Pareto Front Training for Multi-Objective Symbolic Optimization

Jonathan G. Faris[1,2], Conor F. Hayes[1], Andre R. Goncalves[1], Kayla G. Sprenger[2], Daniel Faissol[1], Brenden K. Petersen[1] Mikel Landajuela[1], and Felipe Leno da Silva[1]
[1] Lawrence Livermore National Laboratory, Livermore, USA.
[2] University of Colorado, Boulder, USA.
{jonathan.faris,kayla.sprenger}@colorado.edu,{hayes56,goncalves1,faissol1,bp,landajuelala1,leno}@llnl.gov

ABSTRACT

Although Symbolic Optimization (SO) solutions have successfully been used in applications ranging from Neural Architecture Search to Antibody Therapeutics Optimization, current SO algorithms are typically limited to using a single quality measure to search for optimal solutions. However, for many applications, solutions are more naturally described by multiple measures, e.g., a solar panel must be designed to maximize power generation while minimizing heat generation. Herein, we propose Pareto Front Training (PFT), a SO algorithm that searches for token sequences by training a Recurrent Neural Network on the Pareto front of explored solutions. We evaluate PFT in an antibody optimization scenario using a real SARS-CoV-2 viral strain and show that PFT outperforms the baselines in terms of antibody binding quality, stability, and humanness. We hope PFT will inspire a new family of multi-objective SO algorithms and will help SO achieve varied new applications.

KEYWORDS

Symbolic Optimization, Multi-Objective Learning, Antibody Therapeutics Development

1 INTRODUCTION

The field of symbolic optimization (SO) aims to derive token sequences which describe a solution for a given problem. To uncover an appropriate solution, SO algorithms search the space of possible solutions by efficiently evaluating discrete token sequences. SO has been successfully applied to a diverse set of problems including power converter design [37], pharmaceutical formulation [6], underwater vehicle modeling [47], and others [29, 39]. State-of-the-art methods in this field solve the problem by modelling the SO task as a Deep Reinforcement Learning (DRL) problem, where each token is sampled sequentially (action) based on the partial sequence sampled so far (state) and the token sequence quality is assessed when a complete solution is achieved (reward).

To date, SO methods have been focusing on solving problems by optimizing a single scalar reward [28]. However, in practice, many real-world problems have multiple, often conflicting, objectives [10, 44] (e.g. operators of a wind turbine may aim to maximize power output while minimizing stress on the turbine components). In such settings optimal solutions are derived by optimizing with respect to a human decision maker's preferences over objectives, also known as a utility function [14]. The simplest way to fit SO methods to solve those problems is to a handcraft a scalarization function linearly combining objectives. However, this human decision maker's scalarization utility function may be unknown or can be difficult to specify *a priori*, making it infeasible to solve many problems [14, 34]. It is more desirable instead to present a set of solutions that are considered optimal to a human decision maker, who will be responsible to pick a solution that best reflects their preferences [14]. For example, a patient selecting a treatment may want to maximize the efficacy of the treatment, while minimizing the side effects. In this setting, it may be difficult for the user to exactly specify their preferences *a priori*. Therefore, a set of optimal treatments must be computed and presented to the user, allowing the patient to select their preferred treatment. However, SO methods do not have this capability at the moment.

To compute a set of optimal solutions for multi-objective symbolic optimization problems, we propose a new algorithm, Pareto Front Training (PFT). Our method uses an explicitly multi-objective approach, whereby PFT computes and trains a Recurrent Neural Network (RNN) on a Pareto-efficient data set based on all solutions explored up to the current timestep. To construct a data set of Pareto non-dominated solutions, at each timestep a RNN is used to generate a batch of solutions. Each solution is then evaluated using a multi-objective reward function. After each iteration, PFT updates a Pareto-efficient solution set by considering newly-evaluated samples generated from the RNN. PFT then uses the updated Pareto-efficient solution set to train and update the parameters of the RNN. PFT iterates over this approach, and over time our method is able to approximate the Pareto front in SO settings.

To evaluate our method, experiments are performed in a complex and relevant SO domain: antibody optimization, where we aim to discover highly-effective SARS-CoV-2 antibodies considering effectiveness (binding), stability, and safety (humanness). We benchmark the performance of PFT against relevant baseline algorithms from the SO literature and show that PFT clearly outperforms the baselines in this challenging task.

2 RELATED LITERATURE

There are several strategies that can be employed to solve a SO problem (described in detail in Section 3.1). One of the most popular in the literature is the use of genetic algorithms [6, 13]. We have chosen Non-dominated Sorting Genetic Algorithm II (NSGA-II) as our representative of this group in our experimental evaluation and, as shown in our results, those algorithms have very high sample complexity, which becomes a challenge in domains such as antibody optimization where evaluating solutions is expensive.

Another group of solutions deeply specializes the method to the application at hand, handcrafting heuristics to bias the search towards more promising token sequences [19, 24, 41, 42]. AI Feynman [43], which derived all 100 equations from the Feynman Lectures on Physics, is a famous example of this group. Unfortunately, most of

Proc. of the Adaptive and Learning Agents Workshop (ALA 2024), Avalos, Milec, Müller, Wang, Yates (eds.), May 6-7, 2024, Online, https://ala2024.github.io/. 2024.

those methods are specialized to Symbolic Regression, the flagship application of SO, and cannot be easily adapted to additional applications, such as Antibody Optimization. We are mostly interested in general purpose approaches.

The last group leverages machine learning to solve the SO problem. Methods such as Priority Queue Training (PQT; our base algorithm) and Deep Symbolic Optimization (DSO; added as a baseline), demonstrate the flexibility and efficiency of machine learning models when employed to SO. Those models achieved varied applications [36, 37, 40, 47] and can be easily combined with other methods, forming hybrid approaches [21, 22, 26, 38].

One major limitation of all those aforementioned SO approaches was that most of them (except genetic algorithms) could not handle multiple objectives, and PFT is our contribution to fill this gap.

Many multi-objective approaches optimize with respect to known nonlinear utility functions [15, 32, 33]. However, a nonlinear utility function must be known *a priori*, and as a result this is not always possible and a set of solutions that are optimal for all utility functions must be computed. PFT computes a set of solutions and therefore does not need to have a utility function specified *a priori*. Other approaches learn robust policies for all linear utility functions [1]. Many other multi-objective methods also learn a set of optimal solutions like the Convex-hull [4], Pareto front [31], and Distributional sets [16, 35]. While these approaches are similar to the work presented in this paper, the highlighted methods are defined for reinforcement learning and planning settings and aim to compute policies (probability distributions that map states to actions). The goal of the SO setting is to compute symbolic solutions, and as a result the aforementioned methods cannot be utilized in SO settings. The poor performance of standard RL-based on SO tasks has been discussed elsewhere [21]. Herein, we selected baselines we expected to succeed on the SO task at hand. In conclusion, to the best of our knowledge our contribution represents the first general purpose, dedicated multi-objective algorithm for SO.

3 BACKGROUND

In this section, we describe the relevant background needed to understand our approach. In the following text, we provide definitions for Symbolic Optimization, Priority Queue Training, and Multi-Objective Optimization.

