# BPO: Bayesian Preference Optimization for Multiobjective Discrete Optimization

Rahul Patel[1][0000−0001−6845−2941] and Elias B. Khalil[1][0000−0001−5844−9642]

University of Toronto, Toronto, ON Canada
rm.patel@mail.utoronto.ca, elias.khalil@utoronto.ca

**Abstract.** To approximate the Pareto frontier of a multiobjective discrete optimization problem, it is commonplace to solve a sequence of scalarized problems parameterized by *preference vectors* that regulate the tradeoff between different objectives. A central challenge lies in selecting preferences that lead to a diverse and high-quality approximation of the frontier within a limited computational budget. We propose *Bayesian Preference Optimization* (BPO), a structured framework that adaptively selects preferences to improve frontier quality. The method integrates augmented Tchebycheff scalarization (ATS) with a Bayesian optimization procedure that constructs a surrogate model in *preference space*, mapping preference vectors to objective outcomes. This formulation decouples the learning component from the dimensionality of the decision space, allowing the surrogate to scale with the number of objectives rather than the number of decision variables. Computational experiments on benchmark multiobjective knapsack and assignment problems show that BPO produces higher-quality Pareto frontier approximations than ATS with preferences sampled from a Dirichlet distribution, NSGA-II (a popular evolutionary algorithm), and KSA (an exact method) under comparable time budgets. The code is available at https://github.com/rahulptel/BPO.

**Keywords:** Multiobjective Optimization · Preference Optimization · Bayesian Optimization.

## 1 Introduction

Many real-world decision-making problems require optimizing multiple conflicting objectives simultaneously. A *multiobjective discrete optimization problem* (MODO) can be formulated as

$$\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) = \big(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})\big), \tag{1}$$

where $\mathbf{x}$ denotes a vector of decision variables belonging to a discrete feasible set $\mathcal{X} \subseteq \mathbb{Z}^n$, and $f : \mathcal{X} \to \mathbb{R}^m$ maps each assignment to an $m$-dimensional objective vector. The objectives $f_1, \ldots, f_m$ are often conflicting in that no single solution simultaneously optimizes all objectives. Such tradeoffs arise in numerous applications, including logistics (e.g., cost versus delivery time), finance (e.g., risk versus return), and energy systems (e.g., cost versus environmental impact).

Unlike single-objective optimization, which seeks a single global optimum, MODO aims to identify a set of nondominated points, also known as the Pareto frontier (see Section 2.1 for formal definitions). Exact solution methods [1,4] have been proposed to enumerate the full Pareto frontier. However, for most NP-hard discrete problems, these approaches become computationally intractable for realistic instances. Moreover, the number of Pareto solutions may be extremely large, rendering the full frontier difficult to store and interpret in practice.

Consequently, a significant body of work has focused on methods that aim to generate a high-quality approximation of the Pareto frontier within a fixed computational budget. Prominent examples include metaheuristics, particularly multiobjective evolutionary algorithms (EAs) such as NSGA-II [14] and MOEA/D [39]. While widely used, EAs often struggle to effectively handle the complex feasibility constraints inherent in discrete optimization problems. To address this limitation, a variety of mathematical programming-based heuristics (matheuristics) have been developed [30,31], and more recently, machine learning (ML)-based approaches have also been proposed [37].

At the core of many approximation techniques lie *scalarization-based* methods [28], which transform a multiobjective problem into a sequence of parameterized single-objective problems. These parameters are often represented by a *preference vector*, i.e., a nonnegative weight vector that encodes the relative emphasis assigned to the different objectives in the scalarization. Common scalarization schemes include the weighted-sum method [19,38], which is simple but cannot recover unsupported (non-convex) Pareto-optimal solutions, and the Tchebycheff method, which is capable of identifying both supported and unsupported solutions. By solving scalarized problems corresponding to different preference (or weight) vectors, one can construct an approximation of the Pareto frontier.

A critical challenge in scalarization-based approaches is the selection of preference vectors. A naive strategy, frequently adopted in practice, is to sample preference vectors uniformly at random. This approach can be inefficient, as it ignores information from previously solved scalarized problems and often produces redundant solutions or solutions concentrated in restricted regions of the frontier.

To overcome this limitation, we propose *Bayesian Preference Optimization* (BPO). As illustrated in Figure 1, BPO selects preference vectors that successively maximize an approximation of the hypervolume, yielding an anytime, high-quality approximation of the Pareto frontier. Despite the naming similarity, BPO is distinct from *preferential Bayesian optimization* [20], where the objective is latent and not directly observable, and feedback is instead obtained through pairwise comparisons. In contrast, BPO operates in a fully observable setting, where Bayesian optimization is used to adaptively select scalarization preference vectors, the corresponding scalarized problems are solved, and the resulting objective vectors are observed. Here, "preference" refers to the weight vector of the parameterized problems, rather than human preference judgments. The main contributions of this work are summarized as follows:

– **Novelty:** To the best of our knowledge, BPO is the first framework to employ Bayesian optimization for adaptive preference selection in MODO.

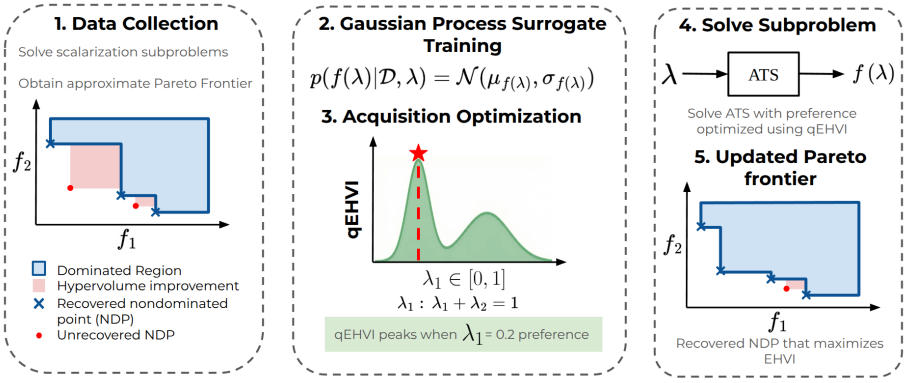



Fig. 1: Overview of the Bayesian Preference Optimization framework.

- **Generality:** The proposed framework is agnostic to the underlying problem structure. BPO can be applied to any MODO problem for which the augmented Tchebycheff scalarization can be solved, treating the underlying solver as a black box.
- **Adaptability:** By operating within an active learning paradigm, BPO generates preference suggestions that are instance-specific.
- **Performance:** Extensive computational experiments on benchmark problems demonstrate that BPO achieves superior hypervolume compared to baselines in both sequential and parallel settings. Notably, in the parallel setting, BPO exhibits performance closest to that of an empirical oracle constructed from the union of all collected solutions.

The remainder of the paper is organized as follows. Section 2 introduces the necessary preliminaries and notation. Related work is reviewed in Section 5. The proposed methodology is presented in Section 3. The computational setup and results are reported and discussed in Section 4. Finally, Section 6 concludes the paper and outlines directions for future research.

## 2 Preliminaries

### 2.1 Multiobjective Discrete Optimization

**Definition 1 (Pareto Dominance).** *Given two feasible solutions* $\mathbf{x}^1, \mathbf{x}^2 \in \mathcal{X}$, *we say that* $\mathbf{x}^1$ ***Pareto dominates*** $\mathbf{x}^2$ *or* $f(\mathbf{x}^1) \prec f(\mathbf{x}^2)$ *iff* $f_i(\mathbf{x}^1) \leq f_i(\mathbf{x}^2), \forall i \in \{1,\ldots,m\}$ *and* $f_j(\mathbf{x}^1) < f_j(\mathbf{x}^2)$ *for at least one* $j \in \{1,\ldots,m\}$.

**Definition 2 (Efficient Solution).** *A solution* $\mathbf{x}^\star \in \mathcal{X}$ *is called* ***efficient*** *(or* ***Pareto-optimal****) if there exists no other feasible solution* $\mathbf{x} \in \mathcal{X}$ *such that* $f(\mathbf{x}) \prec f(\mathbf{x}^\star)$. *The set of all efficient solutions* $\mathcal{X}_E = \{\mathbf{x} \in \mathcal{X} \mid \nexists\, \mathbf{x}' \in \mathcal{X}$ *such that* $f(\mathbf{x}') \prec f(\mathbf{x})\}$ *is called the* ***efficient set****.*

**Definition 3 (Weakly Efficient Solution).** *A solution* $\mathbf{x}^\star \in \mathcal{X}$ *is* ***weakly efficient*** *if there is no* $\mathbf{x} \in \mathcal{X}$ *such that* $f_i(\mathbf{x}) < f_i(\mathbf{x}^\star)$ *for all* $i \in \{1, \dots, m\}$.

