Training Greedy Policy for Proposal Batch Selection in Expensive Multi-Objective Combinatorial Optimization

Deokjae Lee [1 2] Hyun Oh Song [1 2] Kyunghyun Cho [3 4]

Abstract

Active learning is increasingly adopted for expensive multi-objective combinatorial optimization problems, but it involves a challenging subset selection problem, optimizing the batch acquisition score that quantifies the goodness of a batch for evaluation. Due to the excessively large search space of the subset selection problem, prior methods optimize the batch acquisition on the latent space, which has discrepancies with the actual space, or optimize individual acquisition scores without considering the dependencies among candidates in a batch instead of directly optimizing the batch acquisition. To manage the vast search space, a simple and effective approach is the greedy method, which decomposes the problem into smaller subproblems, yet it has difficulty in parallelization since each subproblem depends on the outcome from the previous ones. To this end, we introduce a novel greedy-style subset selection algorithm that optimizes batch acquisition directly on the combinatorial space by sequential greedy sampling from the *greedy policy*, specifically trained to address all greedy subproblems concurrently. Notably, our experiments on the red fluorescent proteins design task show that our proposed method achieves the baseline performance in 1.69$\times$ fewer queries, demonstrating its efficiency.

1. Introduction

In various practical design fields, including biological sequence design, molecular graph optimization, and chip design, challenges are typically posed as expensive multi-objective combinatorial optimization (MOCO) problems. These problems focus on identifying designs, represented as discrete objects like strings or graphs, that optimize multiple attributes, often requiring substantial resources for accurate assessment (Ehrgott, 2005; Gómez-Bombarelli et al., 2016; Stanton et al., 2022; Winter et al., 2019; Mirhoseini et al., 2021). Active learning frameworks, which iteratively propose a batch of candidates and learn from the attributes evaluated on those candidates, are increasingly employed in these fields due to their query efficiency, which is a critical component to handling expensive evaluation costs (Aggarwal et al., 2014; Jain et al., 2022; Gruver et al., 2023; Zhu et al., 2023; Agnesina et al., 2023). In active learning, each round entails an internal problem of selecting a proposal batch of candidates for querying, formulated by cardinality-constrained subset selection problem. This aims to identify the optimal batch $B \subset \mathcal{X}$ of size n that maximizes the batch acquisition function $a : 2^{\mathcal{X}} \rightarrow \mathbb{R}$, which quantifies the goodness of a batch considering interdependencies among candidates (González et al., 2015; Wang et al., 2016; Daulton et al., 2020). Unfortunately, the subset selection problem is challenging due to its prohibitively large search space of size $\mathcal{O}(|\mathcal{X}|^n)$, which increases exponentially as the batch size n increases, while the combinatorial space $\mathcal{X}$ itself often has large size in practical scenarios (Polishchuk et al., 2013).

A natural approach to efficiently solve a subset selection problem is a greedy algorithm that sequentially constructs a subset by adding the optimal candidate that maximizes marginal gain in the objective set function, breaking down the problem into a sequence of substantially smaller, manageable subproblems of size $\mathcal{O}(|\mathcal{X}|)$ (Nemhauser et al., 1978). Notably, the presence of monotone submodularity in prevalent batch acquisition functions such as JES, SM, EHVI, and NEHVI provides a theoretical performance guarantee for the greedy algorithm (Goundan & Schulz, 2007; Azimi et al., 2010). In this regard, active learning methods son continuous spaces already adopt greedy algorithms by solving each subproblem with first-order methods (Tu et al., 2022; Daulton et al., 2020; 2021). However, for MOCO problems, applying greedy algorithms is even more challenging due to their discrete nature, which prohibits the use of first-order solvers. Instead, previous works utilize latent space optimization (LSO) algorithms, which alternatively

[1]Seoul National University [2]Neural Processing Research Center [3]New York University [4]Genentech. Correspondence to: Kyunghyun Cho <kc119@nyu.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

optimize the batch acquisition in the continuous latent space, which has discrepancies with the batch acquisition in the actual space, and obtain the batch by decoding the optimized latent (Gómez-Bombarelli et al., 2016; Stanton et al., 2022), or construct a batch by sampling candidates of high individual acquisition scores $a(\{\mathbf{x}\})$s without considering the interdependencies among candidates explicitly (Jain et al., 2023).

In this work, we propose a greedy-style subset selection algorithm for expensive MOCO problems. One direct approach is sequentially applying any combinatorial optimization algorithm n times to solve n subproblems. However, this sequential construction strategy has a drawback because each subproblem requires the results from preceding subproblems, hindering the parallelization of the overall process. To address this, we propose a novel subset selection approach based on reinforcement learning (RL) that trains only a single *greedy policy*, a set-conditioned policy capable of addressing all subproblems concurrently, instead of sequentially training n distinct policies for n subproblems, thereby amortizing the burden of solving n subproblems. Our contributions can be summarized as follows:

- We propose a novel greedy-style subset selection algorithm that requires training of only a single greedy policy. Also, we suggest a novel training algorithm for obtaining the greedy policy, along with a justification for this approach.
- We extend the theoretical bounds of the approximated greedy algorithm to include both near-submodular functions and diversity functions, broadening its applicability.
- Our method consistently outperforms baseline methods, constructing the batch with a higher batch acquisition in various benchmarks for active learning inner loops. Significantly, our method attains the same Hypervolume indicator value as baseline methods but with $1.69\times$ fewer queries in the multi-round active learning benchmark on red fluorescent proteins (RFP).

2. Preliminaries

2.1. Expensive MOCO

In this work, we consider an expensive MOCO problem, which aims to maximize an m-dimensional expensive, black-box oracle function $\mathbf{f} : \mathcal{X} \to \mathbb{R}^m$ on the combinatorial space $\mathcal{X}$, *e.g.*, the space of amino-acid sequences (Tripp et al., 2020). This can be formulated as

$$\underset{\mathbf{x}\in\mathcal{X}}{\text{maximize}}\ \mathbf{f}(\mathbf{x}) := (f_0(\mathbf{x}), \dots, f_{m-1}(\mathbf{x})), \tag{1}$$

where we define the partial order between two vectors $\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}') \in \mathbb{R}^m$ by the pointwise order, *i.e.*, $\mathbf{f}(\mathbf{x}) \succeq \mathbf{f}(\mathbf{x}')$ if and only if $f_i(\mathbf{x}) \geq f_i(\mathbf{x}')$ for all $i \in [m] := \{0, \dots, m-1\}$ (Blyth, 2005). We say that a candidate $\mathbf{x}$ *dominates* another candidate $\mathbf{x}'$, written by $\mathbf{x} \succ \mathbf{x}'$, if $\mathbf{f}(\mathbf{x}) \succeq \mathbf{f}(\mathbf{x}')$ and $\mathbf{f}(\mathbf{x}) \neq \mathbf{f}(\mathbf{x}')$. Using the notion of dominance, we introduce the set of the optimal solutions of Equation (1), the *Pareto set*, and its images, the *Pareto frontier* (Konak et al., 2006).

Algorithm 1 Multi-Round Active Learning

```
Input: an oracle $\mathbf{f}$, a surrogate model $\tilde{\mathbf{f}}$, a batch size $n$,
the number of rounds $N_r$, and the initial dataset $\mathcal{D}_0$.
for $i = 0$ to $N_r - 1$ do
    Train a surrogate $\tilde{\mathbf{f}}$ using $\mathcal{D}_i$.
    Solve Equation (2) to select the batch $B$ (inner loop).
    Evaluate the batch $B$ with the oracle $\mathbf{f}$.
    Update the dataset $\mathcal{D}_{i+1} \leftarrow \mathcal{D}_i \cup \{(\mathbf{x}, \mathbf{f}(\mathbf{x}))\}_{\mathbf{x}\in B}$.
end for
Return NonDominatedSort$(\mathcal{D}_{N_r})$
```

Definition 2.1. (Pareto set and Pareto frontier) The Pareto set $\mathcal{P}^* \subset \mathcal{X}$ is the set of the optimal solutions that are not dominated by any other candidates in $\mathcal{X}$. Concretely, $\mathcal{P}^* := \{\mathbf{x} \in \mathcal{X} \mid \nexists \mathbf{x}' \in \mathcal{X} \text{ s.t. } \mathbf{x}' \succ \mathbf{x}\}$. Furthermore, the Pareto frontier is the images of the Pareto set under the oracle function $\mathbf{f}$, denoted as $\mathbf{f}(\mathcal{P}^*) \subset \mathbb{R}^m$.

For the non-trivial scenarios, the Pareto set $\mathcal{P}^*$ has more than one solution due to trade-offs among oracle components $f_0, \dots, f_{m-1}$ (Ehrgott, 2005). Since we consider the scenario that a given black-box oracle function is expensive, the goal is to find a good approximated Pareto set $\tilde{\mathcal{P}} \subseteq \mathcal{X}$ in a limited query budget. To assess the quality of an approximation set $\tilde{\mathcal{P}}$, a commonly used metric is the *Hypervolume indicator*, which measures the volume bounded by the reference point $\mathbf{r}_{\text{ref}} \in \mathbb{R}^m$ and the images of the approximation set $\mathbf{f}(\tilde{\mathcal{P}}) \subset \mathbb{R}^m$, written by $\mathbf{HV}(\mathbf{f}(\tilde{\mathcal{P}}); \mathbf{r}_{\text{ref}}) := \text{Vol}(\bigcup_{\mathbf{y}\in\mathbf{f}(\tilde{\mathcal{P}})}\{\mathbf{v} \in \mathbb{R}^m \mid \mathbf{r}_{\text{ref}} \preceq \mathbf{v} \preceq \mathbf{y}\})$ (Guerreiro et al., 2021). In this work, we consider an approximation set with a higher Hypervolume indicator value as a better approximation of the Pareto set.

2.2. Multi-Round Active Learning

Multi-round active learning is a framework widely adopted for optimizing an expensive oracle function in a query-efficient manner (Aggarwal et al., 2014). This framework involves repeatedly suggesting candidates and learning from the oracle's feedback on those candidates (Jain et al., 2022). Due to the significant time costs of oracle queries, a common practice is to select a *batch* of candidates for each round, enabling parallel evaluation by the oracle and thus improving overall efficiency (Daulton et al., 2021). The *batch Bayesian optimization* (BO) framework is a representative active learning approach equipped with a statistical surrogate model that estimates the oracle using the posterior

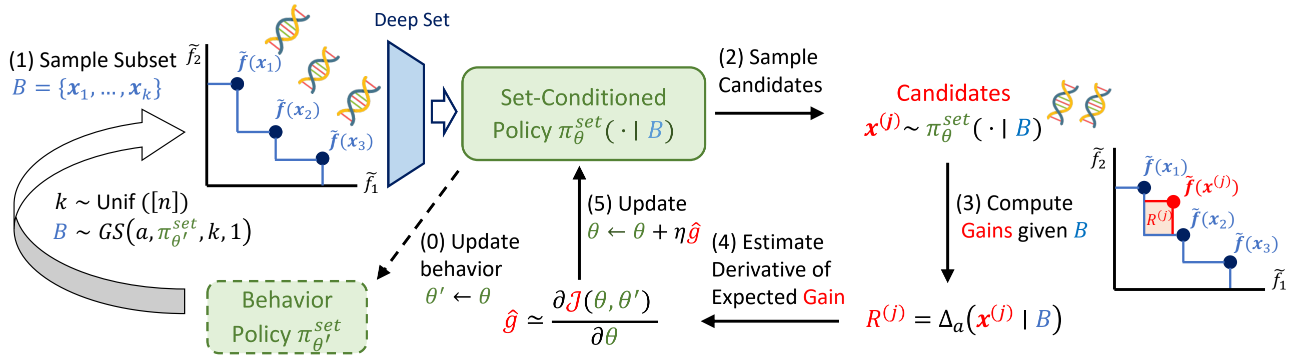

(1) Sample Subset
$B = \{x_1, \dots, x_k\}$
$\tilde{f}(x_1)$
$\tilde{f}(x_2)$
$\tilde{f}(x_3)$
$\tilde{f}_2$
$\tilde{f}_1$
$k \sim$ Unif $([n])$
$B \sim GS(a, \pi_{\theta'}^{set}, k, 1)$
Deep Set
Set-Conditioned Policy $\pi_\theta^{set}(\cdot \mid B)$
(2) Sample Candidates
Candidates
$x^{(j)} \sim \pi_\theta^{set}(\cdot \mid B)$
(0) Update behavior
$\theta' \leftarrow \theta$
Behavior Policy $\pi_{\theta'}^{set}$
(5) Update
$\theta \leftarrow \theta + \eta\hat{g}$
$\hat{g} \simeq \frac{\partial \mathcal{J}(\theta, \theta')}{\partial \theta}$
(4) Estimate Derivative of Expected Gain
(3) Compute Gains given B
$R^{(j)} = \Delta_a(x^{(j)} \mid B)$
$\tilde{f}(x^{(j)})$
$R^{(j)}$

Figure 1: The visualization of our learning method (Section 3.1). At a high level, a set-conditioned policy π_θ^{set} is trained to generate candidates that maximize marginal gain $\Delta_a(\cdot \mid B)$ when conditioned by B, where B is sampled by π_θ^{set} itself.

distribution given previous oracle evaluations (González et al., 2015). Algorithm 1 summarizes the overall active learning process. For each round, a cheaper surrogate model $\tilde{\mathbf{f}}(\cdot;\theta)$ is trained using data from previous steps to estimate the expensive oracle function $\mathbf{f}$. Subsequently, the *inner loop* chooses the proposal batch of n candidates for querying by solving the following cardinality-constrained subset selection problem:

$$\underset{B \subset \mathcal{X}}{\text{maximize}}\ a(B) \quad \text{subject to } |B| \le n, \tag{2}$$

where $a(\cdot;\tilde{f}) : 2^{\mathcal{X}} \to \mathbb{R}$ is a *batch acquisition function*, introduced in the subsequent section. After finishing N_r rounds, active learning returns non-dominated solutions among the evaluated dataset $\mathcal{D}_{N_r}$ as the approximation set to the Pareto set.

2.3. Batch acquisition functions for MOCO

Acquisition functions are designed to quantify the value of evaluating candidates throughout the active learning process, balancing the trade-offs between exploitation and exploration (O'Donoghue et al., 2017). Specifically, batch acquisition functions assess the value of evaluating a batch B, further considering the interdependencies within the batch (Wang et al., 2016; Tu et al., 2022; Yang et al., 2022; Azimi et al., 2010; Park et al., 2023). In this work, we focus on batch acquisition functions based on *Hypervolume improvement* (HVI), widely used in recent studies on expensive multi-objective optimization (Guerreiro et al., 2021; Konakovic Lukovic et al., 2020; Stanton et al., 2022; Jain et al., 2023; Lin et al., 2022). Given the evaluated solution set $\tilde{\mathcal{P}}$ and the reference point $\mathbf{r}_{\text{ref}}$, HVI when evaluating a batch B is defined as

$$\begin{aligned}&\mathbf{HVI}(B;\tilde{\mathbf{f}},\tilde{\mathcal{P}},\mathbf{r}_{\text{ref}})\\ &\quad := \mathbf{HV}(\tilde{\mathbf{f}}(B)\cup\tilde{\mathbf{f}}(\tilde{\mathcal{P}});\mathbf{r}_{\text{ref}}) - \mathbf{HV}(\tilde{\mathbf{f}}(\tilde{\mathcal{P}});\mathbf{r}_{\text{ref}}).\end{aligned}$$

HVI can be directly utilized as an acquisition function for a deterministic surrogate model. For a statistical surrogate model, variations of HVI such as EHVI, NEHVI, and UCB-HVI exist (Daulton et al., 2020; 2021; Emmerich et al., 2015). EHVI and NEHVI compute the expected values of HVI under the assumptions of noiseless and noisy observations, respectively. UCB-HVI utilizes upper confidence bound (UCB) defined as $\tilde{\mathbf{f}}_{\text{UCB}}(\mathbf{x};\beta) := \text{mean}(\tilde{\mathbf{f}}(\mathbf{x})) + \beta\ \text{std}(\tilde{\mathbf{f}}(\mathbf{x})) \in \mathbb{R}^m$ as a proxy vector of the oracle and computes $\mathbf{HVI}(B;\tilde{\mathbf{f}}_{\text{UCB}},\tilde{\mathcal{P}},\mathbf{r}_{\text{ref}})$.

2.4. Reinforcement Learning for Single-Objective Combinatorial Optimization

Combinatorial objects can often be constructed by sequential actions. Deep RL-based methods for single-objective combinatorial optimization, aimed at solving $\max_{\mathbf{x}\in\mathcal{X}} R(\mathbf{x})$, train a parameterized stochastic policy π_θ. This policy, which determines actions at each state of object construction, seeks to maximize the objective function R (Zoph & Le, 2017; Mirhoseini et al., 2021). In this context, the resulting state of a trajectory τ, sampled from a policy π_θ, corresponds to a combinatorial object $\mathbf{x} \in \mathcal{X}$, and the return of the trajectory is given by $R(\mathbf{x})$. For simplicity, we use the same notation for the trajectory and the resulting combinatorial object; therefore, for $\mathbf{x} \sim \pi_\theta$, $R(\mathbf{x})$ represents the objective value of the corresponding combinatorial object $\mathbf{x}$, and $\pi_\theta(\mathbf{x})$ represents the probability of the trajectory $\mathbf{x}$. Then, the given problem $\max_{\mathbf{x}\in\mathcal{X}} R(\mathbf{x})$ can be translated to the following optimization problem:

$$\underset{\theta\in\Theta}{\text{maximize}}\ \mathbb{E}_{\mathbf{x}\sim\pi_\theta}[R(\mathbf{x})].$$

In this work, we consider the most basic algorithm with the REINFORCE update rule, $\theta \leftarrow \theta + \eta R(\mathbf{x})\nabla_\theta \log \pi_\theta(\mathbf{x})$, as the optimization method (Williams, 1992). After training the policy, we can obtain the solution by sampling from the trained policy.