3.1 (Deep) Symbolic Optimization

Broadly, Symbolic Optimization (SO), involves finding solutions which consist of a discrete, symbolic sequence of tokens to maximize a scoring function. Given a library of tokens $\mathcal{L} = \{\lambda^1, \ldots, \lambda^n\}$, we can construct a sequence $\tau = \langle \tau_1, \ldots, \tau_n \rangle$ (where τ_i represents the token at position i), which would represent a potential solution to the problem under investigation. In general, τ may be of arbitrary length, and may have multiple copies of the same token, λ^i. For the purposes of this study, the sequence length is known. Upon generating a sequence, a scoring function, or reward signal, is then calculated $\mathcal{R} : \tau \rightarrow \mathbb{R}$. All combinations of tokens which result in a valid sequence can then be scored according to their fitness via the reward function $\mathcal{R}(\tau)$. Thus, the solution to a symbolic optimization problem takes the form:

$$\underset{n \in \mathbb{N}, \tau}{\text{argmax}}[\mathcal{R}(\tau)] \text{ with } \tau = \langle \tau_1, \ldots, \tau_n \rangle, \text{ and where } \tau_i \in \mathcal{L} \quad (1)$$

That is, SO aims at finding the sequence that optimizes the reward function. Therefore, SO consists of a search problem where we have to efficiently search a huge space of token sequences. This problem can be solved in several ways [6, 19, 22, 24, 36, 37, 41, 42, 47], as discussed in Section 2. Currently, using machine learning techniques to sample sequences is considered to be the state-of-the-art strategy.

The Deep Symbolic Optimization (DSO) [28] algorithm is considered to be the state-of-the-art method after achieving first place on the real-world track of the 2022 SRBench competition[1]. The heart of the algorithm is the *risk-seeking policy gradient* training loss function, which optimizes for finding *the best possible* token sequence, as opposed to the regular policy gradient algorithm that optimizes for average sampling performance. DSO has been applied in a broad range of applications, but has been hampered by the inability to natively handle multiple objectives. To better demonstrate the value of multi-objective symbolic optimization, we add a simple adaptation of single-objective DSO as a baseline to our algorithm.

3.2 Priority Queue Training

Priority Queue Training (PQT), is an algorithm used in the training of RNNs. PQT generates solutions by sampling from the RNN, the best solutions are scored, and the top K solutions are then added to a priority queue. The RNN is then trained on the samples in this queue, a new sample is drawn, scored, the queue is updated, and this process is iterated until convergence. PQT has been used previously for program synthesis [2] and molecular optimization [3]. For the purposes of this study, we used PQT as a baselines during experimentation. For reference, we include a description of the algorithm below. Briefly, a batch of samples, $\mathcal{T}$, is generated by the current policy, Γ_θ. The reward is then calculated for each sequence. The top K performing samples are then filtered, $\mathcal{T}_\mathcal{K} \subset \{\mathcal{T}_{0:N}\} : \mathcal{R}(\tau_K) \geq \mathcal{R}(\tau_{i \in n}); \forall n \geq K$, where i is the rank of a given sequence, and $\mathcal{T}_{0:N}$ represents all of the sequences sampled thus far. Finally, the top K samples are added to the queue, and the RNN weights are updated via the following objective function:

$$J_{PQT}(\theta; k) = \frac{1}{k} \sum_{i=1}^{k} \log p(\mathcal{T}_\mathcal{K} | \theta), \quad (2)$$

where k is the priority queue size, and $\mathcal{T}_\mathcal{K}$ is the top K samples explored thus far.

3.3 Multi-Objective Optimization

Multi-objective (MO) optimization involves the simultaneous optimization of multiple objectives. In many cases, the objectives may present conflicting reward signals resulting in the need to evaluate trade-offs between each objective. In this work, we follow the utility-based approach [34] and assume that for any decision maker there exists some utility function, u, that represents their preferences over objectives. However, the decision maker may not know

[1] https://cavalab.org/srbench/competition-2022/

their utility function during learning. As a result, optimizing with respect to a utility function is not possible. In the taxonomy of multi-objective decision making [14], we are operating in the unknown utility function scenario. An efficient way to tackle problems in this scenario is to compute the Pareto Front (PF). The PF is a set of solutions that are optimal for all monotonically increasing utility functions where each solution in the set is Pareto non-dominated. The Pareto dominance relation $\boldsymbol{\mathcal{R}}^d(\tau') \succ_p \boldsymbol{\mathcal{R}}^d(\tau)$ can be defined as:

$$\begin{aligned} &\boldsymbol{\mathcal{R}}^d(\tau') \succ_p \boldsymbol{\mathcal{R}}^d(\tau) \iff \\ &(\forall d : \boldsymbol{\mathcal{R}}^d(\tau') \geq \boldsymbol{\mathcal{R}}^d(\tau)) \wedge (\exists d : \boldsymbol{\mathcal{R}}^d(\tau') > \boldsymbol{\mathcal{R}}^d(\tau)) \end{aligned} \tag{3}$$

Using the relation outlined in Eqn. 3, the PF can then be determined using:

$$PF(\Pi) = \{\tau \in \Pi \mid \nexists\, \tau' \in \Pi : \boldsymbol{\mathcal{R}}^d(\tau') \succ_p \boldsymbol{\mathcal{R}}^d(\tau)\}, \tag{4}$$

where Π is the set of all possible solutions, $\succ_p$ defines the Pareto dominance, and $\boldsymbol{\mathcal{R}}^d(\tau)$ is the vectorial reward.

By utilizing the utility-based approach, a decision maker can select a solution from the PF once their preferences over the objectives become known after learning.

To measure the quality of the set of solutions found for a given MO problem, we define the following multi-objective metrics. The hypervolume metric calculates the volume of a computed solution set with respect to a predefined reference point, V_{ref}. The reference point V_{ref} must be carefully chosen *a priori* given its selection can impact the final volume calculation. The hypervolume correlates with the spread of a set of undominated solutions over the multi-objective solution space. Given a set of points, or a PF, and a reference point, $V_{ref} \in \mathbb{R}^d$, the hypervolume can be defined as:

$$HV(\mathcal{T}^{PF}, V_{ref}) = \bigcup_{\tau \in \mathcal{T}^{PF}} Volume(V_{ref}, \mathcal{T}^{PF}) \tag{5}$$

However, hypervolume does not evaluate the sparsity of the PF. As shown by Xu et al. [48] two similar PFs can give very similar hypervolume score, but the sparsity of solutions on a given PF can vary widely. Having a dense PF, ensures that sufficient solutions are captured from which the user can select at decision time. As a result we aim to compute a dense PF while simultaneously maximizing hypervolume.

To capture the density of the computed PF, we use the sparsity metric proposed by Xu et al. [48], which is defined as follows:

$$Sp(PF) = \frac{1}{|PF| - 1} \sum_{j=1}^{d} \sum_{i=1}^{|PF|-1} (\tilde{PF}_j(i) - \tilde{PF}_j(i+1))^2 \tag{6}$$

Where PF is the empirically recovered set of Pareto-efficient solutions for d objectives, and $\tilde{PF}_j(i)$ is the i^{th} value in a sorted list of rewards for the j^{th} objective. In words, the sparsity metric gives us an average density of the PF. The metric is the average distance between solutions in the relevant multi-objective space.