**Definition 4 (Nondominated Point).** *The objective vector* $\mathbf{y}^\star = f(\mathbf{x}^\star)$ *corresponding to an efficient solution* $\mathbf{x}^\star$ *is referred to as a* ***nondominated point****. The set of all nondominated points* $\mathcal{Y}_N = \{f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}_E\}$ *is called the nondominated frontier or Pareto frontier.*

**Definition 5 (Supported and Unsupported Efficient Solutions).** *Let* $\mathcal{Y} = \{f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}$ *denote the set of attainable objective vectors. An efficient solution* $\mathbf{x}^\star \in \mathcal{X}_E$ *is called* ***supported*** *if its image* $f(\mathbf{x}^\star)$ *lies on the boundary of the convex hull of* $\mathcal{Y}$*, i.e., there exists a weight vector* $\boldsymbol{\lambda} \in \mathbb{R}^m_{>0}$ *such that* $\mathbf{x}^\star$ *minimizes* $\boldsymbol{\lambda}^\top f(\mathbf{x})$ *over* $\mathcal{X}$*. Otherwise,* $\mathbf{x}^\star$ *is said to be* ***unsupported****.*

*Tchebycheff Scalarization.* The popular Tchebycheff scalarization [34,15,33] can generate both supported and unsupported nondominated points. Let $\mathbf{z}^\star \in \mathbb{R}^m$ denote the (estimated) *ideal point* with components satisfying $z_i^\star \le f_i(\mathbf{x})$ for all feasible $\mathbf{x}$. For a preference vector (weight vector) $\boldsymbol{\lambda} \in \boldsymbol{\Lambda} = \{\boldsymbol{\lambda} \in \mathbb{R}^m_{>0} \mid \sum_i \lambda_i = 1\}$, the (weighted) Tchebycheff scalarization is defined as

$$g_{\mathrm{tch}}(\boldsymbol{\lambda}, \mathbf{z}^\star) \;=\; \min_{\mathbf{x} \in \mathcal{X}} \max_{i \in \{1,\dots,m\}} \; \lambda_i \big(f_i(\mathbf{x}) - z_i^\star\big). \tag{2}$$

Minimizers of (2) are weakly Pareto-optimal under mild regularity conditions. However, in discrete or nonconvex settings, this formulation can lead to degenerate ties or weakly efficient points. To promote strict Pareto efficiency, an augmented Tchebycheff scalarization (ATS) [34,15] adds a small tie-breaking penalty for a user-specified $\rho \in (0, 1)$:

$$g_{\mathrm{aug}}(\boldsymbol{\lambda}, \mathbf{z}^\star, \rho) \;=\; \min_{\mathbf{x} \in \mathcal{X}} \left( \max_{i \in \{1,\dots,m\}} \; \lambda_i \big(f_i(\mathbf{x}) - z_i^\star\big) \;+\; \rho \sum_{i=1}^{m} \lambda_i \big(f_i(\mathbf{x}) - z_i^\star\big) \right). \tag{3}$$

*Hypervolume.* In multiobjective settings, a standard quality indicator for a set of points in the objective space $\mathcal{Y}$ is the hypervolume [40], $\mathrm{HV}(\mathcal{Y}; \mathbf{r})$. It is the Lebesgue measure of the region dominated by $\mathcal{Y}$ and bounded by a reference point $\mathbf{r} \in \mathbb{R}^m$ and is defined as:

$$\mathrm{HV}(\mathcal{Y}; \mathbf{r}) = \mathcal{L}\left( \bigcup_{\mathbf{y} \in \mathcal{Y}} [\mathbf{y}, \mathbf{r}] \right), \tag{4}$$

where $\mathcal{L}$ denotes the Lebesgue measure in $\mathbb{R}^m$ and $[\mathbf{y}, \mathbf{r}] := \{\mathbf{z} \in \mathbb{R}^m \mid \mathbf{y} \le \mathbf{z} \le \mathbf{r}\}$. The hypervolume is popular because it provides a single scalar score that is Pareto-compliant (a nondominated point cannot decrease HV) and rewards both convergence (better objective values) and diversity (well-spread frontiers) [41].

### 2.2 Bayesian Optimization

Bayesian optimization (BO) is a sample-efficient framework for globally optimizing expensive, black-box functions. In the standard setting, we seek to maximize an unknown objective function $h : \mathcal{Z} \to \mathbb{R}$ over a domain $\mathcal{Z}$. The framework consists of two key components: a *surrogate model* that approximates $h$ based on observed data, and an *acquisition function* that directs the search for the optimum.

*Gaussian Processes.* The most common surrogate model in BO is the Gaussian Process (GP). A GP is a non-parametric distribution over functions, fully specified by a mean function $\mu(\mathbf{z})$ and a covariance kernel $k(\mathbf{z}, \mathbf{z}')$. Given a dataset of observed pairs $\mathcal{D}_n = \{(\mathbf{z}_i, h(\mathbf{z}_i))\}_{i=1}^n$, the GP provides a predictive posterior distribution for a new candidate point $\mathbf{z}$. The posterior mean $\mu_n(\mathbf{z})$ predicts the function value, while the posterior variance $\sigma_n^2(\mathbf{z})$ quantifies the uncertainty.

*Acquisition Functions.* An acquisition function $\alpha : \mathcal{Z} \to \mathbb{R}$ assigns a utility value to candidate points in the domain based on the posterior distribution of the surrogate model. It balances *exploitation* (sampling where the model predicts high objective values) and *exploration* (sampling where the model uncertainty is high). At each iteration $t$, the next sampling point is selected by maximizing this function:

$$\mathbf{z}_{t+1} \in \arg\max_{\mathbf{z} \in \mathcal{Z}} \alpha(\mathbf{z} \mid \mathcal{D}_t). \tag{5}$$

The *Expected Hypervolume Improvement* (EHVI) is an acquisition function designed specifically for multiobjective BO. Given the current approximate nondominated set $\hat{\mathcal{Y}}_N$ and a candidate input $\mathbf{z} \in \mathcal{Z}$, EHVI quantifies the expected increase in hypervolume if we evaluate $\mathbf{z}$ and obtain a new objective vector $\mathbf{y}$. Let $I(\mathbf{y}, \hat{\mathcal{Y}}_N) = \mathrm{HV}(\hat{\mathcal{Y}}_N \cup \{\mathbf{y}\}) - \mathrm{HV}(\hat{\mathcal{Y}}_N)$ be the improvement yielded by a new point $\mathbf{y}$. Since $\mathbf{y}$ is unknown before evaluation, EHVI computes the expectation over the predictive posterior distribution of the surrogate model $p(\mathbf{y} \mid \mathbf{z}, \mathcal{D})$:

$$\alpha_{\mathrm{EHVI}}(\mathbf{z}) = \mathbb{E}_{\mathbf{y} \sim p(\cdot \mid \mathbf{z})} \left[ I(\mathbf{y}, \hat{\mathcal{Y}}_N) \right]. \tag{6}$$

By maximizing $\alpha_{\mathrm{EHVI}}$, the optimization algorithm selects the candidate $\mathbf{z}$ that is statistically most likely to expand the dominated region of the current Pareto frontier approximation. However, evaluating it exactly is computationally intractable [8,11] as the number of objectives $m$ increases. Additionally, it is limited to sequential selection of $\mathbf{z}$. To address these challenges, $q$-EHVI was proposed as a differentiable and scalable proxy for approximating EHVI [12].