Algorithm 2 Exact Greedy (Nemhauser et al., 1978)

```
**Input:** a monotone set function $a : 2^{\mathcal{X}} \to \mathbb{R}$, and a cardinality constraint $n$.
**Initialize** $B_0 = \emptyset$.
**for** $i = 0$ **to** $n - 1$ **do**
    $\mathbf{x}_i^* \leftarrow \text{argmax}_{\mathbf{x} \in \mathcal{X} \setminus B_i} \Delta_a(\mathbf{x} \mid B_i)$.
    $B_{i+1} \leftarrow B_i \cup \{\mathbf{x}_i^*\}$.
**end for**
**Return** $B_n$
```

Algorithm 3 Approx. Greedy (Goundan & Schulz, 2007)

```
**Input:** a monotone set function $a : 2^{\mathcal{X}} \to \mathbb{R}$, a maximization algorithm $\mathcal{A}$, and a cardinality constraint $n$.
**Initialize** $B_0 = \emptyset$.
**for** $i = 0$ **to** $n - 1$ **do**
    $\mathbf{x}_i \leftarrow$ solution by $\mathcal{A}$ when maximizing $\Delta_a(\cdot \mid B_i))$.
    $B_{i+1} \leftarrow B_i \cup \{\mathbf{x}_i\}$.
**end for**
**Return** $B_n$
```

Algorithm 4 Greedy Sample $\text{GS}(a, \pi_\theta^{\text{set}}, k, l)$

```
**Input:** a monotone set function $a$, a set-conditioned policy $\pi_\theta^{\text{set}}$, a set size $k$, the number of samples $l$.
**Initialize** $B_0 = \emptyset$.
**for** $i = 0$ **to** $k - 1$ **do**
    Sample $l$ candidates $\mathbf{x}_{i,0}, \dots, \mathbf{x}_{i,l-1} \sim \pi_\theta^{\text{set}}(\cdot \mid B_i)$.
    idx $\leftarrow \text{argmax}_{j \in [l]} \Delta_a(\mathbf{x}_{i,j} \mid B_i)$.
    $B_{i+1} \leftarrow B_i \cup \{\mathbf{x}_{\text{i,idx}}\}$.
**end for**
**Return** $B_k$
```

3. Methods

The subset selection problem (Equation (2)) for each inner loop is challenging due to its large search space of the size $\mathcal{O}(|\mathcal{X}|^n)$, which increases exponentially as the cardinality constraint n increases. To tackle these challenges, a simple yet effective approach is the greedy algorithm (Algorithm 2), which decomposes the subset selection problem into a series of smaller, more manageable subproblems, each of size $\mathcal{O}(|\mathcal{X}|)$ (Nemhauser et al., 1978). Specifically, each greedy subproblem maximizes the *marginal gain* $\Delta_a(\cdot \mid B) : \mathcal{X} \to \mathbb{R}$ of a set function $a : 2^{\mathcal{X}} \to \mathbb{R}$, defined as $\Delta_a(\mathbf{x} \mid B) := a(B \cup \{\mathbf{x}\}) - a(B)$ (Krause & Golovin, 2014). However, the combinatorial space $\mathcal{X}$ often has a large size owing to its high-dimensional characteristics in practical applications, such as in biological sequence design (Stanton et al., 2022). Hence, finding the exact solution for each subproblem remains a formidable challenge. Addressing this, we focus on the approximated greedy algorithm (Algorithm 3), which utilizes a scalable maximization algorithm $\mathcal{A}$ such as sampling-based heuristics, genetic algorithms, or MDP-based methods like RL to approximate solutions within the large space of $\mathcal{X}$ (Goundan & Schulz, 2007; Mirzasoleiman et al., 2015; Prasad et al., 2014; Williams, 1992).

On the other hand, the sequential nature of the greedy-style algorithms potentially hinders efficiency, as each subproblem depends on the solutions of preceding steps, complicating the parallelization of the overall process. To this end, we introduce a novel greedy-style subset selection method utilizing the greedy policy, a set-conditioned policy trained to handle all subproblems, with its novel training algorithm, thereby amortizing the burden of solving n subproblems. Furthermore, we elucidate the theoretical bound of the approximated greedy algorithm under various conditions of the set function, as encountered in realistic scenarios.

3.1. Learning Greedy Policy

To begin, we first introduce the concepts of a set-conditioned policy and a greedy sampling distribution. A *set-conditioned policy* π_θ^{set} is a policy that samples a candidate $\mathbf{x} \sim \pi_\theta^{\text{set}}(\cdot \mid B)$ conditioned on any subset B. For any subproblem with a subset B, which maximizes $\Delta_a(\cdot \mid B)$, we may utilize $\pi_\theta^{\text{set}}(\cdot \mid B)$ as a proposal distribution to sample the solution. In this context, we define a *greedy sampling* distribution $\text{GS}(a, \pi_\theta^{\text{set}}, k, l)$ on k-subsets of $\mathcal{X}$ as in Algorithm 4.

Instead of regarding greedy subproblems as individual problems to solve, we amortize all subproblems into a single problem of training a set-conditioned policy. Specifically, the goal is to train a set-conditioned policy to be the *greedy policy* $\pi_{\theta^*}^{\text{set}}$ that can exactly solve any subproblems encountered during the greedy sampling of $\pi_{\theta^*}^{\text{set}}$ itself. To formally define the greedy policy, we first define the expected gain.

Definition 3.1. (Expected Gain) We define $\mathcal{J}(\theta, \theta')$ as the expected gain by π_θ^{set} given behavior policy $\pi_{\theta'}^{\text{set}}$ as:

$$\mathcal{J}(\theta, \theta') := \frac{1}{n} \sum_{k=0}^{n-1} \mathbb{E}_{B \sim \text{GS}(a, \pi_{\theta'}^{\text{set}}, k, 1)} [\mathbb{E}_{\mathbf{x} \sim \pi_\theta^{\text{set}}(\cdot \mid B)} [\Delta_a(\mathbf{x} \mid B)]].$$

In short, the expected gain is the expected value of the marginal gain $\Delta_a(\mathbf{x} \mid B)$ where $\mathbf{x} \sim \pi_\theta^{\text{set}}(\cdot \mid B)$ and B is any subset encountered during the greedy sampling of $\pi_{\theta'}^{\text{set}}$. Using Definition 3.1, we formally define the greedy policy.

Definition 3.2. (Greedy Policy) A set conditioned policy $\pi_{\theta^*}^{\text{set}}$ is the greedy policy if $\pi_{\theta^*}^{\text{set}}$ is a maximizer of the expected gain given itself, *i.e.*, $\theta^* = \text{argmax}_{\theta \in \Theta} \mathcal{J}(\theta, \theta^*)$.

To explain that the greedy policy defined in Definition 3.2 satisfies the desired property, we introduce Lemma 3.3.

Lemma 3.3. *$\text{GS}(a, \pi_{\theta^*}^{set}, n, l)$ samples exact greedy solutions almost surely if $\pi_{\theta^*}^{set}$ is the greedy policy.*

Hence, the greedy policy is able to address all greedy subproblems and replicate the exact greedy algorithm.

From now on, we introduce the training method to achieve the greedy policy. To start, we define the partial derivative step, a fundamental component of our update rules.

Definition 3.4. Let $F : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ be any continuously differentiable function. We define the *partial derivative step* on $u \in \mathcal{U}$ given any behavior $u' \in \mathcal{U}$ as $\mathrm{PD}_F(u; u', \eta) := u + \eta \frac{\partial}{\partial u} F(u, u')$.

Using the partial derivative step defined in Definition 3.4, we introduce two update rules along with their validity. Similar to the assumptions such as strong convexity or smoothness used to ensure the convergence of the gradient descent algorithm (Bubeck, 2015), we assume several conditions (see Appendix A.1) and demonstrate the convergence of the update rules to the greedy policy under these assumptions.

Theorem 3.5. *Let $F : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ be a function with some nice conditions. Also, assume that there exists $u^* \in \mathcal{U}$ such that $u^* = \mathrm{argmax}_{u \in \mathcal{U}} F(u, u^*)$. Then, by iterating the update rule $u \leftarrow \mathrm{PD}_F(u; u' = u, \eta)$, u converges to u^* for a small $\eta > 0$. Furthermore, for any $N_t > 0$, by iterating the update rule with N_t partial derivative steps with a fixed behavior, i.e., $u \leftarrow (\mathrm{PD}_F(\cdot; u' = u, \eta))^{N_t}(u)$, u converges to u^* for a small $\eta > 0$.*

Proof. Please refer to Appendix A.1 for the detailed statements and proofs. □

For clarity, the update rule with N_t partial derivative steps in Theorem 3.5 on u returns u_n where

$$u_{i+1} = \mathrm{PD}_F(u_i; u' = u, \eta)$$

for $i = 0, \ldots, n-1$, and $u_0 = u$. Note that in practical scenarios, $\mathcal{J}$ may not fulfill these assumptions. However, our method still demonstrates empirical effectiveness in the practical scenarios, as shown in Section 4.

Due to the infeasible expectation in $\mathcal{J}$, we apply the update rule with an MC estimator $\hat{g}$ of $\frac{\partial}{\partial \theta}\mathcal{J}(\theta, \theta')$ which can be computed as in Proposition 3.6.

Proposition 3.6. *(Policy gradient for $\mathcal{J}$) For any baseline set function $b : 2^{\mathcal{X}} \to \mathbb{R}$ and the number of episodes N_e,*

$$\hat{g} = \frac{1}{N_e} \sum_{j=0}^{N_e - 1} (\Delta_a(\mathbf{x}^{(j)} \mid B) - b(B)) \nabla_\theta \log \pi_\theta^{set}(\mathbf{x}^{(j)} \mid B),$$

is an unbiased MC estimator of the partial derivative $\frac{\partial}{\partial \theta}\mathcal{J}(\theta, \theta')$ where $B \sim \frac{1}{n}\sum_{k=0}^{n-1} \mathrm{GS}(a, \pi_{\theta'}^{set}, k, 1)$ and $\mathbf{x}^{(j)} \sim \pi_\theta^{set}(\cdot \mid B)$ for all $j \in [N_e]$.

Algorithm 5 outlines the overall process of our training algorithm. Since every N_t step shares the behavior policy,

Algorithm 5 Training Set-Conditioned Policy

```
**Input:** a monotone set function $a : 2^{\mathcal{X}} \to \mathbb{R}$, a cardinality constraint $n$, the number of updates $N_u$, the number of episodes per update $N_e$, a learning rate $\eta$, and a period of the behavior policy update $N_t$.
**Initialize** $\theta$ randomly.
**for** $i = 0$ **to** $N_u - 1$ **do**
  **if** $i \,\%\, N_t = 0$ **then**
    Update the behavior policy $\pi_{\theta'}^{\text{set}} \leftarrow \pi_\theta^{\text{set}}$.
  **end if**
  Sample a set size $k \sim \mathrm{Unif}([n])$.
  Sample a subset $B \sim \mathrm{GS}(a, \pi_{\theta'}^{\text{set}}, k, 1)$.
  Sample $N_e$ candidates $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N_e)} \sim \pi_\theta^{\text{set}}(\cdot \mid B)$.
  **for** $j = 0$ **to** $N_e - 1$ **do**
    Compute returns $r_j \leftarrow \Delta_a(\mathbf{x}^{(j)} \mid B)$.
    Normalize $\hat{r}_j \leftarrow (r_j - \mathrm{mean}(r))/(\mathrm{std}(r) + \epsilon)$.
  **end for**
  Update $\theta \leftarrow \theta + \eta \sum_{j=0}^{N_e-1} \nabla_\theta[\hat{r}_j \log \pi_\theta^{\text{set}}(\mathbf{x}^{(j)} \mid B)]$.
**end for**
```

we can further speed-up the subset sampling by concurrently sampling N_t subsets parallelly. For the stable training, we use baseline techniques to normalize the returns. Finally, Figure 1 summarizes our proposed learning method.

3.2. Architecture for set-conditioned policy

In this section, we explain the architecture of a set-conditioned policy. Our architecture is inspired by preference-conditioned policies (Xi Lin, 2022; Jain et al., 2023; Zhu et al., 2023). The architecture of a preference-conditioned policy π_θ^{pc} can be summarized as follows:

$$\pi_\theta^{\text{pc}}(\mathbf{s} \mid \mathbf{v}) = \mathrm{Dec}(\mathrm{Enc}_{\text{pref}}(\mathbf{v}) \oplus \mathrm{Enc}_{\text{state}}(\mathbf{s})),$$

where $\mathbf{v}$ is a preference vector, $\mathbf{s}$ is a given state, and the decoder Dec outputs a probability vector on the action space. Instead of the preference encoder $\mathrm{Enc}_{\text{pref}}$, we propose an architecture using a set encoder $\mathrm{Enc}_{\text{set}}$, *i.e.*,

$$\pi_\theta^{\text{set}}(\mathbf{s} \mid B) = \mathrm{Dec}(\mathrm{Enc}_{\text{set}}(B) \oplus \mathrm{Enc}_{\text{state}}(\mathbf{s})).$$

For the set encoder, we utilize the *deep set* architecture, which is designed to encode a set of continuous vectors into a single embedding vector (Zaheer et al., 2017). We extract $\mathrm{feat}(\mathbf{x})$, the lower-dimensional continuous features to guide the policy, for each object $\mathbf{x} \in B$, and utilize the set of extracted features as input to the deep set encoder, *i.e.*, $\mathrm{Enc}_{\text{set}}(B) := \mathrm{DeepSet}(\{\mathrm{feat}(\mathbf{x}) \mid \mathbf{x} \in B\})$. It appears necessary to train an auxiliary feature extractor for the combinatorial space $\mathcal{X}$, but we already have a straightforward candidate for feat, especially when using the deterministic surrogate model $\tilde{\mathbf{f}}$ with HVI, thanks to Lemma 3.7.

Lemma 3.7. *For any $B, B' \subset \mathcal{X}$ satisfying $\tilde{f}(B) = \tilde{f}(B')$, $\Delta_a(\mathbf{x} \mid B) = \Delta_a(\mathbf{x} \mid B')$ for all $\mathbf{x} \in \mathcal{X}$.*

Lemma 3.7 indicates that $\pi_\theta^{\text{set}}(\cdot \mid B)$ and $\pi_\theta^{\text{set}}(\cdot \mid B')$ solve identical problems if $\tilde{\mathbf{f}}(B) = \tilde{\mathbf{f}}(B')$. Hence, we simply set $\text{feat}(\mathbf{x}) = \tilde{f}(\mathbf{x}) \in \mathbb{R}^m$. For cases using HVI-based batch acquisition functions with a statistical surrogate model $\tilde{f}$, we set $\text{feat}(\mathbf{x}) = \tilde{f}_{\text{UCB}}(\mathbf{x}) \in \mathbb{R}^m$.

For the biological sequence design problems, we further utilize the MLM model, trained on previously evaluated data, into the decoder, inspired by the architecture proposed in LaMBO (Stanton et al., 2022). Please refer to Appendix B.2 for more details on architectures.

Utilizing a learned set-conditioned policy $\pi_{\theta^*}^{\text{set}}$, we construct a proposal batch of n candidates for querying by sampling from $\text{GS}(a, \pi_{\theta^*}^{\text{set}}, n, l)$. Please refer to Appendix A.2 for the proofs of Lemma 3.3, Proposition 3.6, and Lemma 3.7

3.3. Bounds for Approximated Greedy Algorithm

In this section, we introduce bounds for the approximated greedy algorithm under various conditions of the set function, as encountered in active learning scenarios. First, we introduce the concept of an α-approximation algorithm which indicates the amount of approximation as in Definition 3.8.

Definition 3.8. (α-Approximation Algorithm) An algorithm $\mathcal{A}$ is *α-approximation algorithm* if $\mathbf{x}_i$ found by $\mathcal{A}$ is an α-approximation to the exact solution $\mathbf{x}_i^*$ for each step in Algorithm 3, *i.e.*, $\Delta_a(\mathbf{x}_i \mid B_i) \geq \alpha \Delta_a(\mathbf{x}_i^* \mid B_i) = \alpha \max_{\mathbf{x} \in \mathcal{X} \setminus B_i} \Delta(\mathbf{x} \mid B_i)$.

Prior research established bounds for approximated greedy algorithms in the term of α for submodular batch acquisition functions (Goundan & Schulz, 2007). Common batch acquisition functions, such as EHVI and PES, are submodular, but their exact values are infeasible to compute due to the expectation involved (Daulton et al., 2020; Garrido-Merchán et al., 2023). Instead, these values are estimated using methods like MC sampling or expectation propagation. Hence, the realized values of these acquisition functions can be near-submodular rather than strictly submodular.

Inspired by Das & Kempe (2018), we propose a bound for the approximated greedy algorithm when the batch acquisition function a is near-submodular, using the submodularity ratio γ which quantifies the degree of submodularity in a.

Theorem 3.9. *Let $a : 2^{\mathcal{X}} \to \mathbb{R}$ be a non-negative monotone set function. If $\mathcal{A}$ is an α-approximation algorithm, the resulting solution B_n of Algorithm 3 is an $(1 - 1/e^{\alpha\gamma_{B_n,n}(a)})$-approximation to the optimal n-subset B_n^*, i.e., $a(B_n) \geq (1 - 1/e^{\alpha\gamma_{B_n,n}(a)})a(B_n^*)$.*

Furthermore, there are works adopting heuristics to enhance the input diversity among evaluated candidates, aiming to identify diverse modes in the search space (Konakovic Lukovic et al., 2020; Jain et al., 2022; 2023; Zhu et al., 2023). The batch acquisition function for this *diversified subset selection problem* can be expressed as $a(B) := s(B) + \lambda \text{div}(B_{\text{prev}} \cup B)$, where λ is a coefficient controlling the tradeoff, s is a generic batch acquisition function, div quantifies the input diversity, and $B_{\text{prev}} \subset \mathcal{X}$ is the set of previously evaluated candidates. We mainly consider the diversity in the form of the *sum-dispersion*, defined as $\text{div}(U) := 1/2\ d(U, U) = 1/2 \sum_{\mathbf{x} \in U} \sum_{\mathbf{x}' \in U} d(\mathbf{x}, \mathbf{x}')$ for any metric d, due to its flexibility (Gollapudi & Sharma, 2009; Borodin et al., 2012). In this case, we can simplify the batch acquisition function a as follows:

$$a(B) = s(B) + \underbrace{\underbrace{\lambda \text{div}(B_{\text{prev}})}_{\text{constant on } B} + \lambda \sum_{\mathbf{x} \in B} \underbrace{d(\mathbf{x}, B_{\text{prev}})}_{\text{unary on } \mathbf{x}}}_{=:\text{aux}(B),\ \text{non-negative monotone modular}} + \lambda \text{div}(B).$$

For simplicity, we assume that the batch acquisition function is given by $a(B) = s(B) + \lambda \text{div}(B)$, neglecting the non-negative monotone modular term $\text{aux}(B)$, since $(s + \text{aux})$ is non-negative monotone for any non-negative monotone s (Fujishige, 2005). Now, we propose a bound for the approximated greedy algorithm for the diversified subset selection problem, extending the bound for submodular cases proved in Borodin et al. (2012).