By utilizing both hypervolume and sparsity it is possible to measure the breadth of the computed Pareto front (maximal hypervolume) while simultaneously providing us with a measure of density (minimal sparsity) of solutions on the Pareto front.

4 PROBLEM FORMULATION

In this work, we consider Multi-Objective Symbolic Optimization (MOSO), which involves finding a set of optimal solutions which consist of a discrete and symbolic sequence of tokens that maximize a number of scoring functions corresponding to the objectives of a given problem domain. Similarly to symbolic optimization, MOSO constructs sequences from a library of tokens, which represent a potential solution for the problem under consideration. In contrast to SO, MOSO utilizes multiple scoring functions (reward signals). Each sequence is evaluated using each scoring function and a vector score is used to represent the sequence, with respect to the scoring functions: $\boldsymbol{\mathcal{R}} : \boldsymbol{\tau} \rightarrow \mathbb{R}^d$, where d is the number of objectives. All combinations of tokens which result in a valid sequence can then be scored according to their fitness via each objective's reward function $\boldsymbol{\mathcal{R}}(\tau)$.

Given that we now have multiple reward functions, the *best* token sequence cannot be trivially defined given that the sequence can be the best for only some of the objectives, but not for all of them at the same time. For this reason, in MOSO we aim to find a set of optimal solutions. In our case, the solution of MOSO problems is the PF of sequences (see Section 3.3).

Therefore, we define the solution to a MOSO problem as follows:

$$PF_{n \in \mathbb{N}, \tau}[\mathcal{R}^1(\tau), \ldots, \mathcal{R}^d(\tau)], \tag{7}$$

with $\tau = \langle \tau_1, \ldots, \tau_n \rangle$, $\tau_i \in \mathcal{L}$, and where PF is the Pareto front defined on the d objectives. In Section 5, we present our novel algorithm to learn an empirical PF in MOSO settings.

5 PARETO FRONT TRAINING

It is often challenging for a human decision maker to accurately specify their preferences over the objectives of a given problem [34]. To overcome this challenge, it is preferred to compute a set of solutions that represent all possible utility functions, also known as the Pareto front (PF) [14]. After learning, a human decision maker can then select a solution from the computed Pareto front that best reflects their preferences. This process is also robust to changes in a decision maker's preferences over time, given the PF remains static [44]. Taking this into consideration, our algorithm aims to compute the PF of token sequences. We present a novel algorithm named Pareto Front Training (PFT). Our method is *anytime*, which means that the learning process can be stopped at any time, and the algorithm will return a set of non-dominated solutions found so far.

PFT utilizes a Recurrent Neural Network (RNN), whereby a solution is generated by sampling from the RNN one token at a time. Tokens are sampled autoregressively, meaning each token is conditioned on the previously sampled tokens. At every sampling step, the RNN defines a categorical distribution $p(\tau|\theta)$ (θ is the RNN weights) over all tokens in the library, and one of them is sampled according to the following distribution:

$$p(\tau_i|\tau_{i-1};\theta) = \text{softmax}(\psi^i_{\mathcal{L}(\tau_i)}) \tag{8}$$

Here, ψ_i represents the RNN outputs, normalized through a softmax layer. Therefore, the likelihood of sampling a given solution, $p(\tau|\theta)$,

is the product of the likelihood of the constituent tokens:

$$p(\tau|\theta) = \prod_{i=1}^{|\tau|} p(\tau_i|\tau_{i-1};\theta) = \prod_{i=1}^{|\tau|} \psi^{(i)}_{\mathcal{L}(\tau_i)} \tag{9}$$

PFT samples a batch of N solutions, $\mathcal{T}$, at each iteration, k, where K is the total number of iterations. To compute the PF, PFT maintains a buffer of Pareto non-dominated solutions, $\mathcal{T}^{PF}$, where each solution, τ is a solution that has been sampled from the RNN. After a batch of solutions has been sampled from the RNN, $\mathcal{T}^{PF}$ is updated by keeping only Pareto-dominant samples.

Naturally, $\mathcal{T}_{PF}$, contains all Pareto-optimal solutions that have been explored over all sampling iterations at any time.

$\mathcal{T}_{PF}$ is used to update the RNN during training. Specifically PFT trains on the log-likelihood loss function computed on *all* Pareto-efficient solutions in the computed Pareto-efficient set. The objective function for PFT is defined as follows:

$$J_{PFT}(\theta) = \frac{w}{|\mathcal{T}^{PF}|} \sum_{i=1}^{|\mathcal{T}^{PF}|} \log p(\mathcal{T}_i^{PF}|\,\theta), \tag{10}$$

where w is a hyperparameter controlling the weight of the loss function, and $\mathcal{T}_i^{PF}$ represents the i^{th} solution in the Pareto-efficient set of solutions. Rather than randomly mutating tokens as a genetic algorithm might, the RNN learns how to explore in the proximity of the Pareto front. The intuition behind doing this is that PFT prioritizes exploring in the neighborhood of currently Pareto-optimal samples, rather than haphazardly arriving at these solutions. A key benefit of this approach lies in the lack of domain-specific tuning required. Whereas a GA will typically require the user to explore a variety of parameters (e.g., crossover frequency, population size, selection criteria) for their problem at hand, PFT needs little adaptation to new areas of study.

To ensure PFT sufficiently explores the search space, an entropy regularization term is added to the loss function defined in Eqn. 10. The entropy regularization term increases exploration by ensuring the categorical distribution over tokens produced by the RNN is not dominated by any one token.

Algorithm 1 Pareto Front Training (PFT)

Require: Γ_θ: Policy network parameterized by θ;
N: Size of the batch;
1: initiate network parameters θ
2: **while** termination condition not achieved **do**
3: $\mathcal{T} \leftarrow \Gamma_\theta(N)$ ▷ Generate samples
4: **for** $\forall \tau \in \mathcal{T}$ **do**
5: $\tau.\mathbf{r} \leftarrow \mathcal{R}(\tau)$ ▷ Score sequences with vector reward
6: **end for**
7: $\mathcal{T}^{PF} = PF^{prune}(\mathcal{T} \cup \mathcal{T}^{PF})$ ▷ Remove dominated samples
8: $\theta \leftarrow LEARN(\theta, \mathcal{T}^{PF})$ ▷ Update Policy Network
9: **end while**
10: **return** $\mathcal{T}^{PF}$

Algorithm 1 outlines the PFT learning process. PFT takes a policy network parameterized by θ (in this case an RNN), and a batch size N as input. At each iteration, PFT samples from the parameterized policy to generate a set of K samples, $\mathcal{T}$. The algorithm computes a PF by taking the union of the current batch of samples and the current approximation of the PF. Using PF^{prune}, PFT updates $\mathcal{T}^{PF}$ by removing the Pareto-dominated solutions. Finally, PFT utilizes the *LEARN* step in Algorithm 1 Line 8, which updates the policy network using the loss function defined in Eqn. 10. This process repeats until a fixed number of iterations has been executed, or a stopping condition is met.

6 EMPIRICAL EVALUATION

Here we examine the performance of PFT in the challenging MO optimization domain of antibody optimization.

