective score values are distributed differently when compared to their corresponding experiments in Fig. 8a. For example, the solutions generated in experiment 7 are much further from the PF when compared to the solutions generated in experiment 1 which lie close to the PF. Additionally, the solutions generated in experiment 10 are scattered throughout the objective space. As a result, mutational shuffling impacts the fitness of each batch, given all experiments with mutational shuffling have a lower fitness compared to their corresponding experiment. Mutational shuffling ensures more diversity in the point mutations of the given batch and without mutational shuffling this level of diversity cannot be guaranteed.

A.5 Further baseline comparisons & discussion

Below we extend our comparison of the ILP runs against with LMG, MODIFY, and SPEA2. We also use an approximation of the Pareto front for further comparisons.

Unique sequences Library design typically requires a fixed number of sequences to be generated to fit a capacity for experimental validation. Therefore, it is important to have control on the number of unique sequence generated by a library design algorithm. Using the ILP, MODIFY, and the LMG it is possible to generate a library of fixed size, given each algorithm generates the required library size of $1,000$. However, the SPEA2 algorithm struggles to generate batches that meet this requirements. SPEA2 (joint) generates a batch of 69 unique sequences, while SPEA2 (independent) generates a batch of 263 sequences. Although the initial population size of the SPEA2 runs was set to $1,000$ many similar, or the same solutions, were generated or dropped due to the algorithms optimization



(a) ILP (no diversity)

(b) ILP (300, 1000)

(c) Pareto front

(d) SPEA2

(e) ILP (Shuffle, no diversity)

(f) ILP (Shuffle, 300, 1000)

(g) LMG

(h) MODIFY

Figure 7: Sequence logo plots of batches generated from various experiments for the ILP, each respective baseline, and the Pareto front.

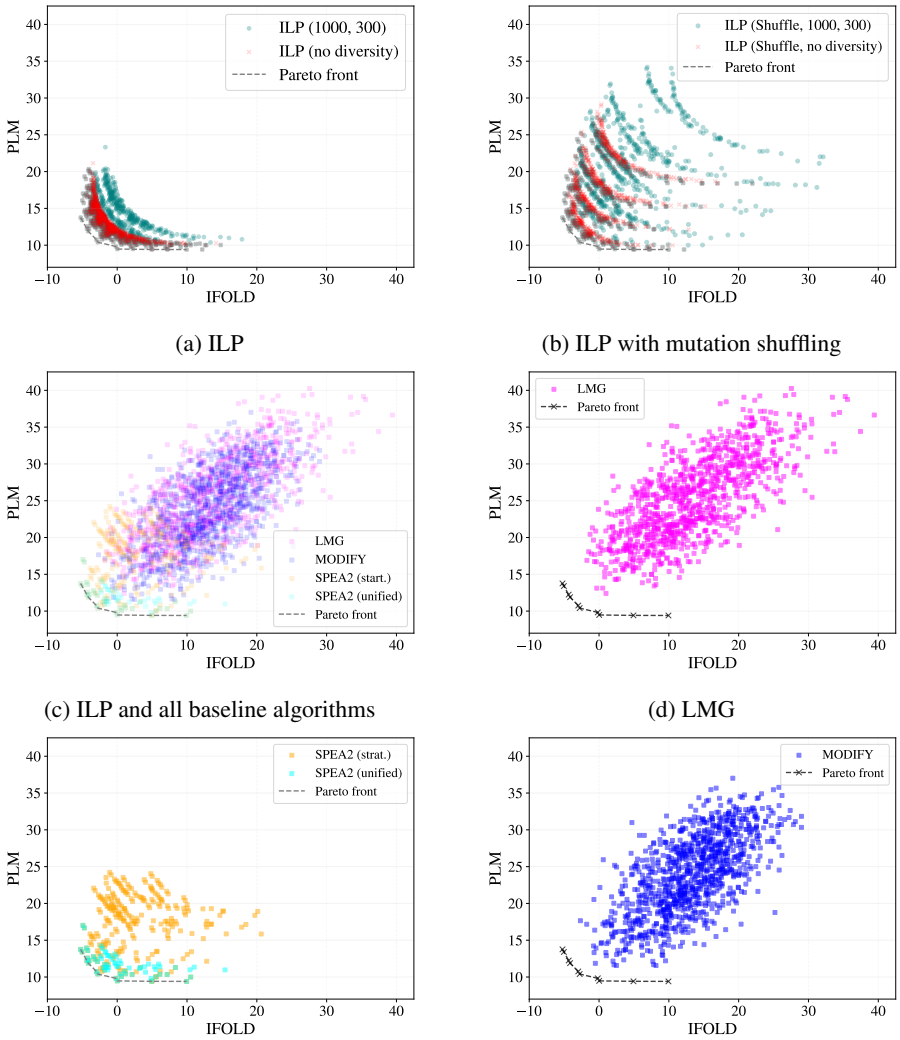
process, resulting in smaller batch sizes. As a result, the inability to control the exact size of the library being generated limits the SPEA2 algorithms use in practical settings.