Theorem 3.10. *Let s be a non-negative monotone set function and div be a sum-dispersion. If $\mathcal{A}$ is an α-approximation algorithm, Algorithm 3 with the set function $(s/2 + \lambda \text{div})$ returns B_n, an $(\alpha\hat{\gamma}/2)$-approximation to the optimal n-subset B_n^* of $(s + \lambda \text{div})$ where $\hat{\gamma} := \gamma_{B_n \cup B_n^*, n}(s)$.*

Theorem 3.9 and Theorem 3.10 demonstrate that even if our learning algorithm does not perfectly learn the greedy policy, the performance bound can still be guaranteed in terms of the level of approximation α in subproblems, even under these realistic conditions beyond submodularity. Please refer to Appendix A.3 for the detailed statements and proofs of the theorems, as well as their connections to prior works.

4. Experiments

We validate the performance of our proposed method on various benchmarks on sequence design problems based on the tasks suggested by Stanton et al. (2022) and Jain et al. (2023). First, we outline the benchmarks and the baseline methods. Next, we compare the subset selection performance of our method with the baseline methods on single-round synthetic tasks, which correspond to the deterministic surrogate model scenario. Finally, we present the results on batch BO scenarios, which utilize stochastic surrogate models. Our implementation is available at https://github.com/snu-mllab/GreedyPolicyForMOCO.

Table 1: Subset selection results on three bigrams tasks with various cardinality constraint n. Each value indicates the Hypervolume indicator of discovered subset. The mean and standard deviation values are calculated for 10 trials.

	Hypervolume Indicator (↑)								
	2 Bigrams		3 Bigrams			4 Bigrams			
Method	$n = 4$	$n = 16$	$n = 4$	$n = 16$	$n = 64$	$n = 4$	$n = 16$	$n = 64$	$n = 256$
Optimum	0.630		0.409			0.106			
Exact Greedy	0.568	0.630	0.350	0.408	0.409	0.055	0.078	0.097	0.106
Ours	**0.568** (0.000)	**0.630** (0.000)	**0.329** (0.005)	**0.349** (0.007)	**0.359** (0.003)	**0.055** (0.000)	**0.077** (0.000)	**0.091** (0.002)	**0.094** (0.003)
PC-RL (TS)	0.558 (0.007)	0.620 (0.003)	0.318 (0.012)	0.347 (0.003)	**0.359** (0.004)	0.040 (0.005)	0.054 (0.003)	0.071 (0.002)	0.082 (0.007)
PC-RL (WS)	0.522 (0.030)	0.583 (0.018)	0.310 (0.007)	0.337 (0.008)	0.347 (0.007)	0.009 (0.009)	0.016 (0.005)	0.028 (0.005)	0.032 (0.006)
Greedy + RL	0.518 (0.002)	0.518 (0.040)	0.320 (0.000)	0.320 (0.008)	0.322 (0.001)	0.047 (0.004)	0.062 (0.005)	0.065 (0.008)	0.070 (0.004)
Greedy + HC	0.063 (0.028)	0.063 (0.028)	0.182 (0.021)	0.182 (0.021)	0.171 (0.038)	0.014 (0.008)	0.014 (0.008)	0.013 (0.008)	0.001 (0.001)
Greedy + RS	0.025 (0.002)	0.027 (0.002)	0.002 (0.001)	0.003 (0.000)	0.003 (0.000)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)

Table 2: Subset selection results on the DNA aptamer design task with various cardinality constraint n. The mean and standard deviation values are calculated for 10 trials.

	Hypervolume Indicator (↑)			
Method	$n = 4$	$n = 16$	$n = 64$	$n = 256$
Ours	**0.662** (0.019)	**0.717** (0.025)	**0.764** (0.045)	**0.778** (0.026)
PC-RL (TS)	0.515 (0.041)	0.658 (0.060)	0.712 (0.052)	0.731 (0.042)
PC-RL (WS)	0.479 (0.035)	0.530 (0.009)	0.587 (0.027)	0.611 (0.017)
Greedy + RL	0.551 (0.029)	0.705 (0.027)	0.749 (0.034)	0.739 (0.023)
Greedy + HC	0.367 (0.039)	0.388 (0.047)	0.377 (0.050)	0.372 (0.042)
Greedy + RS	0.199 (0.012)	0.226 (0.012)	0.231 (0.015)	0.232 (0.015)

4.1. Settings

Single-round experiments on synthetic tasks. In this setting, we assume that a deterministic surrogate model is given as a synthetic function. We mainly consider the subset selection problem that maximizes the Hypervolume indicator value of given deterministic synthetic functions on bigram matching tasks with various numbers of objectives and a DNA aptamer design task with three objectives computed by NUPACK library (Jain et al., 2023; Zadeh et al., 2011). We compare our method with preference-conditioned RL methods with Chebyshev scalarization, denoted PC-RL (TS), and weight scalarization, denoted PC-RL (WS) (Xi Lin, 2022). We also compare our method with the approximated greedy algorithm augmented with combinatorial algorithms: RL (Greedy + RL), hill climbing (Greedy + HC), and random sampling (Greedy + RS) (Williams, 1992; Selman & Gomes, 2006; Mirzasoleiman et al., 2015). For a fair comparison, we fix the number of queries to the deterministic surrogate model across all methods.

Batch BO experiments. In this scenario, we adopt the benchmark tasks proposed by Stanton et al. (2022) to evaluate the performance of our method with statistical surrogate models. We compare our method with several active learning methods: LaMBO, an LSO method; MBGA, a model-based genetic algorithm; and AL-MOGFN, a GFlowNet-based active learning method that maximizes individual acquisition values while enhancing the diversity (Stanton et al., 2022; Jain et al., 2023). For these methods, we use the same architecture and training algorithm for the statistical surrogate model (Shah & Ghahramani, 2016). As in Stanton et al. (2022), we utilize NEHVI as the batch acquisition for all methods. We also consider NSGA-II as a baseline method to show the performance of a model-free method that is not an active learning approach (Deb et al., 2002).

Though active learning frameworks assume that the surrogate model is much cheaper than the oracle, we set the number of surrogate model queries of our method to be comparable to the baseline methods for a fair comparison. Please refer to Appendix B.3 for more details on the experimental settings containing tasks and baselines.

4.2. Results on Synthetic Tasks

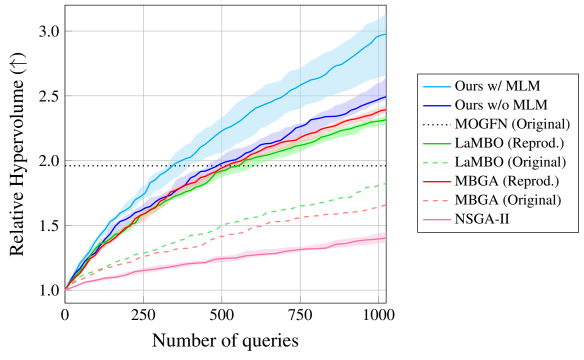
Table 1 summarizes the single-round subset selection results on bigrams tasks. The results show that our method consistently outperforms the baseline methods, searching batches with higher Hypervolume indicator values compared to the baseline methods for all bigram tasks with various cardinality constraint values we consider. In these tasks, we can obtain the optimum and Hypervolume indicator values of exact greedy solutions by rule-based backtracking search. Notably, in the 2 bigrams task, our proposed algorithm finds subsets with Hypervolume indicator values that are the same as the exact greedy solutions. Table 2 summarizes the subset selection results on the DNA aptamer design task. The results state that our method also outperforms the baseline methods in the DNA aptamer task, demonstrating the wide applicability of our method. Please refer to Appendix C.1 for the additional results on single-round synthetic tasks.

4.3. Results on Batch BO Scenarios

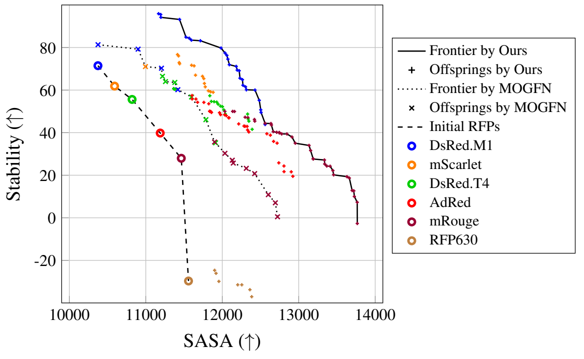
First, we note that the prior implementations of LaMBO and MBGA had several issues, such as incorrect computation of NEHVI. We fix these issues and reproduce the results of baseline methods with more update steps for a

Table 3: Subset selection results in the first round of the batch BO benchmarks (RFP, 3 Bigrams, small molecules) with NEHVI. The mean and standard deviation values are calculated for 10 trials.

Method	NEHVI Value (↑)		
	RFP	3 Bigrams	Molecules
Ours	**0.779** (0.045)	**16.444** (1.529)	**0.698** (0.104)
LaMBO	0.591 (0.033)	9.922 (0.688)	0.545 (0.064)
MBGA	0.654 (0.052)	12.376 (1.576)	0.483 (0.124)



(a) Multi-round active learning results.



(b) Discovered frontiers

Figure 2: Multi-round active learning results and the discovered frontiers on the RFP task under a query limit of 1024. (a) Midpoint, lower, and upper boundaries show the 50th, 30th, and 70th percentiles, respectively, derived from 10 trials. (b) Colored circles indicates ancestor proteins.

fair comparison with our method. For detailed information, please see Appendix D. Figure 2 presents the experimental results on the RFP task. As shown in Figure 2a, both variations of our methods outperform the baseline methods in the RFP task. Remarkably, our method with MLM achieves 25% higher relative Hypervolume indicator value compared to the best baseline results from MBGA. In a different view, our method with MLM demonstrates comparable performance while requiring 1.69 times fewer queries. Figure 2b illustrates the frontiers discovered by our method and AL-MOGFN. The frontiers obtained through our method completely dominate those previously discovered by AL-MOGFN, demonstrating the effectiveness of our proposed active learning method. Additionally, we provide visualizations of the non-dominated offsprings for each color, highlighting that our method successfully discovers improved offsprings for all ancestor proteins.

To directly validate the subset selection performance in batch BO scenarios, we evaluate the resulting batch acquisition value of each method in the first round with the same surrogate model and the initial data. Table 3 demonstrates that our method achieves higher batch acquisition values compared to baseline active learning methods, indicating the superiority of our method in inner loop optimization with a stochastic surrogate model. For additional results on batch BO experiments, please refer to Appendix C.2.

5. Related Works

Genetic algorithms (GAs) for MOCO A widely-used off-the-shelf GA method for multi-objective optimization, NSGA-II, employs random mutations and a non-dominated sorting for iterative population updates (Deb et al., 2002; Konak et al., 2006). Miret et al. (2022) improved GA to better handle large combinatorial search spaces by utilizing graph neural networks. While GAs are recognized for their simplicity and adaptability across various types of problems, generic GA methods may not be suitable for expensive MOCO problems because they often necessitate a large number of queries (Turner et al., 2021).

MOCO based on Markov decision processes (MDPs). An emerging research direction treats combinatorial optimization problems as decision-making problems, framing them within the context of MDPs and training policies via RL or GFlowNet frameworks (Zoph & Le, 2017; Bello et al., 2017; Mirhoseini et al., 2021; Bengio et al., 2023). In MOCO research, this perspective has led to the development of methods that train policies to generate the Pareto set (Roijers et al., 2013; Yang et al., 2019; Li et al., 2019; Kool et al., 2019; Xi Lin, 2022; Jain et al., 2023; Zhu et al., 2023). These methods predominantly utilize preference-based scalarization techniques, such as weighted scalarization or weighted Chebyshev scalarization, to decompose multi-objective problems into single-objective subproblems (Giagkiozis & Fleming, 2015). Li et al. (2019) and Kool et al. (2019) train multiple RL agents, each corresponding to a single subproblem. Yang et al. (2019) amortize subproblems into a single problem with a stochastic reward scalarized by random preference vectors, demonstrating the training of a single RL agent to solve MOCO problems. Xi Lin (2022) introduce a preference-conditioned RL agent, which is trained to maximize a scalarized reward corresponding to the conditioned preference vector. In a different approach, Jain et al. (2023) and Zhu et al. (2023) employ GFlowNet frameworks to enhance the diversity of the learned solutions, applying this technique to solve expensive MOCO problems, such as biological sequence design and molecular graph discovery, in active learning frameworks.

Latent space optimization (LSO) for MOCO. Rather than directly searching for candidates in the explicit combinatorial space, Stanton et al. (2022) proposed LaMBO, a LSO-style inner loop optimization method designed for discrete sequence data. This method involves training an autoencoder and proposing a batch of candidates through first-order optimization of the batch acquisition function in a continuous latent space (Gómez-Bombarelli et al., 2016; Tripp et al., 2020; Shah & Ghahramani, 2016).

6. Conclusion

We propose a query-efficient optimization method for expensive MOCO problems based on active learning utilizing a novel greedy-style subset selection algorithm. In contrast to prior methods, our subset selection method explicitly optimizes the batch acquisition function on the combinatorial space. Moreover, our subset selection algorithm trains a greedy policy to address all greedy subproblems simultaneously, overcoming the typical difficulty of parallelization in greedy-style approaches. By utilizing the trained greedy policy, our algorithm constructs the proposal batch by sequential sampling from the greedy policy. Furthermore, we extend the theoretical bound on approximated greedy algorithms for various types of set functions containing monotone set functions and sum-dispersion functions. Empirical results on single-round subset selection benchmark in biological sequence designs show that our method consistently outperforms baseline methods, finding the batch with higher batch acquisition value. Surprisingly, in multi-round active learning for red fluorescent protein design, our approach achieves the same level of performance as baseline methods but with $1.69\times$ fewer queries, demonstrating its effectiveness and efficiency.

Impact Statement

The primary objective of our research is to improve the end-to-end process for solving expensive MOCO problems. Expensive MOCO problems encompass a wide range of complex challenges in society, including drug discovery and chip design, which have widespread impact. Through our approach of generating superior proposal candidates using a greedy policy, our work has the potential to contribute to the quicker development of new medications, and the creation of more efficient electronic devices, potentially offering benefits to both the industry and society.

Acknowledgement

The work was done as part of the Meta–NYU mentorship program and partly supported by the National Science Foundation (under NSF Award 1922658). Kyunghyun Cho is supported by the Samsung Advanced Institute of Technology (under the project Next Generation Deep Learning: From Pattern Recognition to AI). Deokjae Lee and Hyun Oh Song are supported by Samsung Advanced Institute of Technology, Samsung Electronics Co., Ltd. (IO220810-01900-01), Samsung Electronics' Mobile eXperience (MX) Business, Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [No. RS-2020-II200882, (SW STAR LAB) Development of deployable learning intelligence via self-sustainable and trustworthy machine learning and No. RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)], the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00354036), and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (RS-2023-00274280). Kyunghyun Cho is the corresponding author.

References

Aggarwal, C., Kong, X., Gu, Q., Han, J., and Yu, P. Active learning: A survey. In *Data Classification*, 2014.

Agnesina, A., Rajvanshi, P., Yang, T., Pradipta, G., Jiao, A., Keller, B., Khailany, B., and Ren, H. Autodmp: Automated dreamplace-based macro placement. In *ISPD*, 2023.

Azimi, J., Fern, A., and Fern, X. Batch bayesian optimization via simulation matching. In *NeurIPS*, 2010.

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. Neural combinatorial optimization with reinforcement learning. In *ICLR*, 2017.

Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. Gflownet foundations. In *arXiv 2111.09266*, 2023.

Bickerton, R., Paolini, G., Besnard, J., Muresan, S., and Hopkins, A. Quantifying the chemical beauty of drugs. In *Nature chemistry*, 2012.

Blyth, T. S. Lattices and ordered algebraic structures. In *Springer Verlag*, 2005.

Borodin, A., Lee, H. C., and Ye, Y. Max-sum diversification, monotone submodular functions and dynamic updates. In *ACM SIGMOD-SIGACT-SIGAI*, 2012.

Bubeck, S. Convex optimization: Algorithms and complexity. In *arXiv 1405.4980*, 2015.

Das, A. and Kempe, D. Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection. In *JMLR*, 2018.

Daulton, S., Balandat, M., and Bakshy, E. Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization. In *NeurIPS*, 2020.

Daulton, S., Balandat, M., and Bakshy, E. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. In *NeurIPS*, 2021.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: Nsga-ii. In *IEEE Transactions on Evolutionary Computation*, 2002.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

Ehrgott, M. Multicriteria optimization. In *Springer Science & Business Media*, 2005.

Emmerich, M., Yang, K., Deutz, A., Wang, H., and Fonseca, C. A multicriteria generalization of bayesian global optimization. In *Advances in Stochastic and Deterministic Global Optimization*, 2015.

Fujishige, S. Submodular functions and optimization. In *Elsevier*, 2005.

Garrido-Merchán, E. C., Fernández-Sánchez, D., and Hernández-Lobato, D. Parallel predictive entropy search for multi-objective bayesian optimization with constraints applied to the tuning of machine learning algorithms. In *Expert Systems with Applications*, 2023.

Giagkiozis, I. and Fleming, P. Methods for multi-objective optimization: An analysis. In *Information Sciences*, 2015.

Gollapudi, S. and Sharma, A. An axiomatic approach for result diversification. In *Proceedings of the 18th International Conference on World Wide Web*, 2009.

González, J. I., Dai, Z., Hennig, P., and Lawrence, N. D. Batch bayesian optimization via local penalization. In *AISTATS*, 2015.

Goundan, P. and Schulz, A. Revisiting the greedy approach to submodular set function maximization. In *Manuscript*, 2007.

Gruver, N., Stanton, S. D., Frey, N. C., Rudner, T. G. J., Hotzel, I., Lafrance-Vanasse, J., Rajpal, A., Cho, K., and Wilson, A. G. Protein design with guided discrete diffusion. In *NeurIPS*, 2023.

Guerreiro, A. P., Fonseca, C. M., and Paquete, L. The hypervolume indicator: Computational problems and algorithms. In *ACM Comput. Surv.*, 2021.

Gómez-Bombarelli, R., Duvenaud, D., Hernández-Lobato, J., Aguilera-Iparraguirre, J., Hirzel, T., Adams, R., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. In *ACS Central Science*, 2016.

Jain, M., Bengio, E., Hernández-Garcia, A., Rector-Brooks, J., Dossou, B. F. P., Ekbote, C. A., Fu, J., Zhang, T., Kilgour, M., Zhang, D., Simine, L., Das, P., and Bengio, Y. Biological sequence design with GFlowNets. In *ICML*, 2022.