## 3 Bayesian Preference Optimization

We propose a novel heuristic for generating high-quality approximations of the Pareto frontier in MODO problems by combining ATS and BO. As discussed earlier, the choice of preferences used in solving the ATS has a significant impact on the resulting frontier, and intelligently selecting these preferences can substantially

**Algorithm 1:** The BPO Framework

**Data:** Number of initial samples $N_{\text{init}}$, number of main iterations $N_{\text{iter}}$, surrogate model $\mathcal{M}$, acquisition function $\alpha$
**Result:** Approximate Pareto frontier $\mathcal{Y}$

$\mathcal{Y} \leftarrow \emptyset, \mathcal{D} \leftarrow \emptyset$
Sample $\{\boldsymbol{\lambda}^i\}_{i=1}^{N_{\text{init}}} \sim \text{Dirichlet}(\mathbf{1})$
**for** $k = 1, \ldots, N_{\text{init}}$ **do**
  Solve scalarized problem with parameter $\boldsymbol{\lambda}^k$ and obtain $\mathbf{y}^k$
  $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{\mathbf{y}^k\}, \mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{\lambda}^k, \mathbf{y}^k)\}$
Train surrogate model $\mathcal{M}$ using dataset $\mathcal{D}$
**for** $k = 1, \ldots, N_{\text{iter}}$ **do**
  $\boldsymbol{\lambda}^\star \leftarrow \arg\max_{\boldsymbol{\lambda} \in \boldsymbol{\Lambda}} \alpha(\boldsymbol{\lambda} \mid \mathcal{M})$
  Solve scalarized problem with parameter $\boldsymbol{\lambda}^\star$ and obtain $\mathbf{y}^\star$
  $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{\mathbf{y}^\star\}, \mathcal{D} \leftarrow \mathcal{D} \cup \{(\boldsymbol{\lambda}^\star, \mathbf{y}^\star)\}$
  Retrain surrogate model $\mathcal{M}$ using dataset $\mathcal{D}$
**Return** $\mathcal{Y}$

improve approximation quality. We formulate preference selection as a bilevel optimization problem. Let $\Phi(\cdot)$ denote a real-valued set function that maps a set of objective vectors to a score, where a higher score indicates a better approximation of the Pareto frontier. Given a current set $\mathcal{Y}_k$ of $k$ nondominated solutions, the objective is to identify a preference vector $\boldsymbol{\lambda}^\star$ on the $m$-dimensional simplex $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda} \in \mathbb{R}^m : \boldsymbol{\lambda}^T \mathbf{1} = 1, \boldsymbol{\lambda} \geq \mathbf{0}\}$ that induces a new objective vector $\mathbf{y}^\star$ yielding the largest improvement in frontier quality. This problem can be cast as:

$$\max_{\mathbf{y}, \mathbf{x}, \boldsymbol{\lambda}} \quad \Phi\big(\mathcal{Y}_k \cup \{\mathbf{y}\}\big) \tag{7}$$

$$\text{s.t.} \quad \mathbf{y} = f(\mathbf{x}), \tag{8}$$

$$\mathbf{x} = \arg\min_{\mathbf{x}' \in \mathcal{X}} \left( \max_{i \in \{1,\ldots,m\}} \lambda_i \big(f_i(\mathbf{x}') - z_i^\star\big) \; + \; \rho \sum_{i=1}^{m} \lambda_i \big(f_i(\mathbf{x}') - z_i^\star\big) \right), \tag{9}$$

$$\boldsymbol{\lambda} \in \boldsymbol{\Lambda}. \tag{10}$$

Directly solving the problem (7)–(10) is computationally intractable. First, the lower-level problem (9) is an NP-hard discrete optimization problem. Second, using the hypervolume as the objective function $\Phi(\cdot)$ is problematic as computing it exactly is #P-hard [8].

To address these challenges, we adopt a BO approach and construct a surrogate model $\mathcal{M}$ that learns the mapping from preferences to objective vectors, $\mathcal{M} : \boldsymbol{\lambda} \mapsto \mathbf{y}$. This surrogate enables us to predict the outcome of an expensive scalarized problem without explicitly solving it. In contrast, conventional multiobjective BO methods typically model the mapping from the decision space to the objective space, $\mathbf{x} \mapsto \mathbf{y}$. A key limitation of this approach is that the complexity of the surrogate model scales with the dimensionality of the decision space. Our approach avoids this issue by learning directly in preference space which is typically much

smaller (equal to the number of objectives) and grows more slowly than the number of variables in applications, resulting in a compact surrogate model.

We define an acquisition function $\alpha(\boldsymbol{\lambda} \mid \mathcal{Y}_k)$ over the surrogate model, which quantifies the utility of evaluating a candidate preference given the current approximation of the Pareto frontier. In particular, we employ the $q$EHVI acquisition function, evaluated using the surrogate model's predictive distribution, to identify a preference that approximately maximizes the expected improvement in frontier quality. The selected preference is then evaluated by solving the corresponding scalarized problem (e.g., using an integer programming solver), and the resulting objective vector is added to $\mathcal{Y}_k$. The overall procedure is summarized in Algorithm 1.

*Parallelism* The ATS admits an embarrassingly parallel implementation, as scalarized problems corresponding to different preference vectors can be solved independently. This property naturally extends to the proposed framework. In particular, beyond the single-run variant, we consider a multi-start variant in which multiple independent instances of the BPO framework are executed in parallel, each with a different random seed. The final approximate Pareto frontier is obtained by taking the union of the nondominated solutions from all runs.

The inherent randomness in the ATS method, which is known to be beneficial in practice, also plays a constructive role in our approach. Different random initializations lead to distinct initial training datasets for the surrogate model, which in turn alter the subsequent BO trajectories. As a result, independent runs tend to explore complementary regions of the preference space, yielding a more diverse and higher-quality approximation of the Pareto frontier when aggregated.

## 4 Experimental Results

*Computational setup* We evaluate the proposed framework on two canonical multiobjective combinatorial optimization benchmarks: the multiobjective 0–1 knapsack problem (MOKP) and the multiobjective assignment problem (MOAP). The detailed instance generation procedure is provided in Appendix A. For each problem class, we generate instances with 7 distinct size configurations. For MOKP, the number of items varies from 500 to 1250, with the number of objectives ranging from 3 to 6. For MOAP, the number of agents varies from 50 to 200, while the number of objectives ranges from 3 to 7. We generate 25 random instances for each configuration. These specific sizes were selected to ensure the problems are computationally non-trivial yet tractable for extensive experimentation. In particular, we target instance sizes where solving a single ATS problem requires, on average, between 1 and 5 seconds. Based on the computational cost of solving a scalarized problem, we impose a total time limit of either 240 or 480 seconds per instance, as detailed in Appendix B.

We compare BPO against ATS with preferences sampled uniformly from the unit simplex, the popular evolutionary algorithm NSGA-II [14], and the exact KSA method [24]. All experiments were run on the Nibi cluster using a

single CPU (Intel Xeon 6972P @ 2.4 GHz) with 16 GB RAM and Python 3.10. We used Gurobi 11.0.3 [21] to solve the MIP models arising from the ATS and other baselines. We implemented the NSGA-II baseline in `pymoo` (v0.6.0.1) [7] and used `pygmo` (v2.19.5) to calculate the hypervolume [6]. The BO loop was implemented using `BoTorch` (v0.11.1) [3]. The hyperparameter settings used for different algorithms are specified in Appendix B. To determine the data requirements for effective surrogate modeling within the BPO framework, we compare two versions of the algorithm—BPO-100 and BPO-150—parameterized by an initial sample size ($N_{\text{init}}$) of 100 and 150, respectively. To ensure robust and fair comparisons, all stochastic methods (BPO, ATS, and NSGA-II) are executed using the same set of five random seeds for each problem instance. Since KSA is deterministic, we run it once per instance.

*Results* We evaluate the performance of BPO against the baselines around six research questions addressing (i) solution quality relative to baselines, (ii) scalability with problem size, (iii) sensitivity to the size of the initial design, (iv) sample efficiency, (v) performance under parallel execution, and (vi) the gap to an empirical *Oracle*. We present detailed results in the main text for two representative size configurations per problem class. Specifically, we select sizes that have either the largest number of variables or objectives. Complete results are provided in Appendix C. We note that the key takeaways discussed below remain consistent across the complete dataset.

**Q1. How does BPO perform in comparison to baseline methods?** Table 1 reports summary statistics of the HV attained by each method in a fixed time budget. Overall, BPO substantially outperforms both KSA and NSGA-II across the tested size configurations. KSA exhibits limited scalability as the number of objectives increases and returns fewer nondominated solutions than the other methods. This behavior is consistent with its comparatively high bookkeeping overhead (e.g., maintaining and pruning partitions of the objective space), which reduces the effective effort devoted to improving solution quality. NSGA-II yields stronger approximations than KSA, but remains consistently inferior to scalarization-based approaches; this gap is particularly pronounced on MOAP. Relative to ATS with random preference sampling, BPO achieves a higher mean HV, except for MOAP with 7 objectives.

To further corroborate the performance of BPO, we compute its win-rate relative to the baseline ATS. Specifically, for each problem configuration, we compare the final hypervolume obtained by both methods on every unique instance and random seed. The win-rate is calculated as the proportion of experiments (out of 25 instances $\times$ 5 seeds $=$ 125 runs) in which a method achieves a strictly higher hypervolume than its competitor. These results are presented in Table 2. BPO-100 and BPO-150 collectively exceed a $\approx 98\%$ win rate in two configurations and over 80% in one of them, out of the four cases, over ATS. These results indicate that adaptively selecting preferences via BPO produces higher-quality Pareto front approximations than uniform preference sampling within the same computational budget.