Antibodies serve as one of the primary defensive measures our immune systems have in response to a prolonged infection. They function by binding to the surface of a pathogen protein (antigen) and either (1) physically blocking the antigen from functioning; or (2) marking the pathogen for destruction by another immune cell. The field of antibody design and engineering has evolved drastically in the past few decades, resulting in over a hundred FDA approved antibody therapeutics [7, 46]. These therapies currently represent some of the most effective treatment options for diseases such as cancer, arthritis, and SARS-CoV-2 amongst others [9, 11, 12, 18, 27]. Regardless of the disease one is targeting, optimizing antibodies requires (at least) three major considerations: (1) tight, irreversible binding to the target antigen; (2) high-stability *in vivo*; and (3) low-to-negligible levels of autoimmunity (targeting the patient, rather than the pathogen; i.e., high-confidence in the safety of the antibody) [25]. In this context, we aim at leveraging simulations to computationally solve the antibody optimization problem.

6.1 Antibody Optimization Problem Modeling

Although *de novo* antibody design is possible [23, 30], engineering efforts are typically focused on mutating naturally-evolved antibodies. By starting from a known antibody sequence, there is more confidence the resulting mutant will bind to its target and have fewer side-effects due to off-target binding. For these reasons, in this study we elected not to design the antibody from scratch. Rather, the optimization process begins with a parental antibody strain. Mutations are then introduced into this sequence up to a maximum of five point mutations. Therefore we model this problem as SO as follows:

The library of tokens, $\mathcal{L}$, was defined to be the 20 naturally occurring amino acids (except Cysteine (C) and Proline (P)), for an average of 17 potential mutations at each point in the sequence, τ. To computationally compute our three objectives in practice, we use:

- **Binding** objective: Rosetta simulations to calculate the binding affinity of a given Ab sequence [5];
- **Stability** objective: Free-energy perturbation (FEP) to calculate a stability metric [20]; and
- **Safety** objective: The AbBERT [45] language model to estimate the potential for autoimmunity (using humanness as a proxy).

The reward, $\mathcal{R}(\tau)$, takes the form $\langle r_{binding}, r_{stability}, r_{humanness} \rangle$. Each of these calculations are quite computationally expensive, making experimentation cumbersome and time-consuming. To increase the efficiency of our experiments, we assumed the mutation effects

to be additive. That is, when calculating the effects of n mutations, $M_1, \ldots, M_n$, we approximate this as $\Delta\Delta G_{M_1,\ldots,M_n} = \sum_{i=0}^{n} \Delta\Delta G_i$, which has been shown to be a good approximation for antibody mutations in the literature [17].

In order to validate the performance of our PFT algorithm, we compare the quality of its suggested antibodies against several benchmarks:

- **PQT**: PQT is the single-objective version of our algorithm. We have added PQT as a baseline to validate if PFT can indeed outperform single-objective algorithms.
- **DSO**: DSO is the state-of-the-art algorithm in SO.
- **NSGA-II**: As a representative of solving this problem with a multi-objective genetic algorithm, we have included NSGA-II [8] as a baseline.

For both single-objective baselines (PQT and DSO), we have configured them with ten different utility functions. It is infeasible to sample from a distribution over all possible utility functions, given a parametric form of such a distribution may not exist. However, it is possible to sample from a subset of utility functions that are linear. To sample from a distribution over linear weights, a Dirichlet distribution can be utilized. We select a number of linear weights that give good coverage over the space of all possible weights parameterized by a Dirichlet distribution. Each of those utility functions, u, take the form:

$$u(\mathcal{R}(\tau)) = w_1 * r_{binding} + w_2 * r_{stability} + w_3 * r_{humanness}, \quad (11)$$

where the real numbers $w_i \geq 0$ and $\sum_i^n w_i = 1$ (convex combination).

The weights of each utility function were selected to provide a good coverage of different preferences, such as where each component reward was (1) considered in isolation ($w_i = 1$); (2) a dominant contributor to the reward ($w_i = 0.75$ and $w_i = 0.5$); and (3) represented equally ($w_i = 0.33$). A list of all functions we considered is shown in Table 1.

For all algorithms, we keep the empirical Pareto front of the best samples evaluated by the algorithm so far (regardless of whether the algorithm discards those samples or not). This optimal set is used to compute the hypervolume (higher is better) and sparsity (lower is better) statistics for all algorithms. While measuring the learning rate may suffice for the single-objective algorithms, as the domain we are using to test the algorithm is inherently multi-objective the use of hypervolume and sparsity in our analysis is paramount to adequately evaluating the performance of our algorithm.

The experiments shown in the following were performed using 100 replicates for each algorithm, with a batch size of 1000. Each simulation was run for 750 iterations, i.e., a total sample size of 750,000 sequences per replicate.

6.2 Experimental Results

Given the high number of possible configurations for both DSO and PQT baselines, we present the analysis of the experimental results in parts. Sections 6.2.1 and 6.2.2 describe our efforts to find the best configurations for PQT and DSO. With the top DSO and PQT configurations chosen, we carry out a final evaluation between them and the explicit multi-objective algorithms (PFT and NSGA-II) in Section 6.2.3. The single-objective algorithms are named according to the first floating point of their utility function. For example, $DSO_{0.7,0.1,0.1}$ refers to a linear utility function with weights as follows: $w_1 = 0.75, w_2 = 0.125, w_3 = 0.125$. See Table 1 for all linear utility functions used during experimentation.

6.2.1 PQT Evaluation. Figure 1 (top and bottom, respectively) depicts the hypervolume and sparsity results for all PQT configurations described above during the course of training, while Table 2 shows the final results for each configuration.

Although not all differences across configurations are statistically significant, the results for this algorithm confirm the reasonable intuition that weighting multiple objectives recovers a better PF (higher hypervolume) than optimizing uniquely for a single objective (all three "single-objective" configurations perform significantly worse than the top configuration, which considers all objectives with equal weights). For the sparsity metric, primarily a tie-breaking measure, results are not clearly distinguishable across configurations given the relatively high standard deviation. Therefore, our decisions are made for this comparison based solely on the hypervolume.

Analyzing those results, we observed the trend of the hypervolume consistently increasing as the dominance of single objectives is decreased. While the worst average performance is achieved by the single-objective configurations, the performance increases as the weights are more equally spread, reaching the best average performance for configuration $\text{PQT}_{0.3,0.3,0.3}$. We hypothesize that this observed trend is due to the way the RNN is trained. By weighting each objective equally, the top k samples used for training likely reflect those which are able to maximize each objective simultaneously, thus yielding a large hypervolume.

By comparing the performance of the single-objective configurations, we also observe that obtaining large values of $r_{humanness}$ can arise independently of $r_{binding}$ or $r_{stability}$. The opposite, however, may not be true when looking at $\text{PQT}_{1,0,0}$ or $\text{PQT}_{0,1,0}$. That is, there are likely many human antibodies which are not able to bind to a given target (high $r_{humanness}$; low $r_{binding}$), but those which are capable of binding (high $r_{binding}$) likely still resemble a non-zero level of humanness.

Given we have to choose a configuration to proceed to the next experiments, we selected $\text{PQT}_{0.3,0.3,0.3}$ although the results for this

$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$
1	0	0
0	1	0
0	0	1
0.75	0.125	0.125
0.125	0.75	0.125
0.125	0.125	0.75
0.5	0.25	0.25
0.25	0.5	0.25
0.25	0.25	0.5
0.33	0.33	0.33

Table 1: List of all the weights explored for utility function baselines.

configuration are nearly indistinguishable from $PQT_{0.5,0.2,0.2}$, as it has a slightly better average performance.