Jain, M., Raparthy, S. C., Hernández-Garcia, A., Rector-Brooks, J., Bengio, Y., Miret, S., and Bengio, E. Multi-objective gflownets. In *ICML*, 2023.

Jung, A. A fixed-point of view on gradient methods for big data. In *Frontiers in Applied Mathematics and Statistics*, 2017.

Konak, A., Coit, D. W., and Smith, A. E. Multi-objective optimization using genetic algorithms: A tutorial. In *Reliability engineering & system safety*, 2006.

Konakovic Lukovic, M., Tian, Y., and Matusik, W. Diversity-guided multi-objective bayesian optimization with batch evaluations. In *NeurIPS*, 2020.

Kool, W., van Hoof, H., and Welling, M. Attention, learn to solve routing problems! In *ICLR*, 2019.

Krause, A. and Golovin, D. Submodular function maximization. In *Cambridge University Press*, 2014.

Krenn, M., Häse, F., Nigam, A., Friederich, P., and Aspuru-Guzik, A. Self-referencing embedded strings (selfies): A 100In *Machine Learning: Science and Technology*, 2020.

Li, K., Zhang, T., and Wang, R. Deep reinforcement learning for multiobjective optimization. In *IEEE Transactions on Cybernetics*, 2019.

Lin, X., Yang, Z., Zhang, X.-Y., and Zhang, Q. Pareto set learning for expensive multi-objective optimization. In *NeurIPS*, 2022.

Miret, S., Chua, V. S., Marder, M., Phiellip, M., Jain, N., and Majumdar, S. Neuroevolution-enhanced multi-objective optimization for mixed-precision quantization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022.

Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J. W., Songhori, E. M., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Nazi, A., Pak, J., Tong, A., Srinivasa, K., Hang, W., Tuncer, E., Le, Q. V., Laudon, J., Ho, R., Carpenter, R., and Dean, J. A graph placement methodology for fast chip design. In *Nature*, 2021.

Mirzasoleiman, B., Badanidiyuru, A., Karbasi, A., Vondrák, J., and Krause, A. Lazier than lazy greedy. In *AAAI*, 2015.

Nemhauser, G., Wolsey, L., and Fisher, M. An analysis of approximations for maximizing submodular set functions—i. In *Mathematical Programming*, 1978.

O'Donoghue, B., Osband, I., Munos, R., and Mnih, V. The uncertainty bellman equation and exploration. In *ICML*, 2017.

Park, J. W., Tagasovska, N., Maser, M., Ra, S., and Cho, K. Botied: Multi-objective bayesian optimization with tied multivariate ranks. In *arXiv 2306.00344*, 2023.

Polishchuk, P. G., Madzhidov, T. I., and Varnek, A. Estimation of the size of drug-like chemical space based on gdb-17 data. In *Journal of Computer-Aided Molecular Design*, 2013.

Prasad, A., Jegelka, S., and Batra, D. Submodular meets structured: Finding diverse subsets in exponentially-large structured item sets. In *NeurIPS*, 2014.

Ravi, S. S., Rosenkrantz, D. J., and Tayi, G. K. Heuristic and special case algorithms for dispersion problems. In *Operational Research*, 1994.

Roijers, D. M., Vamplew, P., Whiteson, S., and Dazeley, R. A survey of multi-objective sequential decision-making. In *Journal of Artificial Intelligence Research*, 2013.

Selman, B. and Gomes, C. P. Hill-climbing search. volume 81, pp. 82. Wiley, 2006.

Shah, A. and Ghahramani, Z. Pareto frontier learning with expensive correlated objectives. In *ICML*, 2016.

Shen, M. W., Bengio, E., Hajiramezanali, E., Loukas, A., Cho, K., and Biancalani, T. Towards understanding and improving gflownet training. In *ICML*, 2023.

Stanton, S., Maddox, W., Gruver, N., Maffettone, P., Delaney, E., Greenside, P., and Wilson, A. G. Accelerating bayesian optimization for biological sequence design with denoising autoencoders. In *ICML*, 2022.

Tripp, A., Daxberger, E., and Hernández-Lobato, J. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. In *NeurIPS*, 2020.

Tu, B., Gandy, A., Kantas, N., and Shafei, B. Joint entropy search for multi-objective bayesian optimization. In *NeurIPS*, 2022.

Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., and Guyon, I. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS Competition and Demonstration*, 2021.

Wang, J., Clark, S. C., Liu, E., and Frazier, P. Parallel bayesian global optimization of expensive functions. In *Operational Research*, 2016.

Ward, J. *Oblivious and non-oblivious local search for combinatorial optimization*. PhD thesis, 2012.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Language*, 1992.

Winter, R., Montanari, F., Steffen, A., Briem, H., Noé, F., and Clevert, D.-A. Efficient multi-objective molecular optimization in a continuous latent space. In *Chemical Science*, 2019.

Xi Lin, Zhiyuan Yang, Q. Z. Pareto set learning for neural multi-objective combinatorial optimization. In *ICLR*, 2022.

Yang, K., Affenzeller, M., and Dong, G. A parallel technique for multi-objective bayesian global optimization: Using a batch selection of probability of improvement. In *Swarm and Evolutionary Computation*, 2022.

Yang, R., Sun, X., and Narasimhan, K. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In *NeurIPS*, 2019.

Zadeh, J. N., Steenberg, C. D., Bois, J. S., Wolfe, B. R., Pierce, M. B., Khan, A. R., Dirks, R. M., and Pierce, N. A. Nupack: Analysis and design of nucleic acid systems. *Journal of computational chemistry*, 32(1):170–173, 2011.

Zaheer, M., Kottur, S., Ravanbhakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. J. Deep sets. In *NeurIPS*, 2017.

Zhou, Y. and Spanos, C. J. Causal meets submodular: Subset selection with directed information. In *NeurIPS*, 2016.

Zhu, Y., Wu, J., Hu, C., Yan, J., Hsieh, C.-Y., Hou, T., and Wu, J. Sample-efficient multi-objective molecular optimization with GFlownets. In *NeurIPS*, 2023.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. In *ICLR*, 2017.

A. Mathematical Details

A.1. Proof of Theorem 3.5

To start, we recall Definition 3.4 and Theorem 3.5.

Definition A.1. Let $F : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ be any continuously differentiable function. We define the *partial derivative step* on $u \in \mathcal{U}$ given any behavior $u' \in \mathcal{U}$ as $\mathrm{PD}_F(u; u', \eta) \coloneqq u + \eta \frac{\partial}{\partial u} F(u, u')$.

Theorem A.2. *Let $F : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ be a function with nice conditions. Also, assume that there exists $u^* \in \mathcal{U}$ such that $u^* = \mathrm{argmax}_{u \in \mathcal{U}} F(u, u^*)$. Then, by iterating the update rule $u \leftarrow \mathrm{PD}_F(u; u' = u, \eta)$, u converges to u^* for a small $\eta > 0$. Moreover, for any $N_t > 0$, by iterating the update rule with N_t partial derivative steps with a fixed behavior, i.e., $u \leftarrow (\mathrm{PD}_F(\cdot; u' = u, \eta))^{N_t}(u)$, u converges to u^* for a small $\eta > 0$.*

We first introduce the intuition of the proposed update rules. Assume that $F : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ is a smooth function and $F_{u'} \coloneqq F(\cdot, u') : \mathcal{U} \to \mathbb{R}$ is a strongly concave function on $\mathcal{U}$ for any $u' \in \mathcal{U}$. Under this assumption, if $u, u' \in \mathcal{U}$ satisfies

$$\frac{\partial}{\partial u} F(u, u') \bigg|_{u'=u} = 0, \tag{3}$$

we have the following equality:

$$(\nabla F_u)(u) = \frac{\partial}{\partial u} F(u, u') \bigg|_{u'=u} = 0.$$

Due to the strong concavity, F_u has a unique maximizer and we finally get the desired property as following:

$$u = \operatorname*{argmax}_{v \in \mathcal{U}} F_u(v) = \operatorname*{argmax}_{v \in \mathcal{U}} F(v, u).$$

Inspired by Jung (2017), we design two update rules, whose fixed point satisfies Equation (3), based on partial derivative steps in Definition A.1. For simplicity, we introduce notations $\nabla_1 F$, $\nabla_2 F$, $\nabla_1^2 F$, and $\nabla_2 \nabla_1 F$ as

$$(\nabla_1 F)(u, u') = \left(\frac{\partial}{\partial u} F\right)(u, u'), (\nabla_2 F)(u, u') = \left(\frac{\partial}{\partial u'} F\right)(u, u'),$$

$$(\nabla_1^2 F)(u, u') = \left(\frac{\partial^2}{\partial u^2} F\right)(u, u'), (\nabla_2 \nabla_1 F)(u, u') = \left(\frac{\partial^2}{\partial u' \partial u} F\right)(u, u').$$

From now on, we state the conditions on F required in our proof for justifying the convergence of the update rules as follows.

Conditions $(*)$:

1. F is a smooth function.
2. $F_{u'}$ is a strongly concave function and the eigenvalues of the hessian $(\nabla^2 F_{u'}) = \nabla_1^2 F(u, u')$ is uniformly bounded by two negative values $L \leq M < 0$, *i.e.*,

$$L \leq \lambda_{\text{min}} \left(\nabla_1^2 F(u, u')\right) \leq \lambda_{\text{max}} \left(\nabla_1^2 F(u, u')\right) \leq M < 0,$$

   for all $u, u' \in \mathcal{U}$.
3. The square matrix of second order derivatives $\nabla_2 \nabla_1 F(u, u')$ has singular values bounded above by $M' < -M$, *i.e.*,

$$0 \leq \sigma_{\text{min}} \left(\nabla_2 \nabla_1 F(u, u')\right) \leq \sigma_{\text{max}} \left(\nabla_2 \nabla_1 F(u, u')\right) \leq M' < -M \leq -L,$$

   for all $u, u' \in \mathcal{U}$.

Assuming these three conditions $(*)$ on F, we prove the following proposition first.

Proposition A.3. *If $F : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ satisfies three conditions $(*)$, there exists an $\eta > 0$ and $0 < \beta < 1$ such that the partial derivative step $\mathrm{PD}_F(u; u', \eta)$ satiesfies the following property:*

$$\|\mathrm{PD}_F(u_2; u_2', \eta) - \mathrm{PD}_F(u_1; u_1', \eta)\|_2 \le \beta \|u_2' - u_1'\|_2$$

for all $u_1, u_1', u_2, u_2' \in \mathcal{U}$ such that $\|u_2 - u_1\| \le \|u_2' - u_1'\|$.

Proof.

$$\begin{aligned}
&\|\mathrm{PD}_F(u_2; u_2', \eta) - \mathrm{PD}_F(u_1; u_1', \eta)\|_2 \\
&= \|(u_2 + \eta \nabla_1 F(u_2, u_2')) - (u_1 + \eta \nabla_1 F(u_1, u_1'))\|_2 \\
&= \|(u_2 - u_1) + \eta \left(\nabla_1 F(u_2, u_2') - \nabla_1 F(u_1, u_1')\right)\|_2 \\
&= \|(u_2 - u_1) + \eta \left(\nabla_1 F(u_2, u_2') - \nabla_1 F(u_1, u_2') + \nabla_1 F(u_1, u_2') - \nabla_1 F(u_1, u_1')\right)\|_2 \\
&= \|(u_2 - u_1) + \eta \left(\nabla_1 F(u_2, u_2') - \nabla_1 F(u_1, u_2')\right) + \eta \left(\nabla_1 F(u_1, u_2') - \nabla_1 F(u_1, u_1')\right)\|_2 \\
&= \left\|(u_2 - u_1) + \eta \int_0^1 \nabla_1^2 F(u_1 + t(u_2 - u_1), u_2')(u_2 - u_1) dt + \eta \int_0^1 \nabla_2 \nabla_1 F(u_1, u_1' + t(u_2' - u_1'))(u_2' - u_1') dt \right\|_2 \\
&\le \Bigg\| \underbrace{\left(I + \eta \int_0^1 \nabla_1^2 F(u_1 + t(u_2 - u_1), u_2') dt\right)}_{=:A} (u_2 - u_1) \Bigg\|_2 + \Bigg\| \underbrace{\left(\eta \int_0^1 \nabla_2 \nabla_1 F(u_1, u_1' + t(u_2' - u_1')) dt\right)}_{=:B} (u_2' - u_1') \Bigg\|_2 .
\end{aligned}$$

The last inequality is from the triangle inequality and the last equality is from the fundamental theorem of line integrals. By utilizing the notion of the spectral norm, we have

$$\|A(u_2 - u_1)\|_2 \le \sigma_{\text{max}}(A) \|u_2 - u_1\|_2 = \max(|\lambda_{\text{min}}(A)|, |\lambda_{\text{max}}(A)|) \|u_2 - u_1\|_2,$$

$$\|B(u_2' - u_1')\|_2 \le \sigma_{\text{max}}(B) \|u_2 - u_1\|_2.$$

Due to the second condition in $(*)$, A has eigenvalues bounded by $1 + \eta L$ and $1 + \eta M$, *i.e.*,

$$1 + \eta L \le \lambda_{\text{min}}(A) \le \lambda_{\text{max}}(A) \le 1 + \eta M.$$

Hence, for $0 < \eta < -\frac{1}{2L}$, A is positive definite since $\lambda_{\text{min}}(A) \ge 1 + \eta L > \frac{1}{2} > 0$. Thus, we have

$$\sigma_{\text{max}}(A) = \lambda_{\text{max}}(A) \le 1 + \eta M.$$

Moreover, utilizing the third condition in $(*)$, B has singular values bounded above by $\eta M'$, *i.e.*,

$$\sigma_{\text{max}}(B) \le \eta M'.$$

As a result, we get

$$\begin{aligned}
\|\mathrm{PD}_F(u_2; u_2', \eta) - &\mathrm{PD}_F(u_1; u_1', \eta)\|_2 \\
&\le \|A(u_2 - u_1)\|_2 + \|B(u_2' - u_1')\|_2 \\
&\le \sigma_{\text{max}}(A) \|u_2 - u_1\|_2 + \sigma_{\text{max}}(B) \|u_2' - u_1'\|_2 \\
&\le (\sigma_{\text{max}}(A) + \sigma_{\text{max}}(B)) \|u_2' - u_1'\|_2 \qquad (\because \|u_2 - u_1\|_2 \le \|u_2' - u_1'\|_2) \\
&\le (1 + \eta M + \eta M') \|u_2' - u_1'\|^2.
\end{aligned}$$

Since $0 < M' < -M \le -L$, we have $0 < 1 + \eta M + \eta M' < 1$ for $0 < \eta < -\frac{1}{2L}$. Hence for $0 < \eta < -\frac{1}{2L}$ and $\beta = (1 + \eta M + \eta M') < 1$, we proved that the partial derivative step satisfies the desired property, concluding the proof. $\square$

From Proposition A.3, we can derive two important corollaries.

Corollary A.4. *If $F : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ satisfies three conditions ($*$), there exists an $\eta > 0$ and $0 < \beta < 1$ such that the first update rule of Theorem 3.5, $u \leftarrow \mathrm{PD}_F(u; u' = u, \eta)$, is a β-contraction mapping, i.e.,*

$$\|\mathrm{PD}_F(u_2; u_2, \eta) - \mathrm{PD}_F(u_1; u_1, \eta)\|_2 \le \beta \|u_2 - u_1\|_2$$

for all $u_1, u_2 \in \mathcal{U}$.

Proof. This corollary corresponds to Proposition A.3 of the case that $u_1' = u_1$ and $u_2' = u_2$. □

Corollary A.5. *If $F : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ satisfies three conditions ($*$), there exists an $\eta > 0$ and $0 < \beta < 1$ such that the second update rule of Theorem 3.5, $u \leftarrow (\mathrm{PD}_F(\cdot; u' = u, \eta))^{N_t}(u)$, is a β-contraction mapping, i.e.,*

$$\|(\mathrm{PD}_F(\cdot; u' = u_2, \eta))^{N_t}(u_2) - (\mathrm{PD}_F(\cdot; u' = u_1, \eta))^{N_t}(u_1)\|_2 \le \beta \|u_2 - u_1\|_2.$$

Proof. For any given $u_1, u_2 \in \mathcal{U}$, we define the following recurrence relation:

1. Initialize $u_1^{(0)} = u_1$, $u_2^{(0)} = u_2$.
2. For $i = 0, ..., N_t - 1$,
$$u_1^{(i+1)} = \mathrm{PD}_F(u_1^{(i)}; u_1, \eta), \;\; u_2^{(i+1)} = \mathrm{PD}_F(u_2^{(i)}; u_2, \eta).$$

Let $0 < \beta < 1$ and $\eta > 0$ be the constants that satisfies the property in Proposition A.3. Then, we prove that $\|u_2^{(i)} - u_1^{(i)}\|_2 \le \beta \|u_2 - u_1\|_2$ for all $i = 1, \ldots, N_t$ by the mathematical induction on i. We considered the initial case of $i = 1$ in Corollary A.4. Next, assume that $\|u_2^{(i)} - u_1^{(i)}\|_2 \le \beta \|u_2 - u_1\|_2$ for some $i \ge 1$. By plugging in $u_1 \leftarrow u_1^{(i)}, u_2 \leftarrow u_2^{(i)}, u_1' \leftarrow u_1, u_2' \leftarrow u_2$ to Proposition A.3, we have

$$\|u_1^{(i+1)} - u_2^{(i+1)}\|_2 = \|\mathrm{PD}_F(u_1^{(i)}; u_1, \eta) - \mathrm{PD}_F(u_2^{(i)}; u_2, \eta)\|_2 \le \beta \|u_2 - u_1\|_2$$

since $\|u_2^{(i)} - u_1^{(i)}\|_2 \le \beta \|u_2 - u_1\|_2 < \|u_2 - u_1\|_2$. Thus, we complete the proof by mathematical induction on i. □

By utilizing these corollaries, we prove the convergence of our proposed update rules when F satisfies three conditions ($*$).