Table 1: Final HV results on MOKP and MOAP problem sizes. The column "Iter." reports the average number of iterations a method was able to run. The average number of nondominated solutions found is provided in column $|\hat{\mathcal{Y}}_N|$.

| MOKP | | | | | | | | | MOAP | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Hypervolume % | | | | | | | | | Hypervolume % | | | | | |
| $m$ | $n$ | Method | Mean. | Std. | Min. | Max. | Iter. | $\lvert\hat{\mathcal{Y}}_N\rvert$ | $m$ | $n$ | Method | Mean. | Std. | Min. | Max. | Iter. | $\lvert\hat{\mathcal{Y}}_N\rvert$ |
| 3 | 1250 | KSA | 64.75 | 1.65 | 62.45 | 68.24 | 708 | 706 | 3 | 200 | KSA | 74.11 | 1.19 | 70.93 | 76.64 | 626 | 625 |
| | | NSGA-II | 85.57 | 0.58 | 84.15 | 86.80 | – | 200 | | | NSGA-II | 47.68 | 1.25 | 45.00 | 51.86 | – | 200 |
| | | ATS | 95.49 | 0.42 | 94.18 | 96.24 | 217 | 215 | | | ATS | 96.15 | 1.99 | 74.78 | 97.14 | 185 | 185 |
| | | **BPO-100** | **96.80** | 0.20 | 96.29 | 97.23 | 362 | 159 | | | **BPO-100** | **97.36** | 0.37 | 95.93 | 97.80 | 193 | 163 |
| | | BPO-150 | 96.73 | 0.57 | 94.18 | 97.28 | 296 | 183 | | | BPO-150 | 97.01 | 0.71 | 95.05 | 97.88 | 184 | 173 |
| 6 | 500 | KSA | 29.49 | 2.54 | 22.89 | 35.20 | 26 | 24 | 7 | 50 | KSA | 2.88 | 0.00 | 2.88 | 2.88 | 26 | 18 |
| | | NSGA-II | 65.10 | 1.85 | 58.91 | 69.45 | – | 200 | | | NSGA-II | 8.24 | 0.49 | 7.07 | 9.65 | – | 200 |
| | | ATS | 77.75 | 1.44 | 73.15 | 80.60 | 118 | 118 | | | **ATS** | **39.89** | 2.26 | 34.94 | 46.77 | 233 | 233 |
| | | **BPO-100** | **79.80** | 2.83 | 72.26 | 84.90 | 113 | 107 | | | BPO-100 | 37.13 | 1.51 | 34.44 | 42.21 | 170 | 159 |
| | | BPO-150 | 78.26 | 1.82 | 74.22 | 84.80 | 116 | 115 | | | BPO-150 | 38.73 | 1.70 | 34.85 | 43.95 | 187 | 182 |

Table 2: Win-rate comparison: BPO vs. ATS.

| MOKP | | | | | MOAP | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $m$ | $n$ | BPO-100 | BPO-150 | ATS | $m$ | $n$ | BPO-100 | BPO-150 | ATS |
| 3 | 1250 | 25.60 | **74.40** | 0.00 | 3 | 200 | **62.00** | 36.40 | 1.60 |
| 6 | 500 | **64.80** | 16.40 | 18.80 | 7 | 50 | 0.00 | 8.00 | **92.00** |

**Q2. How does BPO scale with problem size?** The performance of BPO is largely robust to increases in the decision-space dimension $n$, but is more sensitive to increases in the objective-space dimension $m$. As $n$ grows, BPO maintains a consistent advantage over ATS, which aligns with the fact that the surrogate model and acquisition function are defined over the preference space whose dimension equals $m$, rather than $n$. In lower objective dimensions, BPO performs particularly well. However, in higher-dimensional objective settings (e.g., MOAP with $m = 7$), BPO completes fewer BO iterations than ATS (see Table 1), and consequently may underperform in mean HV. The dominant factor is the increased computational cost of optimizing the acquisition function as $m$ grows, which reduces the number of effective iterations and typically results in fewer nondominated points.

**Q3. How does the size of the initial training set affect the performance of BPO?** A comparison between BPO-100 and BPO-150 provides insight into the method's data requirements. In lower-dimensional objective setting ($m = 3$), BPO-100 generally matches or outperforms BPO-150 (Table 2). With fewer initial samples, a larger fraction of the computational budget is allocated to the BO loop, enabling the acquisition-driven search to more rapidly concentrate on high-utility regions of the frontier. In contrast, in higher-dimensional configurations (notably MOAP with $m = 6$), BPO-150 more frequently attains superior HV, suggesting

Table 3: Comparison of parallel (k=5) HV results with Oracle.

| MOKP | | | | | | | | MOAP | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Hypervolume % | | | | | | | | Hypervolume % | | | | |
| $m$ | $n$ | Method | Mean. | Std. | Min. | Max. | $\lvert\hat{\mathcal{Y}}_N\rvert$ | $m$ | $n$ | Method | Mean. | Std. | Min. | Max. | $\lvert\hat{\mathcal{Y}}_N\rvert$ |
| 3 | 1250 | ATS | 96.36 | 0.20 | 95.94 | 96.74 | 1067 | 3 | 200 | ATS | 97.49 | 0.11 | 97.27 | 97.68 | 916 |
| | | BPO-100 | 97.09 | 0.18 | 96.75 | 97.46 | 771 | | | BPO-100 | 98.18 | 0.08 | 98.01 | 98.30 | 740 |
| | | **BPO-150** | **97.10** | 0.18 | 96.77 | 97.47 | 895 | | | **BPO-150** | **98.20** | 0.09 | 97.98 | 98.35 | 831 |
| | | Oracle | 97.15 | 0.18 | 96.81 | 97.51 | 2180 | | | Oracle | 98.31 | 0.08 | 98.14 | 98.43 | 1848 |
| 6 | 500 | ATS | 82.02 | 0.96 | 79.86 | 83.81 | 591 | 7 | 50 | **ATS** | **47.46** | 1.82 | 44.37 | 52.95 | 1164 |
| | | **BPO-100** | **84.75** | 1.72 | 80.73 | 87.53 | 529 | | | BPO-100 | 45.86 | 1.52 | 43.59 | 50.40 | 773 |
| | | BPO-150 | 83.11 | 1.31 | 81.45 | 87.38 | 577 | | | BPO-150 | 46.91 | 1.59 | 44.13 | 51.66 | 903 |
| | | Oracle | 85.26 | 1.51 | 81.66 | 87.96 | 761 | | | Oracle | 49.06 | 1.92 | 45.99 | 54.80 | 1564 |

that a larger initial design can be necessary to obtain a sufficiently accurate surrogate prior to active preference selection when the input dimension is higher.

**Q4. Is BPO more sample-efficient than baselines?** While BPO incurs additional overhead from surrogate training and acquisition optimization, it typically uses the query budget more effectively. As shown in Table 1, the number of nondominated solutions recovered by BPO (i.e., $|\hat{\mathcal{Y}}_N|$) is often comparable to, or slightly smaller than, that of ATS, yet it achieves higher HV. This pattern indicates improved sample efficiency: BPO preferentially allocates evaluations to regions that yield larger marginal HV gains. By selecting preferences to maximize EHVI, BPO reduces redundant sampling in regions that are already well represented, which is a common failure mode of uniform preference sampling.

**Q5. How does BPO perform in a parallel setting compared to ATS?** To emulate a parallel execution regime, we aggregate results from $k$ independent runs (seeds). For each method and instance, we compute the union of nondominated solutions obtained across the $k$ runs and evaluate the resulting HV. We then average this HV across instances and plot it as a function of $k$ in Figure 2, with

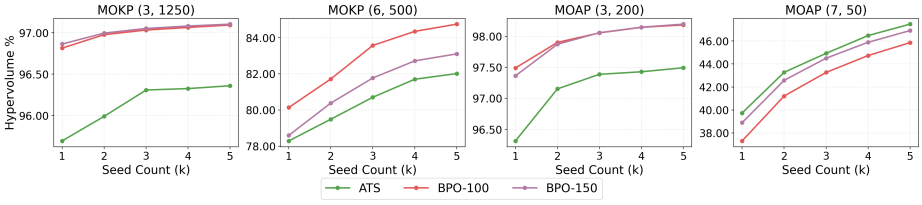



Fig. 2: HV progression over number of seeds, for MOKP and MOAP, averaged across 25 instances.