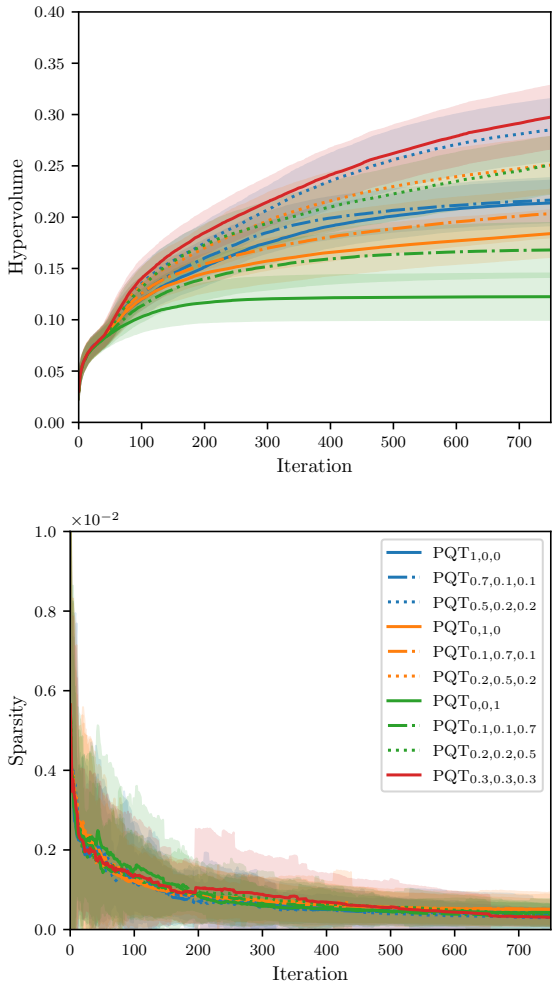

Figure 1: Average hypervolume (top) and sparsity (bottom) across 100 replicates with standard deviation (shaded). Each line represents one configuration of weights in Table 1 with the PQT algorithm.

6.2.2 DSO Evaluation. Figure 2 and Table 3 show the performance for all DSO configurations. The results for DSO are similar to those for PQT in many aspects, demonstrating that particular weights for the objectives leads to similar outcomes, regardless of the optimization algorithm used. The lowest performing utility function ($DSO_{0,0,1}$) again places all of the weight on the humanness score, reinforcing that the $r_{humanness}$ objective seems not very correlated to the other objectives. Likewise in the previous experiment, $DSO_{0.3,0.3,0.3}$ is the best-performing utility function.

Interestingly, DSO reaches convergence much faster (and at a lower performance) than PQT, plateauing very quickly to a lower hypervolume. Our hypothesis for this low performance displayed by

Configuration	Hypervolume (Avg +/- Std)	Sparsity $[10^{-2}]$ (Avg +/- Std)
$PQT_{1,0,0}$	0.21 +/- 0.02	0.039 +/- 0.034
$PQT_{0.7,0.1,0.1}$	0.22 +/- 0.02	0.037 +/- 0.026
$PQT_{0.5,0.2,0.2}$	0.29 +/- 0.03	0.030 +/- 0.026
$PQT_{0,1,0}$	0.18 +/- 0.02	0.051 +/- 0.037
$PQT_{0.1,0.7,0.1}$	0.20 +/- 0.03	0.050 +/- 0.042
$PQT_{0.2,0.5,0.2}$	0.25 +/- 0.03	0.042 +/- 0031
$PQT_{0,0,1}$	0.12 +/- 0.02	0.043 +/- 0.034
$PQT_{0.1,0.1,0.7}$	0.17 +/- 0.03	0.033 +/- 0.030
$PQT_{0.2,0.2,0.5}$	0.25 +/- 0.03	0.036 +/- 0.026
$PQT_{0.3,0.3,0.3}$	0.30 +/- 0.03	0.031 +/- 0.021

Table 2: Comparison across PQT configurations, results refer to 100 experiment repetitions.

Configuration	Hypervolume (Avg +/- Std)	Sparsity $[10^{-2}]$ (Avg +/- Std)
$DSO_{1,0,0}$	0.18 +/- 0.03	0.032 +/- 0.015
$DSO_{0.7,0.1,0.1}$	0.18 +/- 0.04	0.031 +/- 0.015
$DSO_{0.5,0.2,0.2}$	0.12 +/- 0.03	0.030 +/- 0.017
$DSO_{0,1,0}$	0.15 +/- 0.02	0.044 +/- 0.034
$DSO_{0.1,0.7,0.1}$	0.17 +/- 0.02	0.041 +/- 0.027
$DSO_{0.2,0.5,0.2}$	0.18 +/- 0.02	0.029 +/- 0.021
$DSO_{0,0,1}$	0.09 +/- 0.01	0.040 +/- 0.020
$DSO_{0.1,0.1,0.7}$	0.09 +/- 0.01	0.029 +/- 0.011
$DSO_{0.2,0.2,0.5}$	0.10 +/- 0.01	0.023 +/- 0.012
$DSO_{0.3,0.3,0.3}$	0.18 +/- 0.03	0.023 +/- 0.010

Table 3: Comparison across DSO configurations, results refer to 100 experiment repetitions. Italics represent the highest performing configurations in terms of hypervolume.

DSO is that it is caused by the very characteristic that made DSO the top performer in single-objective SO: the risk-seeking nature of the policy gradient. While discarding lower-performers is more trivial in a single-objective problem, the multiple objectives interact in a non-trivial way, and the risk-seeking objective prematurely discards samples thought to be of lower quality, reducing the diversity of solutions explored.

Additionally, we did not observe the same trend of reducing the dominance of a single objective yielding increases in the hypervolume for all objectives. Specifically, when $r_{binding}$ is the dominant contributor to the reward signal we see the opposite trend take hold. While $DSO_{1,0,0}$ and $DSO_{0.7,0.1,0.1}$ generate solution sets with approximately equivalent hypervolumes, $DSO_{0.5,0.2,0.2}$ only achieves approximately 75% of their hypervolume, on average.

In conclusion, we elected the $DSO_{0.3,0.3,0.3}$ configuration to be added to the next experiments, as it was the top configuration for DSO in terms of hypervolume, while simultaneously achieving a relatively lower sparsity score than the others.