Residue entropy & oracle fitness The measured entropy of the distribution over amino acids and positions is utilized score the diversity of a given batch, where a diverse batch of sequences maximizes the entropy of the distribution. We also use the predicted oracle fitness of the batch using a trained predictive model (see Section 4) to measure the extrinsic fitness of each batch. The measured entropy of the LMG generated library is 4.75, resulting in a highly diverse batch of solutions. Similarly, the MODIFY algorithm has a residue entropy score of 4. Both LMG and MODIFY achieve higher diversity scores when compared to ILP experiments presented in Table 2. Fig. 7g and Fig. 7h display the sequence logo plot of the batch produced by the LMG and MODIFY. The difference in diversity obvious when comparing the MODIFY and LMG logo plots with those of the ILP runs (Fig. 7a, Fig. 7b, Fig. 7e, and Fig. 7f). Each position contains a number of amino acids, resulting in a highly diverse batches. However, LMG and MODIFY have lower predicted oracle fitness scores, 17.1 and 18.6, when compared to the ILP runs (61.8 and 58.2) in Table 1.

Both SPEA2 runs also have lower diversity scores when compared to the ILP runs. SPEA2 (joint) has a score of 2.6 and SPEA2 (independent) has a score of 2.74. This is reflected in Fig. 7d, where many amino acids are missing from the logo plot at each mutated position. The SPEA2 (joint) has a fitness score of 68.12 and SPEA2 (independent) has a fitness score of 68.12. Both SPEA2 runs have a lower diversity score when compared to the ILP runs. However, the SPEA2 (independent) run has a higher fitness score than the ILP. The SPEA2 (independent) batch only has 69 sequences, where 10 of those sequences lie on the Pareto front. Therefore, the results from the SPEA2 runs are not directly comparable. The ILP produces a much larger batch when compared to both SPEA2 runs and maintains a high level of both diversity and fitness.

Given multiple objectives are used during optimization, we approximate the Pareto front of the underlying problem and compare the set of sequences in the Pareto front with the ILP and LMG. The set of sequences on the Pareto front have high fitness (80%) but very low diversity (2.21). Although the Pareto front has fit solutions, the Pareto front is not sufficiently diverse to use as a batch for experimental validation. However, we should aim to include the Pareto front as a subset of solutions contained with the overall batch.

The diversity parameters for the LMG and MODIFY algorithms are not configurable, resulting in algorithms that always produce batches with high diversity. Similarly for SPEA2, certain parameters can be altered to enforce diversity, however, the diversity of the resulting set is not directly controllable. In contrast to all other baselines, the ILP can be configured to generate more or less diversity within



(a) ILP

(b) ILP with mutation shuffling

(c) ILP and all baseline algorithms

(d) LMG

(e) SPEA2

(f) MODIFY

Figure 8: The objective values of the batch of sequences generated by the ILP, LMG, SPEA2, and MODIFY algorithms. Each point corresponds to a unique mutated sequence.

a batch when required. Given various library design scenarios can arise where varying levels of diversity may be necessary, having the ability to configure the diversity parameters as required is important.

Fig. 9 presents the trade-off between fitness and diversity with respect to the ILP and each baseline. Each baseline is shown as a single point to reflect the fitness and diversity score of the final batch. The ILP contains multiple points representing each ablation performed. It is clear, that to increase fitness scores the diversity of the batch must decrease. As diversity increases, the fitness of the batch also declines. Therefore it is important to have control over the diversity of a batch, given depending on the objectives certain diversity parameters may be required.

Humanness While the typically objective of library design is to generate a batch of antibodies that bind to the given target, it is also crucial ensure that a given antibody library contains sequences with properties related to *intrinsic fitness* – like developability, human-likeness, etc. – to ensure experimental success. Sequences without properties related to *intrinsic fitness* may not be developable, human-like, or stable. To generate a library with these properties, we optimize the PLM scores as previously outlined. We measure the humanness of a given batch using the log-likelihood scores