Proof of Theorem 3.5. For simplicity we notate two update rules by

$$\mathrm{upd}_1(u) = \mathrm{PD}_F(u; u' = u, \eta) \;\text{ and }\; \mathrm{upd}_2(u) = (\mathrm{PD}_F(\cdot; u' = u, \eta))^{N_t}(u).$$

From the condition in Theorem 3.5, there exists the $u^* \in \mathcal{U}$ such that $u^* = \mathrm{argmax}_{u \in \mathcal{U}} F(u, u^*) = \mathrm{argmax}_{u \in \mathcal{U}} F_{u^*}(u)$. Hence, u^* satisfies $\frac{\partial}{\partial u} F(u, u^*)\big|_{u=u^*} = 0$. Thus, u^* is a fixed point of upd_1 and upd_2. By Corollary A.4, there exists $0 < \beta < 1$ and $\eta > 0$ such that upd_1 is a β-contraction mapping. For any $u \in \mathcal{U}$, we have

$$\|\mathrm{upd}_1(u) - u^*\|_2 = \|\mathrm{upd}_1(u) - \mathrm{upd}_1(u^*)\|_2 \le \beta \|u - u^*\|_2.$$

Thus, by repeatedly applying the update rule i times, we get

$$\|(\mathrm{upd}_1)^i(u) - u^*\|_2 \le \beta^i \|u - u^*\|_2,$$

which converges to 0 as i goes infinity since $0 < \beta < 1$. Also, for the second update rule, we also have $0 < \beta < 1$ and $\eta > 0$ such that upd_2 is a β-contraction mapping by Corollary A.5. By the same logic, we have

$$\|(\mathrm{upd}_2)^i(u) - u^*\|_2 \le \beta^i \|u - u^*\|_2,$$

which converges to 0 as i goes infinity. Hence, for both update rules, we prove the convergence to u^*, finishing the proof. □

A.2. Proof of Lemma 3.3, Proposition 3.6, and Lemma 3.7

For the proof of these statements, we assume that the set-conditioned policy model has enough representative power to model the policies so that any policy that generate candidates given any set B is corresponding to π_θ^{set} for some $\theta \in \Theta$.

In Algorithm 2 and Algorithm 4, we inherently assume the unique maximizer at each step for ease of understanding. However, rigorously, there can be multiple maximizers at each step. In such cases, we assume that Algorithm 2 and Algorithm 4 select the candidate uniformly at random among the maximizers at each step. Consequently, there can be more than one possible solutions from Algorithm 2. To address this, we define *exact greedy solutions* as follows:

Definition A.6. A n-subset $B_n \subseteq \mathcal{X}$ is an exact greedy solution if there exists $\mathbf{x}_0, \dots, \mathbf{x}_{n-1} \in \mathcal{X}$ that satisfies

$$B_n = \{\mathbf{x}_0, \dots, \mathbf{x}_{n-1}\}, \text{ and } \forall\, 0 \le t \le n-1, \quad \mathbf{x}_t \in \operatorname*{argmax}_{\mathbf{x}' \in \mathcal{X}} \Delta_a(\mathbf{x}' \mid \{\mathbf{x}_0, \dots, \mathbf{x}_{t-1}\}),$$

where $\{\mathbf{x}_0, \dots, \mathbf{x}_{t-1}\}$ denotes the empty set $\emptyset$ if $t = 0$.

In other words, an exact greedy solution is an n-subset which can be sampled from Algorithm 2 with a positive probability. Now, we recall and prove Lemma 3.3 by contradiction as following.

Lemma A.7. $\mathrm{GS}(a, \pi_{\theta^*}^{set}, n, l)$ *samples exact greedy solutions almost surely if* π_{θ^*} *is the greedy policy.*

Proof. (Proof by contradiction.) Let $\pi_{\theta^*}^{\text{set}}$ be a greedy policy. According to Definition 3.2, we have

$$\forall\, \theta \in \Theta, \quad \mathcal{J}(\theta^*, \theta^*) \ge \mathcal{J}(\theta, \theta^*). \tag{4}$$

Now, assume the opposite of the desired conclusion, that with a positive probability, $\mathrm{GS}(a, \pi_{\theta^*}^{\text{set}}, n, l)$ generates $\mathbf{x}_0, \dots, \mathbf{x}_{n-1}$ sequentially by Algorithm 4 and results in a set $B_n = \{\mathbf{x}_0, \dots, \mathbf{x}_{n-1}\}$ that is not an exact greedy solution. Then, $\mathrm{GS}(a, \pi_{\theta^*}^{\text{set}}, n, 1)$ also generates $\mathbf{x}_0, \dots, \mathbf{x}_{n-1}$ with a positive probability. We denote this probability by $c_1 > 0$, *i.e.*,

$$c_1 \coloneqq \prod_{i=0}^{n-1} \pi_{\theta^*}^{\text{set}}(\mathbf{x}_i \mid \{\mathbf{x}_0, \dots, \mathbf{x}_{i-1}\}) > 0. \tag{5}$$

Since B_n is not an exact greedy solution, there exists at least one $0 \le t \le n-1$ such that

$$\mathbf{x}_t \notin \operatorname*{argmax}_{\mathbf{x}' \in \mathcal{X}} \Delta_a(\mathbf{x}' \mid \{\mathbf{x}_0, \dots, \mathbf{x}_{t-1}\}), \tag{6}$$

according to Definition A.6. For that t, let π_ϕ^{set} be the policy that replicates $\pi_{\theta^*}^{\text{set}}$ for all set $B \subset \mathcal{X}$ except for $\{\mathbf{x}_0, \dots, \mathbf{x}_{t-1}\}$, that satisfies

$$\forall \mathbf{x} \in \mathcal{X}, \quad \pi_\phi^{\text{set}}(\mathbf{x} \mid B) = \begin{cases} \pi_{\theta^*}^{\text{set}}(\mathbf{x} \mid B) & \text{if } B \ne \{\mathbf{x}_0, \dots, \mathbf{x}_{t-1}\}, \\ \mathbf{1}_{\{\mathbf{x}_t^*\}}(\mathbf{x}) & \text{if } B = \{\mathbf{x}_0, \dots, \mathbf{x}_{t-1}\}, \end{cases}$$

where $\mathbf{x}_t^*$ be any maximizer of the marginal gain at t-th step, *i.e.*, $\mathbf{x}_t^* \in \operatorname{argmax}_{\mathbf{x}' \in \mathcal{X}} \Delta_a(\mathbf{x}' \mid \{\mathbf{x}_0, \dots, \mathbf{x}_{t-1}\})$. Next, we define the difference in expected return between ϕ and θ^* given a set B as following:

$$C(B) \coloneqq \mathbb{E}_{\mathbf{x} \sim \pi_\phi^{\text{set}}(\cdot \mid B)}[\Delta_a(\mathbf{x} \mid B)] - \mathbb{E}_{\mathbf{x} \sim \pi_{\theta^*}^{\text{set}}(\cdot \mid B)}[\Delta_a(\mathbf{x} \mid B)].$$

If $B \ne \{\mathbf{x}_0, \dots, \mathbf{x}_{t-1}\}$, $\pi_\phi^{\text{set}}(\mathbf{x} \mid B) = \pi_{\theta^*}^{\text{set}}(\mathbf{x} \mid B)$ for all $\mathbf{x} \in \mathcal{X}$. Hence, we have a zero difference $C(B) = 0$.

If $B = \{\mathbf{x}_0, \dots, \mathbf{x}_{t-1}\}$, $\pi_\phi^{\text{set}}(\mathbf{x} \mid B) = \mathbf{1}_{\{\mathbf{x}_t^*\}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} = \mathbf{x}_t^* \\ 0 & \text{otherwise} \end{cases}$ for all $\mathbf{x} \in \mathcal{X}$. Hence, we have

$$\begin{aligned}
c_2 &\coloneqq C(B) \\
&= \Delta_a(\mathbf{x}_t^* \mid B) - \mathbb{E}_{\mathbf{x} \sim \pi_{\theta^*}^{\text{set}}(\cdot \mid B)}[\Delta_a(\mathbf{x} \mid B)] \\
&= \left(\max_{\mathbf{x}' \in \mathcal{X}} \Delta_a(\mathbf{x}' \mid B)\right) - \mathbb{E}_{\mathbf{x} \sim \pi_{\theta^*}^{\text{set}}(\cdot \mid B)}[\Delta_a(\mathbf{x} \mid B)] && \left(\because \mathbf{x}_t^* \in \operatorname*{argmax}_{\mathbf{x}' \in \mathcal{X}} \Delta_a(\mathbf{x}' \mid B)\right) \\
&= \mathbb{E}_{\mathbf{x} \sim \pi_{\theta^*}^{\text{set}}(\cdot \mid B)}\left[\left(\max_{\mathbf{x}' \in \mathcal{X}} \Delta_a(\mathbf{x}' \mid B)\right) - \Delta_a(\mathbf{x} \mid B)\right] && \left(\because \left(\max_{\mathbf{x}' \in \mathcal{X}} \Delta_a(\mathbf{x}' \mid B)\right) \text{ is a constant given } B\right) \\
&\ge \pi_{\theta^*}^{\text{set}}(\mathbf{x}_t \mid B)\left[\left(\max_{\mathbf{x}' \in \mathcal{X}} \Delta_a(\mathbf{x}' \mid B)\right) - \Delta_a(\mathbf{x}_t \mid B)\right]. && \left(\because \forall \mathbf{x} \in \mathcal{X}, \left(\max_{\mathbf{x}' \in \mathcal{X}} \Delta_a(\mathbf{x}' \mid B)\right) - \Delta_a(\mathbf{x} \mid B) \ge 0\right)
\end{aligned} \tag{7}$$

Here,

$$\pi^{\text{set}}_{\theta^*}(\mathbf{x}_t \mid B) = \pi^{\text{set}}_{\theta^*}(\mathbf{x}_t \mid \{\mathbf{x}_0, \dots, \mathbf{x}_{t-1}\}) \geq \prod_{i=0}^{n-1} \pi^{\text{set}}_{\theta^*}(\mathbf{x}_i \mid \{\mathbf{x}_0, \dots, \mathbf{x}_{i-1}\}) = c_1 > 0. \qquad (\because \text{Equation (5)}) \tag{8}$$

Moreover,

$$\left(\max_{\mathbf{x}' \in \mathcal{X}} \Delta_a(\mathbf{x}' \mid B)\right) - \Delta_a(\mathbf{x}_t \mid B) > 0. \qquad (\because \text{Equation (6)}) \tag{9}$$

By combining Equations (7), (8), and (9), we finally have $c_2 > 0$. As a result, we have

$$C(B) = \begin{cases} 0 & \text{if } B \neq \{\mathbf{x}_0, \dots, \mathbf{x}_{t-1}\}, \\ c_2 > 0 & \text{if } B = \{\mathbf{x}_0, \dots, \mathbf{x}_{t-1}\}. \end{cases}$$

Using this,

$$\begin{aligned}
&\mathcal{J}(\phi, \theta^*) - \mathcal{J}(\theta^*, \theta^*) \\
&\quad= \frac{1}{n}\sum_{k=0}^{n-1} \mathbb{E}_{B \sim \text{GS}(a, \pi^{\text{set}}_{\theta^*}, k, 1)}[\mathbb{E}_{\mathbf{x} \sim \pi^{\text{set}}_{\phi}(\cdot \mid B)}[\Delta_a(\mathbf{x} \mid B)]] - \frac{1}{n}\sum_{k=0}^{n-1} \mathbb{E}_{B \sim \text{GS}(a, \pi^{\text{set}}_{\theta^*}, k, 1)}[\mathbb{E}_{\mathbf{x} \sim \pi^{\text{set}}_{\theta^*}(\cdot \mid B)}[\Delta_a(\mathbf{x} \mid B)]] \\
&\quad= \frac{1}{n}\sum_{k=0}^{n-1} \mathbb{E}_{B \sim \text{GS}(a, \pi^{\text{set}}_{\theta^*}, k, 1)}[C(B)] \\
&\quad= \frac{1}{n} \underbrace{(\text{GS}(a, \pi^{\text{set}}_{\theta^*}, t, 1))(\{\mathbf{x}_0, \dots, \mathbf{x}_{t-1}\})}_{\text{Probability of sampling } \{\mathbf{x}_0, \dots \mathbf{x}_{t-1}\}} c_2 \\
&\quad\geq \frac{1}{n} \underbrace{\prod_{i=0}^{t-1} \pi^{\text{set}}_{\theta^*}(\mathbf{x}_i \mid \{\mathbf{x}_0, \dots, \mathbf{x}_{i-1}\})}_{\text{Probability of sampling } \mathbf{x}_0 \to \cdots \to \mathbf{x}_{t-1}} c_2 \geq \frac{1}{n} \underbrace{\prod_{i=0}^{n-1} \pi^{\text{set}}_{\theta^*}(\mathbf{x}_i \mid \{\mathbf{x}_0, \dots, \mathbf{x}_{i-1}\})}_{\text{Probability of sampling } \mathbf{x}_0 \to \cdots \to \mathbf{x}_{n-1}} c_2 = \frac{c_1 c_2}{n} > 0.
\end{aligned}$$

Thus, we finally have $\mathcal{J}(\phi, \theta^*) > \mathcal{J}(\theta^*, \theta^*)$ which contradicts to Equation (4). In conclusion, we complete the proof by contradiction. □

Next, we recall and prove Proposition 3.6.

Proposition A.8. *(Policy gradient for $\mathcal{J}$) For any baseline set function $b : 2^{\mathcal{X}} \to \mathbb{R}$ and the number of episodes N_e,*

$$\hat{g} = \frac{1}{N_e} \sum_{j=0}^{N_e - 1} (\Delta_a(\mathbf{x}^{(j)} \mid B) - b(B)) \nabla_\theta \pi^{set}_\theta(\mathbf{x}^{(j)} \mid B),$$

is an unbiased MC estimator of the partial derivative $\frac{\partial}{\partial\theta}\mathcal{J}(\theta, \theta')$ where $B \sim \frac{1}{n}\sum_{k=0}^{n-1} \text{GS}(a, \pi^{set}_{\theta'}, k, 1)$.

Proof. This proposition is a variation of policy gradients and our proof also takes the same idea of the log-derivative trick for obtaining MC estimator of the derivative (Williams, 1992). First of all, we can combine n expectations in $\mathcal{J}(\theta, \theta')$ by utilizing the mixture of distributions $\frac{1}{n}\sum_{k=0}^{n-1} \text{GS}(a, \pi^{\text{set}}_{\theta'}, k, 1)$ as follows:

$$\begin{aligned}
\mathcal{J}(\theta, \theta') &= \frac{1}{n}\sum_{k=0}^{n-1} \mathbb{E}_{B \sim \text{GS}(a, \pi^{\text{set}}_{\theta'}, k, 1)}[\mathbb{E}_{\mathbf{x} \sim \pi^{\text{set}}_{\theta}(\cdot \mid B)}[\Delta_a(\mathbf{x} \mid B)]] \\
&= \mathbb{E}_{B \sim \frac{1}{n}\sum_{k=0}^{n-1} \text{GS}(a, \pi^{\text{set}}_{\theta'}, k, 1)}[\mathbb{E}_{\mathbf{x} \sim \pi^{\text{set}}_{\theta}(\cdot \mid B)}[\Delta_a(\mathbf{x} \mid B)]].
\end{aligned}$$

Since the mixture of the distributions $\frac{1}{n}\sum_{k=0}^{n-1}\mathrm{GS}(a, \pi^{\text{set}}_{\theta'}, k, 1)$ is independent to θ, we denote this distribution by p for simplicity. Then, our goal is to obtain the estimator of partial derivative:

$$\begin{aligned}
\frac{\partial}{\partial\theta}\mathcal{J}(\theta,\theta') &= \frac{\partial}{\partial\theta}\mathbb{E}_{B\sim p}[\mathbb{E}_{\mathbf{x}\sim\pi^{\text{set}}_{\theta}(\cdot|B)}[\Delta_a(\mathbf{x}\mid B)]] \\
&= \frac{\partial}{\partial\theta}\left(\sum_{B\subset\mathcal{X},|B|\leq n} p(B)\sum_{\mathbf{x}\in\mathcal{X}}\Delta_a(\mathbf{x}\mid B)\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B)\right) \\
&= \sum_{B\subset\mathcal{X},|B|\leq n} p(B)\left(\sum_{\mathbf{x}\in\mathcal{X}}\Delta_a(\mathbf{x}\mid B)\nabla_\theta\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B)\right) \\
&= \sum_{B\subset\mathcal{X},|B|\leq n} p(B)\left(\sum_{\mathbf{x}\in\mathcal{X}}\Delta_a(\mathbf{x}\mid B)\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B)\nabla_\theta\log\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B)\right) \qquad \left(\because \nabla_\theta\log\pi^{\text{set}}_{\theta} = \frac{\nabla_\theta\pi^{\text{set}}_{\theta}}{\pi^{\text{set}}_{\theta}}\right) \\
&= \sum_{B\subset\mathcal{X},|B|\leq n} p(B)\mathbb{E}_{\mathbf{x}\sim\pi^{\text{set}}_{\theta}(\cdot|B)}\left[\Delta_a(\mathbf{x}\mid B)\nabla_\theta\log\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B)\right] \\
&= \mathbb{E}_{B\sim p}\left[\mathbb{E}_{\mathbf{x}\sim\pi^{\text{set}}_{\theta}(\cdot|B)}\left[\Delta_a(\mathbf{x}\mid B)\nabla_\theta\log\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B)\right]\right].
\end{aligned} \tag{10}$$

For the next step, we prove that following equality holds for any baseline set function $b: 2^{\mathcal{X}}\to\mathbb{R}$:

$$\mathbb{E}_{B\sim p}\left[\mathbb{E}_{\mathbf{x}\sim\pi^{\text{set}}_{\theta}(\cdot|B)}\left[b(B)\nabla_\theta\log\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B)\right]\right] = 0. \tag{11}$$

For any $B\subset\mathcal{X}$, we have

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}\sim\pi^{\text{set}}_{\theta}(\cdot|B)}[b(B)\nabla_\theta\log\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B)] &= b(B)\mathbb{E}_{\mathbf{x}\sim\pi^{\text{set}}_{\theta}(\cdot|B)}[\nabla_\theta\log\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B)] \\
&= b(B)\sum_{\mathbf{x}\in\mathcal{X}}\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B)\nabla_\theta\log\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B) \\
&= b(B)\sum_{\mathbf{x}\in\mathcal{X}}\nabla_\theta\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B) \\
&= b(B)\nabla_\theta\left(\sum_{\mathbf{x}\in\mathcal{X}}\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B)\right) = b(B)\nabla_\theta(1) = 0.
\end{aligned}$$

Thus, we get Equation (11). By combining Equation (10) and Equation (11), we finally achieve

$$\frac{\partial}{\partial\theta}\mathcal{J}(\theta,\theta') = \mathbb{E}_{B\sim p}\left[\mathbb{E}_{\mathbf{x}\sim\pi^{\text{set}}_{\theta}(\cdot|B)}\left[(\Delta_a(\mathbf{x}\mid B)-b(B))\nabla_\theta\log\pi^{\text{set}}_{\theta}(\mathbf{x}\mid B)\right]\right].$$

Finally, we derive an unbiased MC estimator $\hat{g}$ of the parital derivative $\frac{\partial}{\partial\theta}\mathcal{J}(\theta,\theta')$ by

$$\hat{g} = \frac{1}{N_e}\sum_{j=0}^{N_e-1}(\Delta_a(\mathbf{x}^{(j)}\mid B)-b(B))\nabla_\theta\log\pi^{\text{set}}_{\theta}(\mathbf{x}^{(j)}\mid B),$$

where $B\sim p$ and $\mathbf{x}^{(j)}\sim\pi^{\text{set}}_{\theta}(\cdot\mid B)$. □

Finally, we recall and prove Lemma 3.7.