6.2.3 Final Multi-objective Evaluation. We now compare the performance of our algorithm, PFT, with the performance from the

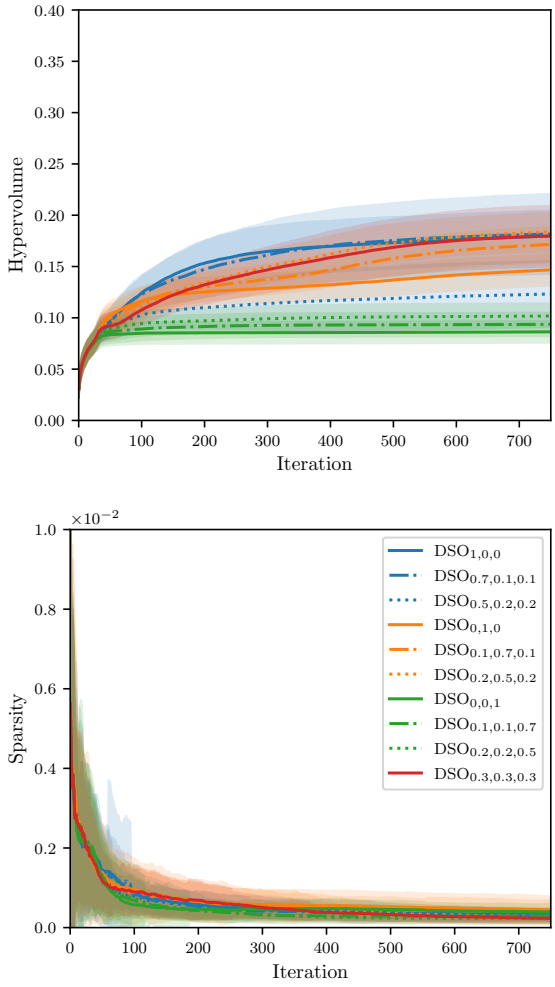

Figure 2: Average hypervolume (top) and sparsity (bottom) across 100 replicates with standard deviation (shaded). Each line represents one configuration of weights in Table 1 with the DSO algorithm.

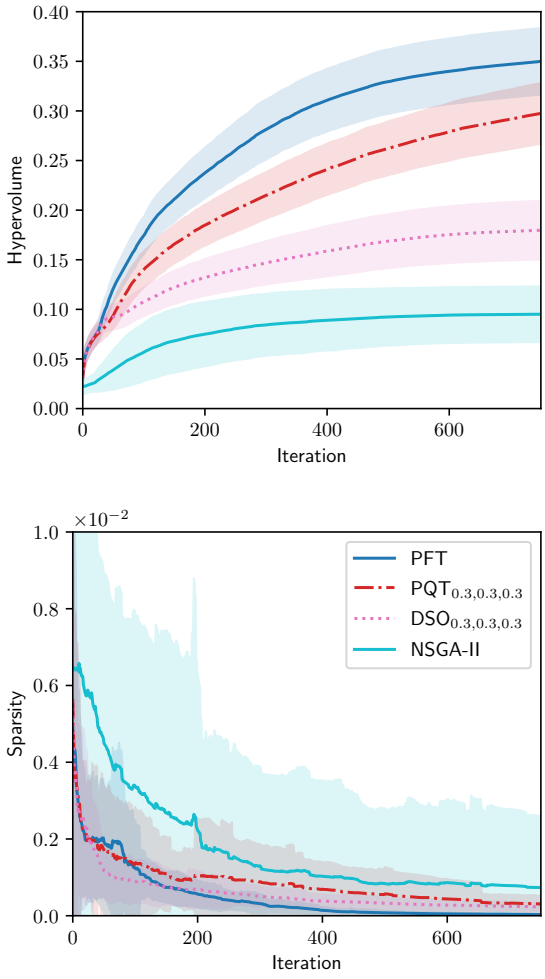

Figure 3: Average hypervolume (top) and sparsity (bottom) across 100 replicates with standard deviation (shaded).

top DSO and PQT configurations, as well as NSGA-II. Figure 3 and Table 4 show the results of this last experiment.

Configuration	Hypervolume (Avg +/- Std)	Sparsity [10^{-2}] (Avg +/- Std)
$PQT_{0.3,0.3,0.3}$	0.30 +/- 0.03	0.031 +/- 0.021
$DSO_{0.3,0.3,0.3}$	0.18 +/- 0.03	0.023 +/- 0.010
NSGA-II	0.10 +/- 0.03	0.073 +/- 0.187
PFT (Ours)	**0.35 +/- 0.03**	**0.003 +/- 0.002**

Table 4: Performance comparison for the winner DSO and PQT configurations against the PFT and NSGA-II multi-objective algorithms. Results refer to 100 experiment repetitions.

Very early in the training process (the first 50 iterations), PFT performs similarly in the hypervolume metric to both $PQT_{0.3,0.3,0.3}$ and $DSO_{0.3,0.3,0.3}$ while greatly outperforming NSGA-II. After this initial period, PFT quickly begins to outperform all three baselines. By the end of the training, not only does PFT achieve a better average hypervolume, but also obtains an average sparsity score almost a full order of magnitude better than the closest scalar configuration ($DSO_{0.3,0.3,0.3}$) and represents a 25-fold improvement over the native multi-objective baseline, NSGA-II.

Considering both metrics together, PFT significantly outperforms all baselines by a good margin. Those metrics reflect that the PF recovered by PFT is not only covering more of the solution space, but is increasing the density of the solution set, which is desired for multi-objective algorithms.

Not only does PFT outperform all algorithms in the experimental evaluation, PFT also does not require a utility function (which we dedicated an entire experiment to, just to find the most appropriate

function for DSO and PQT). Without the bias of requiring a utility function, the optimization scheme is able to not just arrive at a better hypervolume, but also avoids the process of tuning weights altogether.

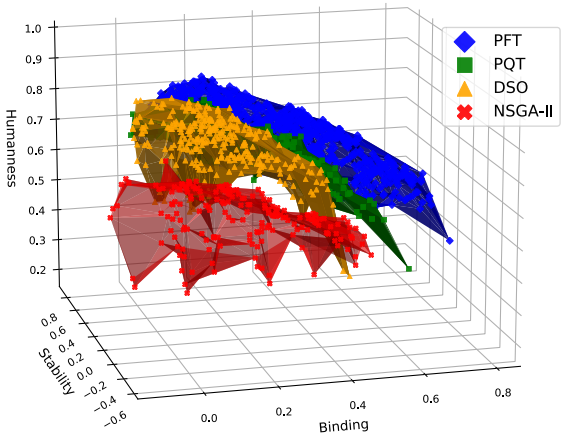

Figure 4: Pareto Fronts for all algorithms (top) showing PFT (blue) enclosing all others (PQT, green; DSO, orange; NSGA-II, red)

6.2.4 Examination of Sample Pareto Fronts. We next sought to examine samples of the Pareto surfaces that were obtained from each algorithm (Figure 4). The visual inspection of the PFs allows one to glean potentially useful information from the best sequences found by each algorithm, which complements the numerical results on hypervolume and sparsity. Although those metrics are useful for comparing algorithms, it is hard to infer the space covered by the PFs through only looking at the numbers. Further, by working with the individual PFs, we hope to provide a more robust intuition for how the hypervolume and sparsity metrics correspond to the relevant objective space.

The sample PFs provide support for the findings outlined previously for the average hypervolume and sparsity data. Namely, the PF generated by PFT encloses the others, corresponding to the larger hypervolume seen in the average across all replicates (Figure 3). Not only is the space occupied by the PFT samples dominating the others, but additionally it outperforms the other algorithms in the critical binding task (without the ability to bind its target, the other objectives are irrelevant). One can also see in Figure 4 that PFT achieves this while still sampling a wide range of trade offs in stability and humanness. By generating a dense, diverse set of solutions a user is able to weigh any benefits or drawbacks, and ultimately make the decision of what sequences to test experimentally.