$k$ varying from 1 to 5. For MOKP, BPO (red and purple curves) consistently dominates ATS (green curve) for all seed counts. The improvement remains substantial even at $k = 5$, indicating that independent instances of BPO explore complementary parts of the frontier and that the resulting pooled set of solutions yields strong gains in HV. Analogous trends can be observed for MOAP, with the exception occurring at $m = 7$.

**Q6. How does BPO perform against an Oracle?** To contextualize the remaining performance gap, we compare against an empirical Oracle constructed by taking, for each instance, the union of all nondominated solutions produced across all methods and seeds, and then computing the HV of this combined set. We emphasize that even the Oracle HV need not equal 100%, since HV is normalized with respect to the ideal point, which is typically unattainable.

Table 3 reports Oracle HV statistics and compares them to BPO and ATS under the pooled $k = 5$ setting. Across most configurations, BPO approaches the Oracle HV while using fewer nondominated solutions, indicating that its preference selection mechanism concentrates evaluations on regions that contribute disproportionately to HV. These results suggest that BPO can recover near-Oracle HV with a comparatively smaller approximate Pareto frontier.

## 5 Related Work

Several exact and heuristic approaches have been proposed in the literature to solve MODO problems [16,17]. The exact methods [24,35,18,1,4,23] aim to enumerate the Pareto frontier and are mainly geared towards completing the search in as little time as possible. However, for large MODO problems, the search is seldom completed within a realistic computational budget. Consequently, the set of nondominated solutions available upon early termination can be of poor quality, i.e., lacking diversity or missing large regions of the frontier. This has led to the development of heuristic methods based on evolutionary algorithms, matheuristics, and machine learning that seek high-quality frontiers under a fixed budget.

Multiobjective evolutionary algorithms (MOEAs) maintain a population of candidate solutions and iteratively apply variation operators (e.g., crossover and mutation) and selection mechanisms designed to promote both (i) convergence towards the nondominated frontier and (ii) diversity along it. A prominent representative is NSGA-II [14], which ranks solutions using fast nondominated sorting and preserves diversity via a crowding-distance estimator. Decomposition-based methods, most notably MOEA/D [39], decompose the problem into a set of scalarized problems and solve them simultaneously using neighborhood information, exploiting locality in the weight space. Indicator-based methods optimize a chosen quality indicator directly; for instance, IBEA [42] defines selection pressure via pairwise indicator values, while SMS-EMOA [5] steers the population by repeatedly removing the solution with the smallest hypervolume contribution. Despite their success, handling *hard constraints* remains a key

challenge for MOEAs. Feasibility is often promoted via penalty functions, repair operators, or constraint-domination rules rather than being guaranteed by the evolutionary search itself [10,13].

Matheuristics hybridize metaheuristic search strategies with exact mathematical programming techniques, such as linear programming relaxations and mixed-integer programming (MIP), to tackle hard combinatorial problems. In the multiobjective context, several approaches have adapted single-objective mechanisms like local branching and the feasibility pump. For instance, variable neighbourhood search has been integrated with local branching to approximate the Pareto front of bi-objective mixed binary integer programs [32]. Multi-Directional Local Search [36] offers another perspective by explicitly guiding local search along multiple directions in the objective space. A significant line of work leverages the feasibility pump heuristic, combining it with local search to generate feasible solutions for bi-objective integer linear programs [30]. This approach was subsequently generalized to generic multi-objective MIPs [31]. More recently, matheuristics specifically for tri-objective binary integer linear programs have been introduced, incorporating path relinking to explore trajectories between elite solutions in the decision space [2].

Recent advances in neural combinatorial optimization have spurred interest in learning-based approaches for MODO. Decomposition-based strategies often adapt the MOEA/D framework, decomposing the MODO problem into multiple scalarized problems solved by deep reinforcement learning [25]. While effective, these methods often require maintaining multiple models to cover the efficient set. To address this, Pareto set learning (PSL) [29] aims to learn a single, unified model conditioned on a preference vector [26]. This paradigm has been further enhanced by explicitly incorporating diversity objectives into the training process to prevent mode collapse [9]. Similarly, for expensive black-box functions, PSL has been utilized to approximate the Pareto set with limited evaluations [27]. Distinct from these end-to-end heuristics, [37] used a graph neural network to improve the search efficiency of an exact solver. However, unlike BPO, they rely on offline training and do not adapt to the Pareto frontier during search.

The most relevant work to our approach is [22], where a preference optimization method for Pareto Set Learning in black-box settings has been proposed. They introduced a bilevel optimization framework to actively optimize the distribution of preference points. Their approach utilizes a differentiable cross-entropy method to ensure that the generated solutions are evenly distributed along the Pareto front, thereby moving beyond static scalarization strategies. However, their framework is tailored to continuous decision spaces and does not apply to discrete optimization problems. It also requires a fixed set of reference directions to be specified a priori.

## 6 Conclusion

This paper introduced BPO, a general framework for approximating Pareto frontiers in multiobjective discrete optimization problems under a limited com-

putational budget. The central idea is to treat the selection of scalarization preferences as a sequential decision problem and to adaptively choose preference vectors using BO. By building a surrogate model in preference space, rather than in the typically high-dimensional decision space, BPO decouples the learning component from the number of decision variables and instead scales primarily with the number of objectives. This design enables principled preference selection while maintaining compatibility with off-the-shelf MIP solvers through ATS.

Across benchmark MOKP and MOAP instances, our method consistently improved hypervolume relative to widely used baselines, including ATS with preferences sampled uniformly from a unit simplex, NSGA-II, and KSA under comparable time limits. In particular, BPO attained higher mean hypervolume in the majority of tested configurations and achieved strong win rates over ATS, while often producing comparable or smaller Pareto frontiers, indicating improved sample efficiency. We also demonstrated that aggregating multiple independent runs yields substantial gains in a parallel setting, suggesting that BPO benefits from multi-start diversity and can effectively exploit additional computational resources. Finally, comparisons against an empirical Oracle showed that the resulting approximations frequently approach the best hypervolume achievable by pooling solutions across methods and seeds, despite using fewer nondominated points.

A key limitation of the current implementation is scalability in the number of objectives. As $m$ increases, optimizing the acquisition function becomes substantially more expensive, which reduces the number of effective BO iterations within a fixed time budget and can diminish hypervolume gains in higher-dimensional objective spaces. Therefore, developing more efficient strategies for acquisition optimization in many-objective settings is an important direction for future work.

## References

1. Adelgren, N., Gupte, A.: Branch-and-bound for biobjective mixed-integer linear programming. INFORMS Journal on Computing **34**(2), 909–933 (2022)
2. An, D., Parragh, S.N., Sinnl, M., Tricoire, F.: A matheuristic for tri-objective binary integer linear programming. Computers & Operations Research **161**, 106397 (2024)
3. Balandat, M., Karrer, B., Jiang, D.R., Daulton, S., Letham, B., Wilson, A.G., Bakshy, E.: BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In: Advances in Neural Information Processing Systems 33 (2020), http://arxiv.org/abs/1910.06403
4. Bergman, D., Bodur, M., Cardonha, C., Cire, A.A.: Network models for multiobjective discrete optimization. INFORMS Journal on Computing **34**(2), 990–1005 (2022)
5. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research **181**(3), 1653–1669 (2007). https://doi.org/10.1016/j.ejor.2006.08.008
6. Biscani, F., Izzo, D.: A parallel global multiobjective framework for optimization: pagmo. Journal of Open Source Software **5**(53), 2338 (2020). https://doi.org/10.21105/joss.02338, https://doi.org/10.21105/joss.02338