Lemma A.9. *For any $B, B'\subset\mathcal{X}$ satisfying $\tilde{f}(B)=\tilde{f}(B')$, $\Delta_a(\mathbf{x}\mid B)=\Delta_a(\mathbf{x}\mid B')$ for all $\mathbf{x}\in\mathcal{X}$ if a is given by HVI and $\tilde{f}$ is deterministic surrogate function.*

Proof. We prove the consequence directly from the definition of HVI as follows:

$$\begin{aligned}
\Delta_a(\mathbf{x}\mid B) = \mathbf{HVI}(B;\tilde{\mathbf{f}},\tilde{\mathcal{P}},\mathbf{r}_{\text{ref}}) &= \mathbf{HV}(\tilde{f}(B)\cup\tilde{f}(\tilde{\mathcal{P}});\mathbf{r}_{\text{ref}}) - \mathbf{HV}(\tilde{f}(\tilde{\mathcal{P}});\mathbf{r}_{\text{ref}}) \\
&= \mathbf{HV}(\tilde{f}(B')\cup\tilde{f}(\tilde{\mathcal{P}});\mathbf{r}_{\text{ref}}) - \mathbf{HV}(\tilde{f}(\tilde{\mathcal{P}});\mathbf{r}_{\text{ref}}) \\
&= \mathbf{HVI}(B';\tilde{\mathbf{f}},\tilde{\mathcal{P}},\mathbf{r}_{\text{ref}}) = \Delta_a(\mathbf{x}\mid B').
\end{aligned}$$

□

A.3. Detailed Explanation of Bounds for Approximated Greedy Algorithm

In this section, we provide a more formal explanation on bounds for approximated greedy algorithm proposed in Section 3.3. First, we introduce the *monotone submodularity*.

Definition A.10. (Monotone Submodularity) A real-valued set function $a : 2^{\mathcal{X}} \to \mathbb{R}$ is submodular if the inequality $\Delta_a(\mathbf{x} \mid B') \geq \Delta_a(\mathbf{x} \mid B)$ holds for all $B' \subset B \subset \mathcal{X}$ and $\mathbf{x} \in \mathcal{X} \setminus B$. Additionally, a set function a is monotone if $\Delta_a(\mathbf{x} \mid B) \geq 0$ for all $B \subset \mathcal{X}$ and $\mathbf{x} \in \mathcal{X} \setminus B$. Finally, a set function s is non-negative if $a(B) \geq 0$ for all $B \subset \mathcal{X}$.

In essence, a monotone submodularity is characterized by a consistently non-negative marginal gain that diminishes as the augmenting set enlarges. For a non-negative monotone submodular function a, the exact greedy algorithm is known to guarantee a $(1 - 1/e)$-approximation to the optimal n-subset B_n^*, *i.e.*, $a(B_n) \geq (1 - 1/e)a(B_n^*)$ (Nemhauser et al., 1978).

In Theorem 3.9, we extend this bound to the approximated greedy algorithm when a is any monotone near-submodular set function using the notion of *submodularity ratio*, which is formally defined as follows.

Definition A.11. (Submodularity Ratio) Let $a : 2^{\mathcal{X}} \to \mathbb{R}$ be a monotone set function. For $B \subset \mathcal{X}$ and $n \geq 1$, the submodularity ratio $\gamma_{B,n}(a)$ is defined as

$$\gamma_{B,n}(a) := \min_{S \subset \mathcal{X}, B' \subset B \setminus S, |S| \leq n} \frac{\sum_{\mathbf{x} \in S} \Delta_a(\mathbf{x} \mid B')}{a(B' \cup S) - a(B')} \leq 1,$$

where we define $0/0 := 1$ ($S = \emptyset$ case).

The submodularity ratio measures the extent to which the function a exhibits submodularity (Zhou & Spanos, 2016).

A.3.1. Proof of Theorem 3.9

To start, we recall Theorem 3.9.

Theorem A.12. *Let $a : 2^{\mathcal{X}} \to \mathbb{R}$ be a non-negative monotone set function. If $\mathcal{A}$ is an α-approximation algorithm, the resulting solution B_n of Algorithm 3 is an $(1 - 1/e^{\alpha \gamma_{B_n,n}(a)})$-approxmiation to the optimal n-subset B_n^*, i.e., $a(B_n) \geq (1 - 1/e^{\alpha \gamma_{B_n,n}(a)})a(B_n^*)$.*

Next, we summarize notations for the proof as follows:

Table 4: Notations for the proof of Theorem 3.9.

Notation	Definition
$\mathcal{A}$	an α-approximation algorithm.
$n \in \mathbb{N}$	a cardinality constraint.
$a : 2^{\mathcal{X}} \to \mathbb{R}$	a non-negative monotone objective set function.
$\mathbf{x}_i^{\mathcal{A}} \in \mathcal{X}$	the candidate appended at i-th step of Algorithm 3 with an algorithm $\mathcal{A}$.
$B_i^{\mathcal{A}} = \{\mathbf{x}_j^{\mathcal{A}} \mid 0 \leq j \leq i-1\} \subset \mathcal{X}$	the i-subset constructed by Algorithm 3 with an algorithm $\mathcal{A}$.
$B_i^* \in \operatorname{argmax}_{B \subset \mathcal{X}, \lvert B \rvert = i} a(B)$	the optimal i-subset of a.

Our proof is an extension of the proof by Das & Kempe (2018). Using the notion of the submodularity ratio, we first prove the following lemma.

Lemma A.13. *For any $0 \leq i < n$, the following inequality holds:*

$$\Delta_a(\mathbf{x}_i^{\mathcal{A}} \mid B_i^{\mathcal{A}}) \geq \frac{\alpha \gamma_{B_n^{\mathcal{A}},n}(a)}{n}(a(B_n^*) - a(B_i^{\mathcal{A}})).$$

Proof. Let $S_i \coloneqq B_n^* \setminus B_i^{\mathcal{A}}$. Then,

$$\begin{aligned}
\Delta_a(\mathbf{x}_i^{\mathcal{A}} \mid B_i^{\mathcal{A}}) &\geq \alpha \max_{\mathbf{x}\in\mathcal{X}} \Delta_a(\mathbf{x} \mid B_i^{\mathcal{A}}) && (\because \mathcal{A} \text{ is an } \alpha\text{-approximation algorithm.}) \\
&\geq \alpha \frac{\sum_{\mathbf{x}\in S_i} \Delta_a(\mathbf{x} \mid B_i^{\mathcal{A}})}{|S_i|} && (\because \text{maximality}) \\
&\geq \alpha \frac{\gamma_{B_n^{\mathcal{A}},n}(a)}{|S_i|}(a(B_i^{\mathcal{A}} \cup S_i) - a(B_i^{\mathcal{A}})) && (\because \text{Definition A.11}, B_i^{\mathcal{A}} \subset B_n^{\mathcal{A}} \setminus S_i, |S_i| \leq |B_n^*| = n) \\
&\geq \frac{\alpha\gamma_{B_n^{\mathcal{A}},n}(a)}{n}(a(B_i^{\mathcal{A}} \cup S_i) - a(B_i^{\mathcal{A}})) && (\because |S_i| \leq |B_n^*| = n) \\
&\geq \frac{\alpha\gamma_{B_n^{\mathcal{A}},n}(a)}{n}(a(B_n^*) - a(B_i^{\mathcal{A}})). && (\because B_n^* \subseteq S_i \cup B_i^{\mathcal{A}}, \text{monotonicity of } a)
\end{aligned}$$

□

For the next step, we prove the following lemma using Lemma A.13.

Lemma A.14. *For any $i \geq 0$, the following inequality holds:*

$$a(B_n^*) - a(B_{i+1}^{\mathcal{A}}) \leq \left(1 - \frac{\alpha\gamma_{B_n^{\mathcal{A}},n}(a)}{n}\right)(a(B_n^*) - a(B_i^{\mathcal{A}})).$$

Proof.

$$\begin{aligned}
a(B_n^*) - a(B_{i+1}^{\mathcal{A}}) &= a(B_n^*) - (a(B_i^{\mathcal{A}}) + \Delta_a(\mathbf{x}_i^{\mathcal{A}} \mid B_i^{\mathcal{A}})) \\
&\leq (a(B_n^*) - a(B_i^{\mathcal{A}})) - \frac{\alpha\gamma_{B_n^{\mathcal{A}},n}(a)}{n}(a(B_n^*) - a(B_i^{\mathcal{A}})) && (\because \text{Lemma A.13}) \\
&= \left(1 - \frac{\alpha\gamma_{B_n^{\mathcal{A}},n}(a)}{n}\right)(a(B_n^*) - a(B_i^{\mathcal{A}})).
\end{aligned}$$

□

Finally, we prove Theorem 3.9.

Proof of Theorem 3.9. By combining Lemma A.14 with $i = 0, \ldots, n-1$, we get

$$\begin{aligned}
a(B_n^*) - a(B_n^{\mathcal{A}}) &\leq \left(1 - \frac{\alpha\gamma_{B_n^{\mathcal{A}},n}(a)}{n}\right)^n (a(B_n^*) - a(B_0^{\mathcal{A}})) \\
&\leq \left(1 - \frac{\alpha\gamma_{B_n^{\mathcal{A}},n}(a)}{n}\right)^n a(B_n^*). && (\because \text{non-negativity of } a)
\end{aligned}$$

Finally, we get the desired bound as follows:

$$\begin{aligned}
a(B_n^{\mathcal{A}}) &\geq a(B_n^*) - \left(1 - \frac{\alpha\gamma_{B_n^{\mathcal{A}},n}(a)}{n}\right)^n a(B_n^*) \\
&= \left(1 - \left(1 - \frac{\alpha\gamma_{B_n^{\mathcal{A}},n}(a)}{n}\right)^n\right) a(B_n^*) \\
&\geq \left(1 - \frac{1}{e^{\alpha\gamma_{B_n^{\mathcal{A}},n}(a)}}\right) a(B_n^*). && (\because (1-1/t)^t \leq 1/e \text{ for any } t \geq 1)
\end{aligned}$$

□

A.3.2. Proof of Theorem 3.10

First, we recall Theorem 3.10.

Theorem A.15. *Let s be a non-negative monotone set function and* div *be a sum-dispersion function. If $\mathcal{A}$ is an α-approximation algorithm, Algorithm 3 with the set function $a = s/2 + \lambda\mathrm{div}$ returns B_n, an $(\alpha\hat{\gamma}/2)$-approximation to the optimal n-subset B_n^* of $(s + \lambda\mathrm{div})$, where $\hat{\gamma} := \gamma_{B_n^* \cup B_n, n}(s)$.*

Theorem 3.10 suggests a bound for non-oblivious variation of the approximated greedy algorithm which guide the algorithm with the set function $a = s/2 + \lambda\mathrm{div}$ that is different to the actual objective function $s + \lambda\mathrm{div}$ (Ward, 2012). Our bound extends the theoretical bound of the non-oblivious exact greedy algorithm proved by Borodin et al. (2012) when the objective set function is the sum of a submodular function and a sum-dispersion function. Our proof combines the ideas from Borodin et al. (2012) and Das & Kempe (2018). To start, we define some notations as following:

Table 5: Notations for the proof of Theorem 3.10.

Notation	Definition
$\mathcal{A}$	an α-approximation algorithm.
$n \in \mathbb{N}$	a cardinality constraint.
$s : 2^{\mathcal{X}} \to \mathbb{R}$	a non-negative monotone set function.
$\mathrm{div} : 2^{\mathcal{X}} \to \mathbb{R}$	a dispersion function defined as $\mathrm{div}(B) = \frac{1}{2}\sum_{\mathbf{x}\in B}\sum_{\mathbf{x}'\in B} d(\mathbf{x}, \mathbf{x}')$ for a metric d.
$\lambda > 0$	a real-valued coefficient that controlls tradeoff between s and div.
$a = s + \lambda\mathrm{div}$	the actual objective set function to optimize.
$\tilde{a} = s/2 + \lambda\mathrm{div}$	the set function to optimize during subproblems of Algorithm 3.
$\mathbf{x}_i^{\mathcal{A}} \in \mathcal{X}$	the candidate appended at i-th step of Algorithm 3 with $\tilde{a} = \frac{1}{2}s + \lambda\mathrm{div}$ and $\mathcal{A}$.
$B_i^{\mathcal{A}} = \{\mathbf{x}_j^{\mathcal{A}} \mid 0 \le j \le i-1\} \subset \mathcal{X}$	the i-subset constructed by Algorithm 3 with $\tilde{a} = \frac{1}{2}s + \lambda\mathrm{div}$ and $\mathcal{A}$.
$B_i^* \in \mathrm{argmax}_{B\subset\mathcal{X}, \lvert B\rvert = i}\, a(B)$	the optimal i-subset of $a = s + \lambda\mathrm{div}$.
$\hat{\gamma} := \gamma_{B_n^* \cup B_n^{\mathcal{A}}, n}(s)$	the submodularity index.

For notational simplicity, we define $d(B, B') := \sum_{\mathbf{x}\in B}\sum_{\mathbf{x}'\in B'} d(\mathbf{x}, \mathbf{x}')$ for any $B, B' \subset \mathcal{X}$. Then, $\mathrm{div}(B) = \frac{1}{2}d(B, B)$. We introduce two lemmas from the previous works (Ravi et al., 1994; Borodin et al., 2012). For the completeness, we contain the proof of these lemmas.

Lemma A.16. *(Ravi et al., 1994) For a given metric function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and two disjoint sets $B, B' \subset \mathcal{X}$, we have the following inequality: $(\lvert B'\rvert - 1)d(B, B') \ge \lvert B\rvert\mathrm{div}(B')$.*

Proof.

$$\begin{aligned}
\lvert B\rvert\mathrm{div}(B') &= \frac{1}{2}\lvert B\rvert d(B', B') \\
&= \frac{1}{2}\sum_{\mathbf{x}\in B}\sum_{\mathbf{x}'\in B'}\sum_{\mathbf{x}''\in B'} d(\mathbf{x}', \mathbf{x}'') \\
&= \frac{1}{2}\sum_{\mathbf{x}\in B}\sum_{\mathbf{x}'\in B'}\sum_{\mathbf{x}''\in B'\setminus\{\mathbf{x}'\}} d(\mathbf{x}', \mathbf{x}'') && (\because d(\mathbf{x}', \mathbf{x}') = 0) \\
&\le \frac{1}{2}\sum_{\mathbf{x}\in B}\sum_{\mathbf{x}'\in B'}\sum_{\mathbf{x}''\in B'\setminus\{\mathbf{x}'\}} (d(\mathbf{x}, \mathbf{x}') + d(\mathbf{x}, \mathbf{x}'')) && (\because \text{triangle inequality}) \\
&= \frac{1}{2}\sum_{\mathbf{x}\in B}\sum_{\mathbf{x}'\in B'}\sum_{\mathbf{x}''\in B'\setminus\{\mathbf{x}'\}} d(\mathbf{x}, \mathbf{x}') + \frac{1}{2}\sum_{\mathbf{x}\in B}\sum_{\mathbf{x}''\in B'}\sum_{\mathbf{x}'\in B'\setminus\{\mathbf{x}''\}} d(\mathbf{x}, \mathbf{x}'') \\
&= \frac{\lvert B'\rvert - 1}{2}\sum_{\mathbf{x}\in B}\sum_{\mathbf{x}'\in B'} d(\mathbf{x}, \mathbf{x}') + \frac{\lvert B'\rvert - 1}{2}\sum_{\mathbf{x}\in B}\sum_{\mathbf{x}''\in B'} d(\mathbf{x}, \mathbf{x}'') \\
&= (\lvert B'\rvert - 1)\sum_{\mathbf{x}\in B}\sum_{\mathbf{x}'\in B'} d(\mathbf{x}, \mathbf{x}') = (\lvert B'\rvert - 1)d(B, B').
\end{aligned}$$

□

Lemma A.17. *(Borodin et al., 2012) For $1 \le i \le n$, let $U = B_n^* \cap B_i^{\mathcal{A}}$, $V = B_i^{\mathcal{A}} - U$, and $W = B_n^* - U$. If $|W| > 1$, we have the following inequality:*

$$d(B_i^{\mathcal{A}}, W) \ge \frac{i|W|}{n(n-1)} \mathrm{div}(B_n^*).$$

Proof. Using Lemma A.16,

$$(|W|-1)d(V,W) \ge |V| \mathrm{div}(W), \tag{12}$$
$$(|W|-1)d(U,W) \ge |U| \mathrm{div}(W), \tag{13}$$
$$(|U|-1)d(U,W) \ge |W| \mathrm{div}(U). \tag{14}$$

Also,

$$\begin{aligned}
\mathrm{div}(B_n^*) &= \frac{1}{2}\sum_{\mathbf{x}\in B_n^*}\sum_{\mathbf{x}'\in B_n^*} d(\mathbf{x},\mathbf{x}') \\
&= \frac{1}{2}\sum_{\mathbf{x}\in U\cup W}\sum_{\mathbf{x}'\in U\cup W} d(\mathbf{x},\mathbf{x}') && (\because B_n^* = U\cup W) \\
&= \frac{1}{2}\left(\sum_{\mathbf{x}\in U}\sum_{\mathbf{x}'\in U} d(\mathbf{x},\mathbf{x}')\right) + \left(\sum_{\mathbf{x}\in U}\sum_{\mathbf{x}'\in W} d(\mathbf{x},\mathbf{x}')\right) + \frac{1}{2}\left(\sum_{\mathbf{x}\in W}\sum_{\mathbf{x}'\in W} d(\mathbf{x},\mathbf{x}')\right) \\
&= \mathrm{div}(U) + d(U,W) + \mathrm{div}(W).
\end{aligned} \tag{15}$$