Further, we can begin to explore where PFT was able to outperform our baselines in the antibody engineering domain. Looking at the samples from DSO (orange), PQT (green), and PFT (blue), they each uncovered similar regions of the objective space when maximizing the humanness objective, but differed in their ability to maintain this reward in combination with the others for the PFs shown. That is, PFT maintains a high degree of humanness *and* a high level of binding, whereas the others do not. We should also note the lack of success from NSGA-II (Figure 4; red) in this sample. While the algorithm does balance the three objectives, it does so at a much lower level compared to the other configurations.

The data presented here is a single sample from each algorithm, which is especially important to consider when looking at the results of NSGA-II, as this algorithm had the greatest standard deviation in both numerical metrics (Table 4). Nevertheless, the analysis presented here conveys the importance of interrogating the outputs in the relevant objective space in addition to any higher-level metrics. These observations require additional significance testing, and warrant further exploration with other pertinent domains, but nonetheless seem to be interesting properties worth investigating.

7 CONCLUSION AND FURTHER WORK

Although Symbolic Optimization (SO) algorithms have successfully solved challenging tasks in the past years, state-of-the-art SO methods can only handle a single objective, which makes it challenging to solve applications that are more naturally described by multiple objectives. In this work, we present a novel algorithm, Pareto Front Training (PFT), for applications in multi-objective symbolic optimization. We applied PFT, as well as several other baselines, to an antibody design task. PFT significantly outperformed all tested baselines, in addition to not requiring the parameterization of a utility function.

As future work, we aim to improve the performance of PFT by altering the data set used by PFT to update the policy network. For example, Reymond et al. [31] found, in reinforcement learning settings, training explicitly on the Pareto front to not be sufficient. Instead, Reymond et al. train on a number of sub-optimal solutions that are close to the Pareto front based on crowding distance and other metrics. This enhancement led to a performance boost. Approaches like those outlined by Reymond et al. [31] are an interesting starting point for future work. Furthermore, exploring recent advances in multi-fidelity SO [40], to assess how multi-fidelity algorithms would be impacted by multi-objective optimization would be another interesting avenue for future work.

ACKNOWLEDGMENTS

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 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-855513.

REFERENCES

[1] Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. 2019. Dynamic weights in multi-objective deep reinforcement learning. In *International conference on machine learning*. PMLR, 11–20.

[2] Daniel A Abolafia, Mohammad Norouzi, Jonathan Shen, Rui Zhao, and Quoc V Le. 2018. Neural program synthesis with priority queue training. *arXiv preprint arXiv:1801.03526* (2018).

[3] Sungsoo Ahn, Junsu Kim, Hankook Lee, and Jinwoo Shin. [n. d.]. Guiding Deep Molecular Optimization with Genetic Exploration. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*.

[4] Lucas Nunes Alegre, Ana Bazzan, and Bruno C Da Silva. 2022. Optimistic linear support and successor features as a basis for optimal policy transfer. In *International Conference on Machine Learning*. PMLR, 394–413.

[5] Kyle A Barlow, Shane Ó Conchúir, Samuel Thompson, Pooja Suresh, James E Lucas, Markus Heinonen, and Tanja Kortemme. 2018. Flex ddG: Rosetta ensemble-based estimation of changes in protein–protein binding affinity upon mutation. *The Journal of Physical Chemistry B* 122, 21 (2018), 5389–5399.

[6] P. Barmpalexis, K. Kachrimanis, A. Tsakonas, and E. Georgarakis. 2011. Symbolic regression via genetic programming in the optimization of a controlled release pharmaceutical formulation. *Chemometrics and Intelligent Laboratory Systems* 107 (5 2011), 75–82. Issue 1. https://doi.org/10.1016/j.chemolab.2011.01.012

[7] Henry Hongrong Cai and Ayesha Pandit. 2021. Therapeutic monoclonal antibodies approved by FDA in 2020. *Clin. Res. Immunol* 4 (2021), 1–2.

[8] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.

[9] S Di Martino, A Rainone, A Troise, M Di Paolo, S Pugliese, S Zappavigna, A Grimaldi, and D Valente. 2015. Overview of FDA-approved anti cancer drugs used for targeted therapy. *WCRJ* 2, 3 (2015), e553.

[10] Matthias Ehrgott. 2005. *Multicriteria optimization*. Vol. 491. Springer Science & Business Media.

[11] Loretta Fala. 2016. Portrazza (Necitumumab), an IgG1 Monoclonal Antibody, FDA Approved for Advanced Squamous Non–Small-Cell Lung Cancer. *American health & drug benefits* 9, Spec Feature (2016), 119.

[12] Marco Falcone, Giusy Tiseo, Beatrice Valoriani, Chiara Barbieri, Sara Occhineri, Paola Mazzetti, Maria Linda Vatteroni, Lorenzo Roberto Suardi, Niccolò Riccardi, Mauro Pistello, et al. 2021. Efficacy of bamlanivimab/etesevimab and casirivimab/imdevimab in preventing progression to severe COVID-19 and role of variants of concern. *Infectious diseases and therapy* 10, 4 (2021), 2479–2488.

[13] Stephanie Forrest. 1996. Genetic algorithms. *ACM computing surveys (CSUR)* 28, 1 (1996), 77–80.

[14] Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. 2022. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems* 36, 1 (2022), 1–59.

[15] Conor F Hayes, Mathieu Reymond, Diederik M Roijers, Enda Howley, and Patrick Mannion. 2023. Monte Carlo tree search algorithms for risk-aware and multi-objective reinforcement learning. *Autonomous Agents and Multi-Agent Systems* 37, 2 (2023), 26.

[16] Conor F Hayes, Diederik M Roijers, Enda Howley, and Patrick Mannion. 2022. Decision-Theoretic Planning for the Expected Scalarised Returns. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 1621–1623.

[17] Sherlyn Jemimah and M. Michael Gromiha. 2018. Exploring additivity effects of double mutations on the binding affinity of protein-protein complexes. *Proteins: Structure, Function and Bioinformatics* 86 (5 2018), 536–547. Issue 5. https://doi.org/10.1002/prot.25472

[18] Disha Kesharwani, Rishi Paliwal, Trilochan Satapathy, and Swarnali Das Paul. 2019. Rheumatiod arthritis: An updated overview of latest therapy and drug delivery. *Journal of pharmacopuncture* 22, 4 (2019), 210.

[19] Yaroslav Kharkov, Oles Shtanko, Alireza Seif, Przemyslaw Bienias, Mathias Van Regemortel, Mohammad Hafezi, and Alexey V. Gorshkov. 2021. Discovering hydrodynamic equations of many-body quantum systems. (11 2021). http://arxiv.org/abs/2111.02385

[20] Seonghoon Kim, Hiraku Oshima, Han Zhang, Nathan R Kern, Suyong Re, Jumin Lee, Benoît Roux, Yuji Sugita, Wei Jiang, and Wonpil Im. 2020. CHARMM-GUI free energy calculator for absolute and relative ligand solvation and binding free energy simulations. *Journal of chemical theory and computation* 16, 11 (2020), 7207–7218.

[21] Mikel Landajuela, Chak Shing Lee, Jiachen Yang, Ruben Glatt, Claudio P Santiago, Ignacio Aravena, Terrell Mundhenk, Garrett Mulcahy, and Brenden K Petersen. 2022. A unified framework for deep symbolic regression. *Advances in Neural Information Processing Systems* 35 (2022), 33985–33998.