7. Blank, J., Deb, K.: pymoo: Multi-objective optimization in python. IEEE Access **8**, 89497–89509 (2020)
8. Bringmann, K., Friedrich, T.: Approximating the volume of unions and intersections of high-dimensional geometric objects. Computational Geometry **43**(6-7), 601–610 (2010)
9. Chen, J., Zhang, Z., Cao, Z., Wu, Y., Ma, Y., Ye, T., Wang, J.: Neural multi-objective combinatorial optimization with diversity enhancement. Advances in Neural Information Processing Systems **36**, 39176–39188 (2023)
10. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Computer Methods in Applied Mechanics and Engineering **191**(11–12), 1245–1287 (2002). https://doi.org/10.1016/S0045-7825(01)00323-1
11. Couckuyt, I., Deschrijver, D., Dhaene, T.: Fast calculation of multiobjective probability of improvement and expected improvement criteria for pareto optimization. Journal of Global Optimization **60**(3), 575–594 (2014)
12. Daulton, S., Balandat, M., Bakshy, E.: Differentiable expected hypervolume improvement for parallel multi-objective bayesian optimization. Advances in neural information processing systems **33**, 9851–9864 (2020)
13. Deb, K.: An efficient constraint handling method for genetic algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO). pp. 844–851 (2000)
14. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6**(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
15. Despontin, M.: Multiple criteria optimization: Theory, computation, and application, ralph e. steuer (ed.). wiley, palo alto, ca (1986). European Journal of Operational Research **32**, 3–7 (1986), https://api.semanticscholar.org/CorpusID:63093894
16. Ehrgott, M.: Multicriteria Optimization. Springer, 2 edn. (2005). https://doi.org/10.1007/3-540-27659-9
17. Ehrgott, M., Gandibleux, X.: A survey and annotated bibliography of multiobjective combinatorial optimization. OR-spektrum **22**(4), 425–460 (2000)
18. Forget, N., Gadegaard, S.L., Klamroth, K., Nielsen, L.R., Przybylski, A.: Branch-and-bound and objective branching with three or more objectives. Computers & Operations Research **148**, 106012 (2022)
19. Gass, S., Saaty, T.: The computational algorithm for the parametric objective function. Naval research logistics quarterly **2**(1-2), 39–45 (1955)
20. González, J., Dai, Z., Damianou, A., Lawrence, N.D.: Preferential bayesian optimization. In: International Conference on Machine Learning. pp. 1282–1291. PMLR (2017)
21. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2026), https://www.gurobi.com
22. Haishan, Z., Das, D., Tsuda, K.: Preference-optimized pareto set learning for blackbox optimization. arXiv preprint arXiv:2408.09976 (2024)
23. Halffmann, P., Schäfer, L.E., Dächert, K., Klamroth, K., Ruzika, S.: Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey. Journal of Multi-Criteria Decision Analysis **29**(5-6), 341–363 (2022)
24. Kirlik, G., Sayın, S.: A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. European Journal of Operational Research **232**(3), 479–488 (2014)

25. Li, K., Zhang, T., Wang, R.: Deep reinforcement learning for multiobjective optimization. IEEE transactions on cybernetics **51**(6), 3103–3114 (2020)
26. Lin, X., Yang, Z., Zhang, Q.: Pareto set learning for neural multi-objective combinatorial optimization. In: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net (2022), https://openreview.net/forum?id=QuObT9BTWo
27. Lin, X., Yang, Z., Zhang, X., Zhang, Q.: Pareto set learning for expensive multiobjective optimization. Advances in neural information processing systems **35**, 19231–19247 (2022)
28. Miettinen, K.: Nonlinear multiobjective optimization, vol. 12. Springer Science & Business Media (1999)
29. Navon, A., Shamsian, A., Fetaya, E., Chechik, G.: Learning the pareto front with hypernetworks. In: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net (2021), https://openreview.net/forum?id=NjF772F4ZZR
30. Pal, A., Charkhgard, H.: A feasibility pump and local search based heuristic for bi-objective pure integer linear programming. INFORMS Journal on Computing **31**(1), 115–133 (2019)
31. Pal, A., Charkhgard, H.: Fpbh: A feasibility pump based heuristic for multi-objective mixed integer linear programming. Computers & Operations Research **112**, 104760 (2019)
32. Soylu, B.: Heuristic approaches for biobjective mixed 0–1 integer linear programming problems. European Journal of operational research **245**(3), 690–703 (2015)
33. Steuer, R.E.: The tchebycheff procedure of interactive multiple objective programming. In: Multiple criteria decision making and risk analysis using microcomputers, pp. 235–249. Springer (1989)
34. Steuer, R.E., Choo, E.U.: An interactive weighted tchebycheff procedure for multiple objective programming. Mathematical programming **26**(3), 326–344 (1983)
35. Tamby, S., Vanderpooten, D.: Enumeration of the nondominated set of multiobjective discrete optimization problems. INFORMS Journal on Computing **33**(1), 72–85 (2021)
36. Tricoire, F.: Multi-directional local search. Computers & operations research **39**(12), 3089–3101 (2012)
37. Wu, Y., Song, W., Cao, Z., Zhang, J., Gupta, A., Lin, M.: Graph learning assisted multi-objective integer programming. Advances in Neural Information Processing Systems **35**, 17774–17787 (2022)
38. Zadeh, L.: Optimality and non-scalar-valued performance criteria. IEEE transactions on Automatic Control **8**(1), 59–60 (1963)
39. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation **11**(6), 712–731 (2007). https://doi.org/10.1109/TEVC.2007.892759
40. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation **3**(4), 257–271 (1999). https://doi.org/10.1109/4235.797969
41. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation **7**(2), 117–132 (2003). https://doi.org/10.1109/TEVC.2003.810758
42. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. Parallel Problem Solving from Nature (PPSN VIII) pp. 832–842 (2004). https://doi.org/10.1007/978-3-540-30217-9_84

## A Instance Generation

*MOKP* For the MOKP, we consider $n$ items, each characterized by a weight $w_i$ and a profit $p_{ik}$ corresponding to objective $k$. We generate instances by sampling weights and profits uniformly from $[1, 1000]$. The knapsack capacity is fixed at $C = \lceil 0.5 \sum_{i=1}^{n} w_i \rceil$. Letting $x_i \in \{0, 1\}$ indicate the selection of item $i$, the goal is to maximize the vector of total profits without exceeding the capacity:

$$\max_{\mathbf{x}\in\{0,1\}^n} \quad (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})), \quad \text{where } f_k(\mathbf{x}) = \sum_{i=1}^{n} p_{ik} x_i, \tag{11}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} w_i x_i \leq C. \tag{12}$$

*MOAP* The MOAP involves finding a one-to-one mapping between $n$ agents and $n$ tasks. The cost $c_{ijk}$ associated with assigning agent $i$ to task $j$ for objective $k$ is drawn uniformly from $[1, 20]$. With binary variables $x_{ij}$ representing the assignments, the problem seeks to minimize the cost vector:

$$\min_{\mathbf{x}\in\{0,1\}^{n\times n}} \quad (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})), \quad \text{where } f_k(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ijk} x_{ij}, \tag{13}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in \{1, \ldots, n\}, \tag{14}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \in \{1, \ldots, n\}. \tag{15}$$

## B Hyperparameter Configuration

The hyperparameter settings and problem-specific time limits are summarized in Table 4 and Table 5, respectively.

## C Additional results

This section provides the complete experimental data for all problem configurations, supplementing the representative results discussed in the main text. Table 6 presents the comprehensive HV statistics, averaged across all 25 instances and 5 random seeds, serving as an extension to Table 1. Similarly, the full win-rate analysis for every problem size and objective dimension is detailed in Table 8, extending the results in Table 2.

We also extend the analysis of the parallel (multi-start) setting. Figure 3 illustrates the progression of the cumulative HV as the number of independent

Table 4: Hyperparameter settings for different methods.

| Method | Hyperparameter | Value |
|---|---|---|
| BPO | Surrogate Model | SingleTaskGP |
| | Query Batch Size ($q$) (`batch_size_q`) | 4 |
| | Optimization Restarts (`num_restarts`) | 5 |
| | Initialization (Raw) Samples (`raw_samples`) | 64 |
| | Monte Carlo (MC) Samples (`mc_samples`) | 128 |
| NSGA-II (MOKP) | Sampling | BinaryRandomSampling |
| | Crossover | SinglePointCrossover |
| | Mutation | BitflipMutation |
| | Population size | 200 |
| NSGA-II (MOAP) | Sampling | PermutationRandomSampling |
| | Crossover | OrderCrossover |
| | Mutation | InversionMutation |
| | Population size | 200 |
| ATS | Augmentation penalty $\rho$ | 1e-4 |

Table 5: Time limit across different problems and sizes.

| Problem | Time limit (s) | Size (m, n) |
|---|---|---|
| MOKP | 240 | (3, 1000), (4, 750), (5, 500), (6, 500) |
| | 480 | (3, 1250), (4, 1000), (5, 750) |
| MOAP | 240 | (3, 175), (4, 100), (6, 50) |
| | 480 | (3, 200), (4, 125), (5, 75), (7, 50) |

seeds increases, expanding on Figure 2. The corresponding quantitative comparison against the empirical Oracle for all problem sizes is provided in Table 7. Notably, while BPO-150 shows a lower average HV than ATS for the 6-objective case in the single-seed analysis (Table 6), it outperforms ATS in the multi-start setting (Table 7). This suggests that in higher-dimensional objective spaces, the diversity introduced by independent parallel runs effectively compensates for the exploration limitations of a single surrogate model.