By combining four equations as

$$\text{Equation (12)} \times \frac{1}{|W|-1} + \text{Equation (13)} \times \frac{|W|-|V|}{n(|W|-1)} + \text{Equation (14)} \times \frac{i}{n(n-1)} + \text{Equation (15)} \times \frac{i|W|}{n(n-1)},$$

we have the following inequality:

$$d(U,W) + d(V,W) - \frac{i|W|(n-|W|)}{n(n-1)(|W|-1)}\mathrm{div}(W) \ge \frac{i|W|}{n(n-1)}\mathrm{div}(B_n^*). \tag{16}$$

Hence, we have the desired result as follows:

$$\begin{aligned}
d(B_i^{\mathcal{A}}, W) &= d(U\cup V, W) \\
&= d(U,W) + d(V,W) && (\because U\cap V = \emptyset) \\
&\ge d(U,W) + d(V,W) - \frac{i|W|(n-|W|)}{n(n-1)(|W|-1)}\mathrm{div}(W) && (\because 1 < |W| \le |B_n^*| = n) \\
&\ge \frac{i|W|}{n(n-1)}\mathrm{div}(B_n^*). && (\because \text{Equation (16)})
\end{aligned}$$

□

For the convenience, we introduce the following lemma:

Lemma A.18. *For any $B \subset \mathcal{X}$ and $\mathbf{x} \in \mathcal{X}$, the following inequality holds: $\frac{1}{2}\Delta_a(\mathbf{x} \mid B) \le \Delta_{\tilde{a}}(\mathbf{x} \mid B) \le \Delta_a(\mathbf{x} \mid B)$.*

Proof.

$$\begin{aligned}
\Delta_{\tilde{a}}(\mathbf{x} \mid B) = \frac{1}{2}\Delta_s(\mathbf{x} \mid B) + \lambda\Delta_{\mathrm{div}}(\mathbf{x} \mid B) &\le \Delta_s(\mathbf{x} \mid B) + \lambda\Delta_{\mathrm{div}}(\mathbf{x} \mid B) && (\because \text{monotoniciy of } s) \\
&= \Delta_a(\mathbf{x} \mid B),
\end{aligned}$$

$$\begin{aligned}
\Delta_{\tilde{a}}(\mathbf{x} \mid B) = \frac{1}{2}\Delta_s(\mathbf{x} \mid B) + \lambda\Delta_{\mathrm{div}}(\mathbf{x} \mid B) &\ge \frac{1}{2}\left(\Delta_s(\mathbf{x} \mid B) + \lambda\Delta_{\mathrm{div}}(\mathbf{x} \mid B)\right) && (\because \text{monotoniciy of div}) \\
&= \frac{1}{2}\Delta_a(\mathbf{x} \mid B).
\end{aligned}$$

□

Proof of Theorem 3.10.
(Case 1: $n = 1$**)**
Let $\mathbf{x}^\dagger \in \mathcal{X}$ be the maximizer of $\Delta_{\tilde{a}}(\cdot \mid \emptyset)$ and $\mathbf{x}^* \in \mathcal{X}$ be the maximizer of $\Delta_a(\cdot \mid \emptyset)$, *i.e.*$\{\mathbf{x}^*\} = B_1^*$. Then, we have

$$\begin{aligned}
a(B_1^{\mathcal{A}}) = a(\{\mathbf{x}_0^{\mathcal{A}}\}) &= s(\{\mathbf{x}_0^{\mathcal{A}}\}) + \lambda \mathrm{div}(\{\mathbf{x}_0^{\mathcal{A}}\}) \\
&\geq \frac{1}{2} s(\{\mathbf{x}_0^{\mathcal{A}}\}) + \lambda \mathrm{div}(\{\mathbf{x}_0^{\mathcal{A}}\}) && (\because \text{ non-negativity of } s) \\
&= \tilde{a}(\{\mathbf{x}_0^{\mathcal{A}}\}) \\
&= \Delta_{\tilde{a}}(\mathbf{x}_0^{\mathcal{A}} \mid \emptyset) \\
&\geq \alpha \Delta_{\tilde{a}}(\mathbf{x}^\dagger \mid \emptyset) && (\because \mathbf{x}_0^{\mathcal{A}} \text{ is an } \alpha\text{-approximation}) \\
&\geq \alpha \Delta_{\tilde{a}}(\mathbf{x}^* \mid \emptyset) && (\because \text{ optimality of } \mathbf{x}^\dagger) \\
&= \alpha \left(\frac{1}{2} s(\{\mathbf{x}^*\}) + \mathrm{div}(\{\mathbf{x}^*\}) \right) \\
&\geq \frac{\alpha}{2} (s(\{\mathbf{x}^*\}) + \mathrm{div}(\{\mathbf{x}^*\})) && (\because \text{ non-negativity of div}) \\
&= \frac{\alpha}{2} a(\{\mathbf{x}^*\}) = \frac{\alpha}{2} a(B_1^*) \geq \frac{\alpha\hat{\gamma}}{2} a(B_1^*).
\end{aligned}$$

(**Case 2**: $n > 1$)
For any $1 \leq i < n$, let $U = B_n^* \cap B_i^{\mathcal{A}}$, $V = B_i^{\mathcal{A}} - U$, and $W = B_n^* - U$ as in Lemma A.16.
(**Case 2.a**: $n > 1$ and $|W| = 1$)
In this case, we have $i = n - 1$ and $B_i^{\mathcal{A}} \subset B_n^*$ since $i < n$. Let $\mathbf{x}^* \in B_{n-1}^{\mathcal{A}}$ be the element that is not in B_n^*, *i.e.*, $\{\mathbf{x}^*\} = B_n^* \setminus B_{n-1}^{\mathcal{A}}$. Let $\mathbf{x}^\dagger \in \mathcal{X} \setminus B_{n-1}^{\mathcal{A}}$ be the maximizer of $\Delta_{\tilde{a}}(\cdot \mid B_{n-1}^{\mathcal{A}})$. Then,

$$\begin{aligned}
a(B_n^{\mathcal{A}}) &= a(B_{n-1}^{\mathcal{A}}) + \Delta_a(\mathbf{x}_{n-1}^{\mathcal{A}} \mid B_{n-1}^{\mathcal{A}}) && (\because B_n^{\mathcal{A}} = B_{n-1}^{\mathcal{A}} \cup \{\mathbf{x}_{n-1}^{\mathcal{A}}\}) \\
&\geq a(B_{n-1}^{\mathcal{A}}) + \alpha \Delta_a(\mathbf{x}^\dagger \mid B_{n-1}^{\mathcal{A}}) && (\because \mathbf{x}_{n-1}^{\mathcal{A}} \text{ is an } \alpha\text{-approximation}) \\
&\geq a(B_{n-1}^{\mathcal{A}}) + \alpha \Delta_{\tilde{a}}(\mathbf{x}^\dagger \mid B_{n-1}^{\mathcal{A}}) && (\because \text{ Lemma A.18}) \\
&\geq a(B_{n-1}^{\mathcal{A}}) + \alpha \Delta_{\tilde{a}}(\mathbf{x}^* \mid B_{n-1}^{\mathcal{A}}) && (\because \text{ optimality of } \mathbf{x}^\dagger) \\
&\geq a(B_{n-1}^{\mathcal{A}}) + \frac{\alpha}{2} \Delta_a(\mathbf{x}^* \mid B_{n-1}^{\mathcal{A}}) && (\because \text{ Lemma A.18}) \\
&\geq \frac{\alpha}{2} (a(B_{n-1}^{\mathcal{A}}) + \Delta_a(\mathbf{x}^* \mid B_{n-1}^{\mathcal{A}})) && (\because \text{ non-negativity of } a) \\
&= \frac{\alpha}{2} a(B_n^*) \geq \frac{\alpha\hat{\gamma}}{2} a(B_n^*). && (\because \text{ Definition A.11})
\end{aligned}$$

(**Case 2.b**: $n > 1$ and $|W| > 1$)
Now we can consider the case that $n > 1$ and $|W| > 1$. Using Lemma A.17, we have

$$d(B_i^{\mathcal{A}}, W) \geq \frac{i|W|}{n(n-1)} \mathrm{div}(B_n^*). \tag{17}$$

Using the monotonicity of s and the fact that $B_i^{\mathcal{A}} \cup W \subset B_i^{\mathcal{A}} \cup B_n^* \subset B_n^{\mathcal{A}} \cup B_n^*$, $|W| \leq |B_n^*| = n$, and $W \cap B_i^{\mathcal{A}} = \emptyset$ with Definition A.11, we have

$$\sum_{\mathbf{x} \in W} \Delta_s(\mathbf{x} \mid B_i^{\mathcal{A}}) \geq \hat{\gamma} \left(s(B_i^{\mathcal{A}} \cup W) - s(B_i^{\mathcal{A}}) \right) \geq \hat{\gamma}(s(B_n^*) - s(B_n^{\mathcal{A}})). \tag{18}$$

Thus,

$$\begin{aligned}
\sum_{\mathbf{x} \in W} \Delta_{\tilde{a}}(\mathbf{x} \mid B_i^{\mathcal{A}}) &= \sum_{\mathbf{x} \in W} \left(\frac{1}{2} \Delta_s(\mathbf{x} \mid B_i^{\mathcal{A}}) + \lambda d(\mathbf{x}, B_i^{\mathcal{A}}) \right) \\
&= \sum_{\mathbf{x} \in W} \frac{1}{2} \Delta_s(\mathbf{x} \mid B_i^{\mathcal{A}}) + \lambda d(W, B_i^{\mathcal{A}}) \\
&\geq \frac{\hat{\gamma}}{2} (s(B_n^*) - s(B_n^{\mathcal{A}})) + \frac{i\lambda|W|}{n(n-1)} \mathrm{div}(B_n^*). \qquad (\because \text{ Equation (17) and Equation (18)})
\end{aligned} \tag{19}$$

By utilizing the previous inequality, we have

$$\begin{aligned}\Delta_{\tilde{a}}(\mathbf{x}_i^{\mathcal{A}} \mid B_i^{\mathcal{A}}) &\geq \alpha \max_{\mathbf{x}\in\mathcal{X}\setminus B_i^{\mathcal{A}}} \Delta_{\tilde{a}}(\mathbf{x} \mid B_i^{\mathcal{A}}) && (\because \mathbf{x}_i^{\mathcal{A}} \text{ is an } \alpha\text{-approximation})\\ &\geq \frac{\alpha}{|W|}\sum_{\mathbf{x}\in W}\Delta_{\tilde{a}}(\mathbf{x} \mid B_i^{\mathcal{A}})\\ &\geq \frac{\alpha\hat{\gamma}}{2|W|}(s(B_n^*) - s(B_n^{\mathcal{A}})) + \frac{i\alpha\lambda}{n(n-1)}\mathrm{div}(B_n^*) && (\because \text{Equation (19)})\\ &\geq \frac{\alpha\hat{\gamma}}{2n}(s(B_n^*) - s(B_n^{\mathcal{A}})) + \frac{i\alpha\lambda}{n(n-1)}\mathrm{div}(B_n^*). && (\because |W| \leq |B_n^*| = n)\end{aligned}$$

By summing the inequality above for all i from 0 to $n-1$, we have

$$\begin{aligned}\frac{1}{2}s(B_n^{\mathcal{A}}) + \lambda\mathrm{div}(B_n^{\mathcal{A}}) = \tilde{a}(B_n^{\mathcal{A}}) &= \sum_{i=0}^{n-1}\Delta_{\tilde{a}}(\mathbf{x}_i^{\mathcal{A}} \mid B_i^{\mathcal{A}})\\ &\geq \frac{\alpha\hat{\gamma}}{2}(s(B_n^*) - s(B_n^{\mathcal{A}})) + \frac{\alpha\lambda}{2}\mathrm{div}(B_n^*)\\ &\geq \frac{\alpha\hat{\gamma}}{2}(s(B_n^*) - s(B_n^{\mathcal{A}}) + \lambda\mathrm{div}(B_n^*)). && (\because \hat{\gamma} \leq 1 \text{ and non-negativity of div})\end{aligned}$$

Hence,

$$\frac{1+\alpha\hat{\gamma}}{2}s(B_n^{\mathcal{A}}) + \lambda\mathrm{div}(B_n^{\mathcal{A}}) \geq \frac{\alpha\hat{\gamma}}{2}(s(B_n^*) + \lambda\mathrm{div}(B_n^*)).$$

Finally, we have

$$\begin{aligned}a(B_n^{\mathcal{A}}) = s(B_n^{\mathcal{A}}) + \lambda\mathrm{div}(B_n^{\mathcal{A}}) &\geq \frac{1+\hat{\gamma}\alpha}{2}s(B_n^{\mathcal{A}}) + \lambda\mathrm{div}(B_n^{\mathcal{A}}) && (\because \hat{\gamma}, \alpha \leq 1)\\ &\geq \frac{\alpha\hat{\gamma}}{2}(s(B_n^*) + \lambda\mathrm{div}(B_n^*)) = \frac{\alpha\hat{\gamma}}{2}a(B_n^*).\end{aligned}$$

□

A.3.3. Connection to Prior Bounds

Table 6: Bounds for the exact greedy algorithm and the approximated greedy algorithm.

	Exact Greedy Algorithm		Approximated Greedy Algorithm	
Condition on Acquisition	w/o diversity	w/ diversity	w/o diversity	w/ diversity
Submodular	$1-1/e$ (Nemhauser et al., 1978)	$1/2$ (Borodin et al., 2012)	$1-(1/e)^{\alpha}$ (Goundan & Schulz, 2007)	$\alpha/2$ (Theorem 3.10, $\gamma=1$)
Near-submodular	$1-(1/e)^{\gamma}$ (Das & Kempe, 2018)	$\gamma/2$ (Theorem 3.10, $\alpha=1$)	$1-(1/e)^{\alpha\gamma}$ (Theorem 3.9)	$\alpha\gamma/2$ (Theorem 3.10)

Since $\gamma_{B,n}(a) = 1$ for any B, n when a is monotone submodular, Theorem 3.9 directly contains a bound for the monotone submodular case (Corollary A.19) proved by Goundan & Schulz (2007).

Corollary A.19. *Let $a : \mathcal{X} \to \mathbb{R}$ be a non-negative monotone submodular set function. If $\mathcal{A}$ is an α-approximation algorithm, Algorithm 3 returns an $(1-1/e^{\alpha})$-approximation to the optimal n-subset.*

Similarly, Theorem 3.10 directly contains the following corollary.

Corollary A.20. *Let s be a non-negative monotone submodular set function and* div *be a sum-dispersion function. If $\mathcal{A}$ is an α-approximation algorithm, Algorithm 3 with the set function $(s/2 + \lambda\mathrm{div})$ returns B_n, an $(\alpha/2)$-approximation to the optimal n-subset of $(s + \lambda\mathrm{div})$.*

Note that the $\alpha = 1$ case of Corollary A.20 is the same as the bound for the exact greedy algorithm proved by Borodin et al. (2012). Finally, Table 6 summarizes the prior bounds and new bounds for the exact greedy algorithm and the approximated greedy algorithm for various conditions on the set function.

B. Implementation Details

B.1. MDP Designs

This paper considers the benchmark tasks on sequence data such as proteins and aptamers. There are a variety of MDP designs to model the sequence data as MDPs (Shen et al., 2023). In this paper, we consider two designs, *appending MDP* and *editing MDP* of MDPs following prior works (Jain et al., 2023). First, appending MDP designs any sequence from scratch. Assume a simple scenario that a search space is given by the fixed length sequence space $\mathcal{X} = \mathcal{V}^L$ is given by its length L and a vocabulary set $\mathcal{V}$. For this space, we can write the state space and the action space as follows:

$$\mathcal{S} = \bigcup_{i=0}^{L} \mathcal{V}^L \cup \{s_{\text{term}}\}, \mathcal{A} = \mathcal{V} \cup \{a_{\text{term}}\}.$$

Here, the initial state is always given as an empty sequence, and the MDP has a deterministic transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$. Concretely,

$$\mathcal{T}(s, a) = \begin{cases} s \oplus a & \text{if } a \in \mathcal{V} \text{ and } \text{len}(s) \le L - 1 \ , \\ s_{\text{term}} & \text{otherwise.} \end{cases}$$

In other words, action $a \in \mathcal{A}$ appends the chosen token to the current state $s \leftarrow s \oplus a$ or terminate the construction $s \leftarrow s_{\text{term}}$. As introduced in Section 2.4, a reward is given at the terminal state by the objective value of a resulting sequence. Hence, we do not distinguish the terms rewards and returns in this scenario. In this MDP design, a generic policy model $\pi_\theta : \mathcal{S} \to \mathcal{A}$ outputs an action distribution (token distribution) given state s (subsequence).

Otherwise, editing MDP designs any sequence by editing a sequence in the given pool of candidates. Hence, an action corresponds to an edit operation on the sequence. As in Jain et al. (2023), we concentrate on substitution operations for the action space. In short, the state space is a possible set of sequences obtained by editing a given pool of candidates. The action space can be represented by

$$\mathcal{A} = ([L_{\text{max}}] \times \mathcal{V}) \cup \{a_{\text{term}}\}$$

where L_{max} is the maximum length of the sequence that MDP considers, and $\mathcal{V}$ is a vocabulary set. Briefly, an action $(l, t) \in [L_{\text{max}}] \times \mathcal{V}$ substitutes l-th token in the target sequence to t.

B.2. Architectures

State encoders and action decoders. We adopt the policy model architectures described by Jain et al. (2023) for encoding states and decoding action logits in both appending and editing MDPs. The encoder architectures for both types of MDPs leverage transformer architectures to convert sequences into hidden features Devlin et al. (2019). In the appending MDP scenario, an MLP head predicts the action logit corresponding to the next token to be appended. For editing MDPs, an additional MLP head is employed to predict logits on positions, indicating the probability of substituting at that position. For a fair comparison, we maintain consistent configurations for state encoders and action decoders across all MDP-based subset selection methods (PC-RL, PC-MOGFN, Greedy + RL, and Ours).

Set encoders. Like preference conditioning methods, we incorporate a set encoder to extract features from given sets, employing a deep set architecture designed for point cloud classification tasks (Zaheer et al., 2017). This architecture includes 3 equivariant max-pooling layers with tanh activations and a 2-layer MLP head to derive hidden features from the sets. The parameter count in the set encoder depends on the dimension m of each point in the set and the hidden feature dimension N_{hid}, where m aligns with the number of objectives in our approach. Note that the set encoder's total parameter count, calculated as $2mN_{\text{hid}} + 6N_{\text{hid}}^2$, is substantially lower than that of transformer-based state encoders. For instance, a 3-layer transformer employed in bigrams tasks possesses more than $30N_{\text{hid}}^2$ parameters.