[22] Haochen Li, Fabian Waschkowski, Yaomin Zhao, and Richard D. Sandberg. 2023. Turbulence Model Development based on a Novel Method Combining Gene Expression Programming with an Artificial Neural Network. (1 2023). http://arxiv.org/abs/2301.07293

[23] Tong Li, Robert J Pantazes, and Costas D Maranas. 2014. OptMAVEn–a new framework for the de novo design of antibody variable region models targeting specific antigen epitopes. *PloS one* 9, 8 (2014), e105954.

[24] Yi Li, Aws Albarghouthi, Zachary Kincaid, Arie Gurfinkel, and Marsha Chechik. 2014. Symbolic optimization with SMT solvers. *Conference Record of the Annual ACM Symposium on Principles of Programming Languages*, 607–618. https://doi.org/10.1145/2535838.2535857

[25] Jennifer Maynard and George Georgiou. 2000. Antibody engineering. *Annual review of biomedical engineering* 2, 1 (2000), 339–376.

[26] Terrell Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio P Santiago, Brenden K Petersen, et al. 2021. Symbolic regression via deep reinforcement learning enhanced genetic programming seeding. *Advances in Neural Information Processing Systems* 34 (2021), 24912–24923.

[27] John O'Horo, Douglas W Challener, Ryan J Anderson, Richard F Arndt, Sara E Ausman, Scott T Hall, Alexander Heyliger, Brian D Kennedy, Perry W Sweeten, Ravindra Ganesh, et al. 2022. Rates of severe outcomes after bamlanivimab-etesevimab and casirivimab-imdevimab treatment of high-risk patients with mild to moderate coronavirus disease 2019. In *Mayo Clinic Proceedings*, Vol. 97. Elsevier, 943–950.

[28] Brenden K Petersen, Mikel Landajuela, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim. 2021. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *Proceeding of the International Conference on Learning Representations (ICLR)* (2021).

[29] Jacob F Pettit, Brenden K Petersen, Chase Cockrell, Dale B Larie, Felipe Leno Silva, Gary An, and Daniel M Faissol. 2021. Learning sparse symbolic policies for sepsis treatment. In *Interpretable ML in Healthcare Workshop at ICML*.

[30] Venkata Giridhar Poosarla, Tong Li, Boon Chong Goh, Klaus Schulten, Thomas K Wood, and Costas D Maranas. 2017. Computational de novo design of antibodies binding to a peptide with high affinity. *Biotechnology and bioengineering* 114, 6 (2017), 1331–1342.

[31] Mathieu Reymond, Eugenio Bargiacchi, and Ann Nowé. 2022. Pareto Conditioned Networks. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*. 1110–1118.

[32] Mathieu Reymond, Conor F Hayes, Denis Steckelmacher, Diederik M Roijers, and Ann Nowé. 2023. Actor-critic multi-objective reinforcement learning for non-linear utility functions. *Autonomous Agents and Multi-Agent Systems* 37, 2 (2023), 23.

[33] Diederik M Roijers, Denis Steckelmacher, and Ann Nowé. [n. d.]. Multi-objective reinforcement learning for the expected utility of the return.

[34] Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. 2013. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research* 48, 1 (2013), 67–113.

[35] Willem Röpke, Conor F Hayes, Patrick Mannion, Enda Howley, Ann Nowé, and Diederik M Roijers. 2023. Distributional Multi-Objective Decision Making. *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)* (2023).

[36] Jennifer J. Schnur and Nitesh V. Chawla. 2023. Information fusion via symbolic regression: A tutorial in the context of human health. , 326-335 pages. https://doi.org/10.1016/j.inffus.2022.11.030

[37] Felipe Leno da Silva, Ruben Glatt, Wencong Su, Van-Hai Bui, Fangyuan Chang, Shivam Chaturvedi, Mengqi Wang, Yi Lu Murphey, Can Huang, Lingxiao Xue, and Rong Zeng. 2023. AutoTG: Reinforcement Learning-Based Symbolic Optimization for AI-Assisted Power Converter Design. *IEEE Journal of Emerging and Selected Topics in Industrial Electronics* (2023), 1–10. https://doi.org/10.1109/JESTIE.2023.3303836

[38] Felipe Leno da Silva, Andre Goncalves, Sam Nguyen, Denis Vashchenko, Ruben Glatt, Thomas Desautels, Mikel Landajuela, Daniel Faissol, and Brenden Petersen. 2023. Language model-accelerated deep symbolic optimization. *Neural Computing and Applications* (2023), 1–17.

[39] Felipe Leno da Silva, Andre Goncalves, Sam Nguyen, Denis Vashchenko, Ruben Glatt, Thomas Desautels, Mikel Landajuela, Brenden Petersen, and Daniel Faissol. 2022. Leveraging Language Models to Efficiently Learn Symbolic Optimization Solutions. In *Adaptive and Learning Agents (ALA) Workshop at AAMAS*.

[40] Felipe Leno da Silva, Jiachen Yang, Mikel Landajuela, Andre Goncalves, Alexander Ladd, , Daniel Faissol, and Brenden Petersen. 2023. Toward Multi-Fidelity Reinforcement Learning for Symbolic Optimization. In *Adaptive and Learning Agents (ALA) Workshop at AAMAS*.

[41] Fangzheng Sun, Yang Liu, Jian-Xun Wang, and Hao Sun. 2022. Symbolic Physics Learner: Discovering governing equations via Monte Carlo tree search. (5 2022). http://arxiv.org/abs/2205.13134

[42] Wassim Tenachi, Rodrigo Ibata, and Foivos I. Diakogiannis. 2023. Deep symbolic regression for physics guided by units constraints: toward the automated discovery of physical laws. (3 2023). http://arxiv.org/abs/2303.03192

[43] Silviu-Marian Udrescu and Max Tegmark. 2020. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances* 6, 16 (2020), eaay2631.

[44] Peter Vamplew, Benjamin J Smith, Johan Källström, Gabriel Ramos, Roxana Rădulescu, Diederik M Roijers, Conor F Hayes, Fredrik Heintz, Patrick Mannion, Pieter JK Libin, et al. 2022. Scalar reward is not enough: A response to silver, singh, precup and sutton (2021). *Autonomous Agents and Multi-Agent Systems* 36, 2 (2022), 41.

[45] Denis Vashchenko, Sam Nguyen, Andre Goncalves, Felipe Leno da Silva, Brenden Petersen, Thomas Desautels, and Daniel Faissol. 2022. AbBERT: Learning Antibody Humanness via Masked Language Modeling. In *Workshop on Healthcare AI and Covid-19*.

[46] Shawn Shouye Wang, Yifei Yan, and Kin Ho. 2021. US FDA-approved therapeutic antibodies with high-concentration formulation: summaries and perspectives. *Antibody therapeutics* 4, 4 (2021), 262–272.

[47] Nai Long Wu, Xu Yang Wang, Tong Ge, Chao Wu, and Rui Yang. 2017. Parametric identification and structure searching for underwater vehicle model using symbolic regression. *Journal of Marine Science and Technology (Japan)* 22 (3 2017), 51–60. Issue 1. https://doi.org/10.1007/s00773-016-0396-8

[48] Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. 2020. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *International conference on machine learning*. PMLR, 10607–10616.