Table 6: Final HV results on MOKP and MOAP across all problem sizes.

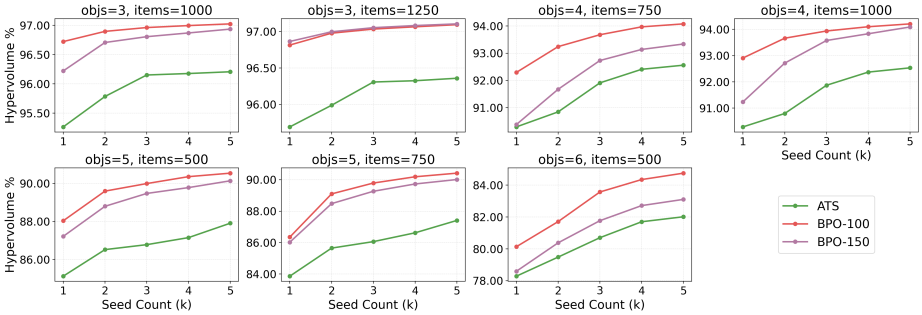
| MOKP | | | | | | | | | MOAP | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Hypervolume | | | | | | | | | Hypervolume | | | | | |
| $m$ | $n$ | Method | Mean. | Std. | Min. | Max. | Iter. | $\lvert\hat{\mathcal{Y}}_N\rvert$ | $m$ | $n$ | Method | Mean. | Std. | Min. | Max. | Iter. | $\lvert\hat{\mathcal{Y}}_N\rvert$ |
| 3 | 1000 | KSA | 64.68 | 1.47 | 62.12 | 68.07 | 504 | 502 | 3 | 175 | KSA | 70.03 | 1.45 | 66.35 | 72.54 | 469 | 468 |
| | | NSGA-II | 85.10 | 0.79 | 83.36 | 87.50 | – | 200 | | | NSGA-II | 46.07 | 1.32 | 43.55 | 50.41 | – | 200 |
| | | ATS | 95.15 | 0.60 | 92.68 | 96.30 | 161 | 160 | | | ATS | 95.46 | 0.68 | 92.37 | 96.39 | 127 | 127 |
| | | **BPO-100** | **96.69** | 0.25 | 96.00 | 97.26 | 194 | 126 | | | **BPO-100** | **96.52** | 0.74 | 92.50 | 97.42 | 136 | 127 |
| | | BPO-150 | 95.90 | 1.05 | 93.27 | 97.32 | 159 | 147 | | | BPO-150 | 95.81 | 0.69 | 93.19 | 97.33 | 133 | 130 |
| 3 | 1250 | KSA | 64.75 | 1.65 | 62.45 | 68.24 | 708 | 706 | 3 | 200 | KSA | 74.11 | 1.19 | 70.93 | 76.64 | 626 | 625 |
| | | NSGA-II | 85.57 | 0.58 | 84.15 | 86.80 | – | 200 | | | NSGA-II | 47.68 | 1.25 | 45.00 | 51.86 | – | 200 |
| | | ATS | 95.49 | 0.42 | 94.18 | 96.24 | 217 | 215 | | | ATS | 96.15 | 1.99 | 74.78 | 97.14 | 185 | 185 |
| | | **BPO-100** | **96.80** | 0.20 | 96.29 | 97.23 | 362 | 159 | | | **BPO-100** | **97.36** | 0.37 | 95.93 | 97.80 | 193 | 163 |
| | | BPO-150 | 96.73 | 0.57 | 94.18 | 97.28 | 296 | 183 | | | BPO-150 | 97.01 | 0.71 | 95.05 | 97.88 | 184 | 173 |
| 4 | 750 | KSA | 49.13 | 1.88 | 46.21 | 52.74 | 137 | 134 | 4 | 100 | KSA | 33.79 | 2.31 | 28.04 | 37.97 | 175 | 174 |
| | | NSGA-II | 78.51 | 1.03 | 75.25 | 80.37 | – | 200 | | | NSGA-II | 31.30 | 0.86 | 29.51 | 34.47 | – | 200 |
| | | ATS | 90.29 | 1.09 | 80.85 | 92.04 | 124 | 124 | | | ATS | 84.15 | 0.64 | 82.39 | 85.98 | 152 | 152 |
| | | **BPO-100** | **92.31** | 1.33 | 88.70 | 93.94 | 127 | 111 | | | **BPO-100** | **85.16** | 0.78 | 82.10 | 86.86 | 156 | 149 |
| | | BPO-150 | 90.76 | 1.10 | 88.53 | 93.57 | 115 | 113 | | | BPO-150 | 84.75 | 1.05 | 82.06 | 86.82 | 152 | 152 |
| 4 | 1000 | KSA | 48.79 | 1.89 | 44.71 | 53.05 | 171 | 167 | 4 | 125 | KSA | 37.97 | 1.96 | 34.06 | 41.54 | 212 | 211 |
| | | NSGA-II | 77.81 | 0.89 | 75.59 | 79.50 | – | 200 | | | NSGA-II | 31.12 | 1.01 | 28.54 | 35.41 | – | 200 |
| | | ATS | 90.26 | 0.66 | 88.05 | 92.00 | 119 | 119 | | | ATS | 86.49 | 0.66 | 84.64 | 88.04 | 182 | 182 |
| | | **BPO-100** | **92.78** | 1.07 | 88.05 | 94.09 | 172 | 121 | | | **BPO-100** | **87.37** | 0.76 | 84.02 | 88.97 | 221 | 197 |
| | | BPO-150 | 91.70 | 1.47 | 87.14 | 93.83 | 139 | 127 | | | BPO-150 | 87.24 | 1.03 | 83.93 | 89.18 | 179 | 176 |
| 5 | 500 | KSA | 38.23 | 2.43 | 31.93 | 43.62 | 52 | 48 | 5 | 75 | KSA | 14.11 | 1.20 | 11.83 | 16.24 | 72 | 71 |
| | | NSGA-II | 73.59 | 1.58 | 69.34 | 77.50 | – | 200 | | | NSGA-II | 21.03 | 0.82 | 18.51 | 23.47 | – | 200 |
| | | ATS | 84.76 | 1.28 | 80.69 | 87.93 | 173 | 173 | | | ATS | 70.64 | 1.33 | 67.22 | 73.51 | 261 | 261 |
| | | **BPO-100** | **88.28** | 1.06 | 85.33 | 90.71 | 173 | 134 | | | BPO-100 | 69.62 | 1.19 | 66.13 | 72.67 | 216 | 193 |
| | | BPO-150 | 86.87 | 2.12 | 81.32 | 90.77 | 172 | 162 | | | **BPO-150** | **70.73** | 1.18 | 67.28 | 73.06 | 220 | 209 |
| 5 | 750 | KSA | 37.56 | 2.49 | 31.88 | 42.07 | 61 | 58 | 6 | 50 | KSA | 6.50 | 0.86 | 4.90 | 8.02 | 31 | 28 |
| | | NSGA-II | 71.51 | 1.27 | 68.63 | 74.56 | – | 200 | | | NSGA-II | 13.78 | 0.71 | 11.90 | 15.70 | – | 200 |
| | | ATS | 83.78 | 1.17 | 80.20 | 86.17 | 115 | 115 | | | **ATS** | **52.01** | 1.95 | 47.21 | 56.05 | 202 | 202 |
| | | **BPO-100** | **87.42** | 2.07 | 81.36 | 89.93 | 166 | 121 | | | BPO-100 | 50.32 | 1.48 | 46.47 | 53.12 | 147 | 141 |
| | | BPO-150 | 85.86 | 2.08 | 81.36 | 89.78 | 136 | 125 | | | BPO-150 | 51.60 | 1.69 | 47.42 | 54.62 | 167 | 166 |
| 6 | 500 | KSA | 29.49 | 2.54 | 22.89 | 35.20 | 26 | 24 | 7 | 50 | KSA | 2.88 | 0.00 | 2.88 | 2.88 | 26 | 18 |
| | | NSGA-II | 65.10 | 1.85 | 58.91 | 69.45 | – | 200 | | | NSGA-II | 8.24 | 0.49 | 7.07 | 9.65 | – | 200 |
| | | ATS | 77.75 | 1.44 | 73.15 | 80.60 | 118 | 118 | | | **ATS** | **39.89** | 2.26 | 34.94 | 46.77 | 233 | 233 |
| | | **BPO-100** | **79.80** | 2.83 | 72.26 | 84.90 | 113 | 107 | | | BPO-100 | 37.13 | 1.51 | 34.44 | 42.21 | 170 | 159 |
| | | BPO-150 | 78.26 | 1.82 | 74.22 | 84.80 | 116 | 115 | | | BPO-150 | 38.73 | 1.70 | 34.85 | 43.95 | 187 | 182 |

Table 7: Comparison of parallel (k=5) HV results with Oracle.