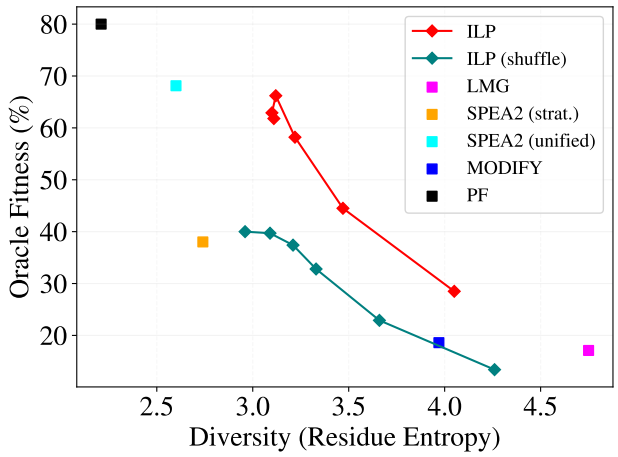


Figure 9: Predicted oracle fitness and residue entropy based diversity of each ILP ablations against the baseline algorithms.

from a PLM, specifically ProtGPT2. The LMG and MODIFY generate libraries with scores of -1309.75 and -1308.27. In comparison, the ILP runs have humanness scores of -1293.68 and -1296.73. Additionally, the SPEA2 algorithm produces batches with -1290.01 and -1282.57. We aim to maximize the humanness score of a given batch. The LMG and MODIFY scores are worse performing when compared to the ILP and SPEA2 scores. Again, it is important to note the SPEA2 batches contain fewer sequences when compared to all other baselines, average humanness scores for a given batch are not directly comparable. However, given the scores of both the ILP and SPEA2 batches both libraries of sequences are more likely to contain the *intrinsic fitness* properties related to developability when compared to LMG and MODIFY. A library designed to contain developable (human-like) sequences may avoid the biases or failure modes that can arise when optimizing specifically for binding. A resulting library may be more likely to avoid experimental failure by ensuring the key properties related to intrinsic fitness are contained within the library.

Hypervolume To understand how effectively each objective is being optimized, we compute and compare the hypervolume (HV) of the Pareto front of the batch of solutions computed by the ILP and all baselines. The Pareto fronts computed by all algorithms are presented in Fig. 4, where the LMG (HV = 1937.7) and MODIFY (HV = 2012.43.7) algorithms struggle to compute solutions that effectively minimize the objectives. Both ILP and SPEA2 runs uncover the full approximated Pareto front which contains the Convex hull (see Fig. 6), resulting in an equal HV score of 2231.90. The ILP and SPEA2, contain good approximations of the Pareto front, given they both contain the computed Convex hull. Therefore, both the ILP and SPEA2 effectively optimize the problem objectives.

Expected utility of the batch To further measure the quality of each batch with respect to the defined objectives we compute the *batch expected utility* (BEU) score for each algorithm. Here, we aim to minimize the BEU because a lower BEU score corresponds to a batch that effectively minimizes the objective values across a range of utility functions. Both LMG (BEU $= 10.7$) and MODIFY (BEU $= 10.14$) achieve a similar BEU score. However, the SPEA2 and ILP BEU scores are much lower when compared to the LMG and MODIFY. The batch produced by the SPEA2 (joint) algorithm achieves the lowest BEU score, however, as previously stated, the batch contains only 69 sequences making the BEU score for SPEA2 runs not directly comparable. The ILP runs achieve slightly higher scores of 4.30 and 4.612, however these scores are achieved over much larger batches. As expected, the Pareto front has the lowest BEU score.

Diversity of the Pareto front Many multi-objective optimization methods aim to compute the Pareto front [44, 49], where the Pareto front is the set of Pareto non-dominated solutions. Pareto dominance enforces diversity in the objective space, given in the Pareto front no two solutions can be the same, and, by the nature Pareto dominance, the objective values within the set span the range of feasible solutions. Figure 7c displays the positional and mutational diversity of the Pareto front

some positions contain no mutations while others contain little mutational variation. Additionally, the PF has a lowest measured entropy value of 2.21. Given the lack of diversity of the set of solutions on the Pareto front it is important to explicitly optimize for diversity in sequence space. As a result, solutions that may be Pareto dominated can potentially be included in the final batch given these solutions may promote diversity.

ILP solve-and-remove enables computation of the Pareto front Methods that utilize a linear scalarization are known to only be able to recover solutions that lie on the convex regions of the Pareto front, i.e the Convex hull [49]. ILP algorithms that use a weighted-sum scalarization are limited in this manner. However, the solve-and-remove algorithm enable the ILP to compute solutions that lie in concave regions on the Pareto front, and in our experiments recovered the full Pareto front. The solve-and-remove algorithm alters the shape of the solution landscape ensuring that previously selected solutions are removed from consideration. Throughout the execution of the algorithm, the landscape changes, and as solutions are removed regions of the landscape becomes convex. Therefore, previously concave regions become convex allowing for the ILP to compute the given solution using the weighted sum methods.