Incorporating MLM logits for decoding actions. Following Jain et al. (2023), we utilize appending MDP for single-round subset selection tasks with deterministic objective functions. For batch BO experiments in biological sequences, we utilize editing MDP architectures. Drawing inspiration from the LaMBO architecture (Stanton et al., 2022), we introduce a variant of our method that integrates an MLM model trained on previously evaluated data points during action decoding. Like LaMBO, which optimizes in MLM latent space using data-trained autoencoders, our approach benefits from initiating optimization with tokens likely found in the evaluated data. Note that, in our experimental setup, the MLM model is concurrently trained with the surrogate model. Additionally, for the pool of candidates to be edited, we employ hashing on the MLM predictions for each position, introducing a minimal additional computational cost.

B.3. Experimental Settings

Single-round subset selection with Bigrams tasks. For our main experiments in single-round subset selection, we employ the bigrams benchmark tasks as implemented by Stanton et al. (2022) and Jain et al. (2023). Each task is designed around target bigrams, with each corresponding to a specific objective. In this setup, synthetic bigram matching objectives serve as a deterministic surrogate model for tackling the active learning inner loop problem, with no prior data points evaluated and the Hypervolume indicator functioning as the batch acquisition function. Table 7 details the bigrams tasks.

We compare our method against PC-RLs and greedy-based approaches. Note that we train a single set-conditioned policy model to sample subsets with various cardinalities in this experiment. Our set-conditioned policy is trained with parameters $n_{\text{train}} = 64$, $N_t = 4$, over $N_u = 4000$ update steps, and a batch size of $N_e = 128$. At every 500 steps, we perform greedy sampling $\text{GS}(a, \pi_\theta^{\text{set}}, n, l = 128)$ for each cardinality constraint n to evaluate the Hypervolume indicator value. After the training ends, we report the best evaluated results for each n.

Given that $n_{\text{train}}/2$ samples are used for subset B conditioning at each step, our method uses $N_e + n_{\text{train}}/2 = 128 + 32 = 160$ samples per step. In contrast, other RL-based methods using $N_e = 128$ operate with 20% fewer samples. For a fair comparison, we provide other RL-based methods, including PC-RL (TS), PC-RL (WS), and Greedy + RL, with total 5000 update steps, 25% more steps than ours.

For the PC-RL baselines, preference vectors are sampled from a Dirichlet distribution with $\alpha = 1$. Like our method, we train a single PC-policy and utilize this policy to sample subsets in various cardinalities ns. To construct n-subset, we sample n preference vectors. For each vector, $l = 128$ candidates are sampled, conditioned on the preference vector. Then, the n-subset is formed by choosing the top candidate for each preference vector, adhering to protocols established in PC-based methods (Jain et al., 2023; Zhu et al., 2023). Similar to our approach, this sampling process is executed for each cardinality n at every 500 steps, continuing until the total number of update steps, $N_u = 5000$, is attained. For greedy approaches, optimization is carried out individually for each n, with a total budget of $B = N_u * N_e + (N_u/500) \times n \times l$ allocated for surrogate model queries across the process for a fair comparison. This allocation allows each iteration of the greedy method to use B/n budget for optimization. Greedy + RS selects the best sequence from B/n randomly sampled sequences at each iteration. Greedy + HC starts from a random sequence and iteratively moves to the optimal sequence within a 1-Hamming distance, restarting from another random point if necessary until the surrogate model budget of B/n is reached. Specifically, Greedy + RL utilizes $N_e = 128$ and sets N_u/n as update steps for each greedy loop. We determine the number of samples to deploy during each greedy loop based on B/n.

Table 7: Settings of bigrams tasks.

Task	Target Bigrams	Min. Len.	Max. Len.	Cardinalities
2 Bigrams	AV, VC	32	36	4, 16
3 Bigrams	AV, VC, CA	32	36	4, 16, 64
4 Bigrams	AV, VC, CA, AW	32	36	4, 16, 64, 256

Table 8: Hyperparameters for bigrams tasks and the DNA aptamer task

Hyperparameter	Values
η	1E-4, 1E-5, 1E-6
Random Action Prob.	0, 0.05

All RL-based methods employ a transformer encoder architecture with 3 layers, 8 heads, and a hidden dimension of 128. Also, we normalize returns as in Algorithm 5 for all RL-based methods. For RL-based methods (Ours, PC-RLs, Greedy + RL), we tune the hyperparameters among the combinations in Table 8. For each combination of hyperparameters, we run 10 trials and report the result from the best hyperparameters.

Single-round subset selection with DNA aptamers. We utilize three objectives, the number of hairpins, the number of pairs, and the energy value computed by the NUPACK library (Zadeh et al., 2011), adopting the implementation of Jain et al. (2023). In this setting, we use a larger transformer architecture with 4 layers, 16 heads, and a hidden dimension of 256. We set $N_u = 2000$ for our method and allocate 25% more update steps to other methods as in bigrams tasks. Other parameter settings are identical to the bigrams tasks.

Batch BO Experiments. For the batch BO, we consider three benchmarks from Stanton et al. (2022). Our primary benchmark is the RFP task, which optimizes the stability and solvent-accessible surface area (SASA) of RFPs. Additionally, we conduct experiments on two other benchmarks: 3 Bigrams (Table 7) and small molecules. The latter optimizes the logP and quantitative estimate of drug-likeness (QED) of SELFIES-encoded small molecules (Bickerton et al., 2012; Krenn et al., 2020).

Beyond addressing the issues identified in LaMBO's implementation (as detailed in Appendix D), we adopt similar experimental setups, with the exception of the number of samples generated at each inner loop. We train a set-conditioned policy with $n_{\text{train}} = n = 16$, $N_t = 1$, over $N_u = 256$ update steps and a batch size of $N_e = 128$. At every 64 step, we perform greedy sampling $\text{GS}(a, \pi_\theta^{\text{set}}, n = 16, l = 16)$, and propose the best sampled subset with the highest batch acquisition value for the proposal batch. To ensure a fair comparison, we increase the number of samples generated per step of each baseline method, from 32 to 2048 for MBGA, and from 16 to 256 for LaMBO. These modification leads to improvement in performance of baseline methods as illustrated in Figure 6. In addition, the modification results in the end-to-end process for the RFP task taking a similar scale of runtime between 2 to 3 days for all active learning based methods we consider in this scenario. Also, we utilize the same architecture and training algorithm for updating MTGP models for a fair compari-

Table 9: Ablation on training cardinality constraint. The mean and standard deviation values are calculated for 10 trials.

	Hypervolume Indicator (↑)								
	2 Bigrams		3 Bigrams			4 Bigrams			
Method	$n = 4$	$n = 16$	$n = 4$	$n = 16$	$n = 64$	$n = 4$	$n = 16$	$n = 64$	$n = 256$
Optimum	0.630		0.409			0.106			
Exact Greedy	0.568	0.630	0.350	0.408	0.409	0.055	0.078	0.097	0.106
Ours ($n_{\text{train}} = 128$)	**0.568** (0.000)	**0.630** (0.000)	**0.329** (0.005)	0.345 (0.003)	0.354 (0.006)	**0.055** (0.001)	0.076 (0.001)	0.090 (0.002)	**0.094** (0.002)
Ours ($n_{\text{train}} = 64$)	**0.568** (0.000)	**0.630** (0.000)	**0.329** (0.005)	**0.349** (0.007)	**0.359** (0.003)	**0.055** (0.000)	**0.077** (0.000)	**0.091** (0.002)	**0.094** (0.003)
Ours ($n_{\text{train}} = 32$)	**0.568** (0.000)	**0.630** (0.000)	0.328 (0.003)	0.345 (0.007)	**0.359** (0.005)	**0.055** (0.000)	0.076 (0.000)	0.086 (0.002)	0.089 (0.002)
Ours ($n_{\text{train}} = 16$)	0.564 (0.013)	0.622 (0.023)	0.326 (0.006)	0.352 (0.005)	0.355 (0.005)	**0.055** (0.000)	0.074 (0.001)	0.081 (0.002)	0.084 (0.001)
Ours ($n_{\text{train}} = 4$)	0.525 (0.000)	0.537 (0.000)	0.326 (0.007)	0.347 (0.007)	0.355 (0.005)	0.052 (0.001)	0.065 (0.002)	0.069 (0.003)	0.072 (0.003)

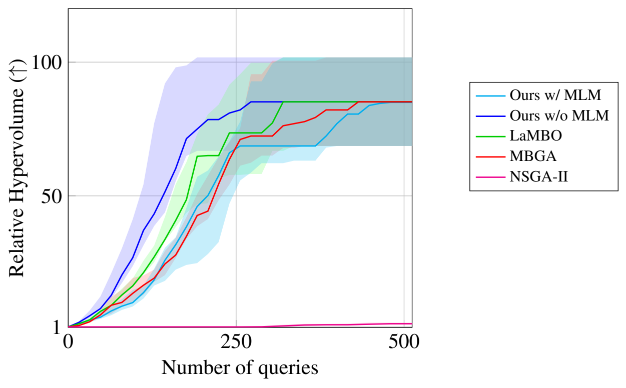


Figure 3: Multi-round active learning results on the 3 Bigrams task when using NEHVI as the batch acquisition function under a query limit of 512. Midpoint, lower, and upper boundaries show the 50th, 30th, and 70th percentiles, respectively, derived from 10 trials.

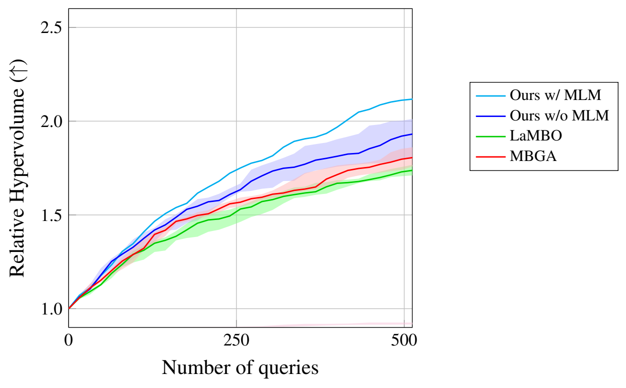


Figure 4: Multi-round active learning results on the RFP task when using UCBHVI as the batch acquisition function under a query limit of 512. Midpoint, lower, and upper boundaries show the 50th, 30th, and 70th percentiles, respectively, derived from 10 trials.

son. For set conditioning, we set the continuous features, $\text{feat}(\mathbf{x}) := \tilde{f}_{\text{UCB}}(\mathbf{x}; \beta = 0.1)$, for all experiments with statistical surrogate models. Also, we set the learning rate $\eta = 0.0001$ and set the random action probability to 0. Finally, we set the maximum edit budget of editing MDP to 1 as in LaMBO and MBGA.

C. Additional Experiments

C.1. Additional Results on Synthetic Tasks

Ablation on cardinality constraint. In our experiments, we differentiate training cardinality constraints from sampling constraints during our experiments on bigrams tasks. We adjust the training cardinality (n_{train}) and evaluate the effectiveness of models across varying set sizes through greedy sampling. To ensure comparable execution speed, N_t, the training step count, is set to $\max(1, n_{\text{train}}/16)$. Table 9 demonstrates that models trained with larger set sizes yield superior results across both smaller and larger cardinalities in bigrams tasks. As we introduced in Appendix B.3, 'Ours' with training cardinality constraint $n_{\text{train}} = 64$ corresponds to the 'Ours' in Table 1.

C.2. Additional Results on Batch BO Benchmarks

Additional multi-round batch BO results. Figure 3 illustrates the multi-round batch BO results on the 3 bigrams task with the MTGP surrogate model and NEHVI batch acquisition function. The results show that our method without MLM achieves higher relative Hypervolume indicator values faster than the baseline active learning results. However, for this synthetic task, MLM based strategy was not helpful for achieving the better performance as reported in Stanton et al. (2022). Next, we conduct multi-round batch BO with UCBHVI batch acquisition function on the RFP task. Figure 4 illustrates the performance of our method and the baseline methods equipped with UCBHVI. The results show that our methods achieve superior performance in this setting.

Additional subset selection results in the first round. To demonstrate the scalability and broad applicability of our method, we additionally evaluate the first-round subset selection performance, including runtime, across various batch acquisition functions. Table 10 presents the acquisition values and runtime obtained by optimizing NEHVI, UCBHVI,

Table 10: Subset selection results in the first round when optimizing various batch acquisition functions (NEHVI, UCBHVI, PES) on the RFP task. 'Ours-half' refers to our method with half the number of update steps. The mean and standard deviation values are calculated for 10 trials.

(a) NEHVI.

Method	NEHVI Value (↑)	Runtime (mins) (↓)
Ours	**0.779 (0.045)**	18.9
Ours-half	0.778 (0.033)	**9.5**
LaMBO	0.591 (0.033)	24.4
MBGA	0.654 (0.052)	14.0

(b) UCBHVI.

Method	UCBHVI Value (↑)	Runtime (mins) (↓)
Ours	**1.019 (0.032)**	4.1
Ours-half	1.005 (0.034)	**2.1**
LaMBO	0.776 (0.022)	16.7
MBGA	0.844 (0.054)	9.4

(c) PES.

Method	PES Value (↑)	Runtime (mins) (↓)
Ours	**1.233 (0.040)**	**12.8**
LaMBO w/ FD	0.070 (0.015)	18.0
MBGA	0.207 (0.107)	29.8

and PES in the RFP task. For PES, we use the implementation in the *BoTorch* framwork[1] and we modify LaMBO to use a gradient approximated by finite differences (FD) due to the non-differentiability of PES computation (Balandat et al., 2020). Our method ('Ours-half' for NEHVI and 'Ours' for UCBHVI and PES) achieved higher batch acquisition values in less runtime compared to baseline methods, demonstrating the effectiveness and scalability of our approach when optimizing various batch acquisition functions with statistical surrogate models.

C.3. Diversified subset selection results

Figure 5 illustrates the results of diversified subset selection for 2 bigrams tasks, comparing our method with PC-MOGFN. The results show that our method succeed to generate diverse candidates while keeping ability to generate near optimal solutions in the 2 bigrams task. Table 11 provides a summary of the diversified subset selection results in the first round on the RFP task in comparison with AL-MOGFN. The findings indicate that our method is capable of constructing subsets that are more diverse and have higher NEHVI values than those generated by the baseline method. However, unlike HVI-based batch acquisition functions, the features used for set conditioning in our method, $\text{feat}(\mathbf{x}) = \tilde{f}(\mathbf{x})$, might not offer enough information to steer the policy towards generating a variety of candidates.

[1]https://botorch.org/tutorials/information_theoretic_acquisition_functions.

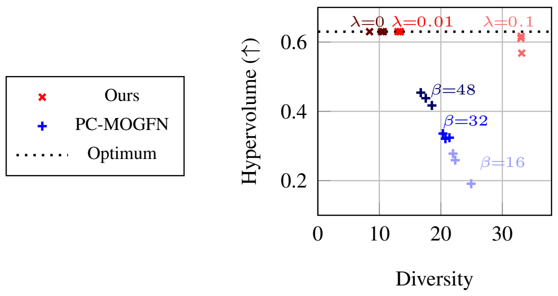


Figure 5: Diversified subset selection results on 2 bigrams task traversing tradeoff parameters. For each tradeoff parameter, β for PC-MOGFN, and λ for Ours, we plot 3 points for 3 different runs.

Table 11: Diversified subset selection results in the first round of the RFP task with NEHVI. The mean and standard deviation values are calculated for 10 trials.

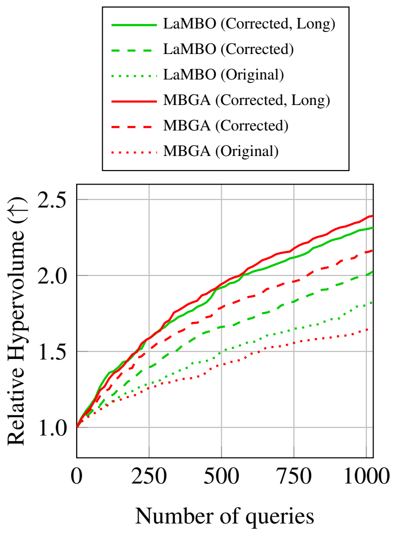
Method	NEHVI Value (↑)	Diversity (↑)
Ours w/o MLM ($\lambda = 0.0$)	0.779 (0.045)	78.352 (6.194)
Ours w/o MLM ($\lambda = 1.0$)	**0.731** (0.033)	**95.917** (0.35)
AL-MOGFN ($\beta = 16$)	0.608 (0.074)	93.253 (0.346)
AL-MOGFN ($\beta = 24$)	0.613 (0.061)	93.240 (0.200)

The development of techniques for extracting features that enhance diversity remains an area for future research in our study.

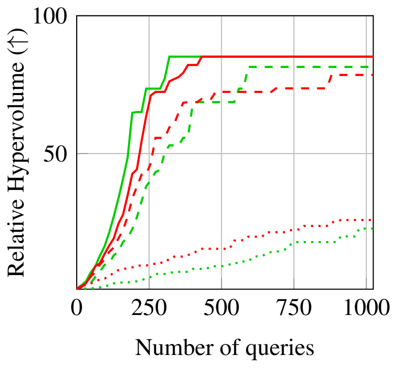
D. Addressing Previous Issues in LaMBO

The work by Stanton et al. (2022) has made significant contributions to establishing benchmarks for biological sequence design. Nonetheless, certain challenges were identified in the LaMBO implementation that impacted its performance. Firstly, the original implementation wrongly calculated the NEHVI batch acquisition values for batches containing more than one element. Secondly, an error in the mutation operation used in the GA methods was discovered, adversely affecting performance across several tasks.

In our study, we rectify these issues and conduct a performance comparison between the original version, our corrected version, and an enhanced version with a larger sample size, which we use as the baselines in our paper for a fair comparison. Figure 6 presents the benchmark results for the RFP task and 3 bigrams task. Notably, correcting these issues led to a substantial improvement in performance on the 3 bigrams task, achieving more than 3 times larger relative Hypervolume compared to the original implementation. Additionally, it was observed that increasing the number of samples during the inner loop contributed to improved performance for these tasks.



(a) RFP



(b) 3 Bigrams

Figure 6: Multi-round active learning results on the RFP task and 3 bigrams task, comparing performance before and after the implementation corrections. For clarity, only the median performance from 10 trials is depicted. The corrections resulted in significant performance enhancements in these tasks.