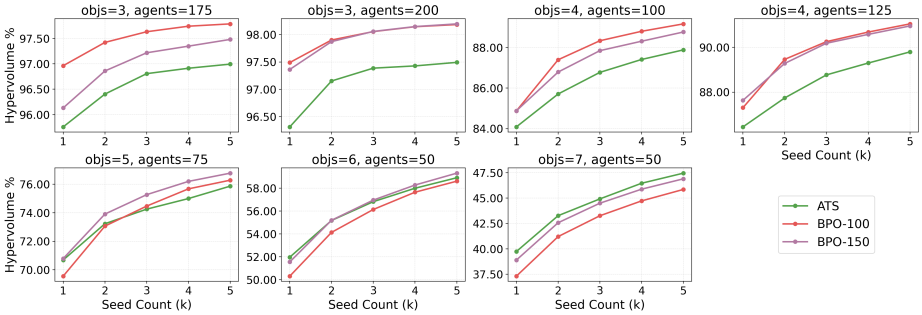
| MOKP | | | | | | | | MOAP | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | Hypervolume | | | | | | | | Hypervolume | | | | |
| $m$ | $n$ | Method | Mean. | Std. | Min. | Max. | $\lvert\hat{\mathcal{Y}}_N\rvert$ | $m$ | $n$ | Method | Mean. | Std. | Min. | Max. | $\lvert\hat{\mathcal{Y}}_N\rvert$ |
| 3 | 1000 | ATS | 96.21 | 0.25 | 95.78 | 96.72 | 791 | 3 | 175 | ATS | 96.99 | 0.17 | 96.54 | 97.28 | 629 |
| | | **BPO-100** | **97.02** | 0.23 | 96.63 | 97.50 | 613 | | | **BPO-100** | **97.79** | 0.12 | 97.45 | 97.98 | 606 |
| | | BPO-150 | 96.94 | 0.34 | 95.78 | 97.48 | 721 | | | BPO-150 | 97.48 | 0.30 | 96.88 | 97.95 | 642 |
| | | Oracle | 97.07 | 0.22 | 96.69 | 97.54 | 1434 | | | Oracle | 97.89 | 0.12 | 97.63 | 98.10 | 1240 |
| 3 | 1250 | ATS | 96.36 | 0.20 | 95.94 | 96.74 | 1067 | 3 | 200 | ATS | 97.49 | 0.11 | 97.27 | 97.68 | 916 |
| | | BPO-100 | 97.09 | 0.18 | 96.75 | 97.46 | 771 | | | BPO-100 | 98.18 | 0.08 | 98.01 | 98.30 | 740 |
| | | **BPO-150** | **97.10** | 0.18 | 96.77 | 97.47 | 895 | | | **BPO-150** | **98.20** | 0.09 | 97.98 | 98.35 | 831 |
| | | Oracle | 97.15 | 0.18 | 96.81 | 97.51 | 2180 | | | Oracle | 98.31 | 0.08 | 98.14 | 98.43 | 1848 |
| 4 | 750 | ATS | 92.56 | 0.39 | 91.76 | 93.23 | 620 | 4 | 100 | ATS | 87.88 | 0.37 | 87.13 | 88.66 | 760 |
| | | **BPO-100** | **94.07** | 0.52 | 92.54 | 94.78 | 544 | | | **BPO-100** | **89.16** | 0.34 | 88.49 | 89.85 | 724 |
| | | BPO-150 | 93.34 | 0.69 | 91.70 | 94.44 | 562 | | | BPO-150 | 88.76 | 0.56 | 87.54 | 89.62 | 758 |
| | | Oracle | 94.17 | 0.50 | 92.61 | 94.85 | 836 | | | Oracle | 89.71 | 0.37 | 89.08 | 90.52 | 1202 |
| 4 | 1000 | ATS | 92.53 | 0.34 | 91.91 | 93.21 | 595 | 4 | 125 | ATS | 89.80 | 0.35 | 88.84 | 90.60 | 908 |
| | | **BPO-100** | **94.21** | 0.32 | 93.55 | 94.76 | 593 | | | **BPO-100** | **91.05** | 0.34 | 90.23 | 91.83 | 929 |
| | | BPO-150 | 94.09 | 0.37 | 93.08 | 94.60 | 632 | | | BPO-150 | 90.96 | 0.50 | 89.56 | 91.98 | 871 |
| | | Oracle | 94.30 | 0.31 | 93.67 | 94.82 | 922 | | | Oracle | 91.70 | 0.37 | 90.68 | 92.51 | 1639 |
| 5 | 500 | ATS | 87.91 | 0.83 | 86.09 | 89.68 | 866 | 5 | 75 | ATS | 75.87 | 0.90 | 73.92 | 77.58 | 1305 |
| | | **BPO-100** | **90.54** | 0.74 | 88.73 | 92.09 | 651 | | | BPO-100 | 76.28 | 0.86 | 74.37 | 78.01 | 911 |
| | | BPO-150 | 90.14 | 1.04 | 88.39 | 92.07 | 804 | | | **BPO-150** | **76.78** | 0.92 | 74.61 | 78.70 | 1023 |
| | | Oracle | 90.81 | 0.78 | 88.96 | 92.40 | 1156 | | | Oracle | 78.34 | 0.95 | 76.31 | 80.23 | 1985 |
| 5 | 750 | ATS | 87.41 | 0.80 | 85.94 | 88.72 | 577 | 6 | 50 | ATS | 58.91 | 1.53 | 55.38 | 61.70 | 1009 |
| | | **BPO-100** | **90.42** | 0.56 | 89.51 | 91.50 | 594 | | | BPO-100 | 58.61 | 1.33 | 55.50 | 60.87 | 685 |
| | | BPO-150 | 90.01 | 0.69 | 87.49 | 91.19 | 623 | | | **BPO-150** | **59.31** | 1.55 | 55.60 | 61.79 | 826 |
| | | Oracle | 90.60 | 0.53 | 89.69 | 91.67 | 822 | | | Oracle | 61.04 | 1.60 | 57.24 | 63.70 | 1283 |
| 6 | 500 | ATS | 82.02 | 0.96 | 79.86 | 83.81 | 591 | 7 | 50 | **ATS** | **47.46** | 1.82 | 44.37 | 52.95 | 1164 |
| | | **BPO-100** | **84.75** | 1.72 | 80.73 | 87.53 | 529 | | | BPO-100 | 45.86 | 1.52 | 43.59 | 50.40 | 773 |
| | | BPO-150 | 83.11 | 1.31 | 81.45 | 87.38 | 577 | | | BPO-150 | 46.91 | 1.59 | 44.13 | 51.66 | 903 |
| | | Oracle | 85.26 | 1.51 | 81.66 | 87.96 | 761 | | | Oracle | 49.06 | 1.92 | 45.99 | 54.80 | 1564 |

Table 8: Win-rate comparison: BPO vs. ATS.

| MOKP | | | | | MOAP | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $m$ | $n$ | BPO-100 | BPO-150 | ATS | $m$ | $n$ | BPO-100 | BPO-150 | ATS |
| 3 | 1000 | **63.20** | 36.80 | 0.00 | 3 | 175 | **85.20** | 8.40 | 6.40 |
| 3 | 1250 | 25.60 | **74.40** | 0.00 | 3 | 200 | **62.00** | 36.40 | 1.60 |
| 4 | 750 | **77.60** | 12.80 | 9.60 | 4 | 100 | **64.00** | 33.20 | 2.80 |
| 4 | 1000 | **72.40** | 26.40 | 1.20 | 4 | 125 | **51.20** | 46.40 | 2.40 |
| 5 | 500 | **74.40** | 25.60 | 0.00 | 5 | 75 | 1.60 | **59.20** | 39.20 |
| 5 | 750 | **74.40** | 24.40 | 1.20 | 6 | 50 | 3.20 | 21.20 | **75.60** |
| 6 | 500 | **64.80** | 16.40 | 18.80 | 7 | 50 | 0.00 | 8.00 | **92.00** |




(a) HV progression over number of seeds for MOKP.




(b) HV progression over number of seeds for MOAP.

Fig. 3: HV progression over seeds for (a) MOKP and (b) MOAP, averaged across 25 problem instances.