GraphAllocBench: A Flexible Benchmark for Preference-Conditioned Multi-Objective Policy Learning

Zhiheng Jiang[1] , **Yunzhe Wang**[2] , **Ryan Marr**[2] , **Ellen Novoseller**[3] , **Benjamin T. Files**[3] and **Volkan Ustun**[2]
[1]University of California, Los Angeles
[2]USC Institute for Creative Technologies
[3]U.S. Army DEVCOM Army Research Laboratory
zhjiang@g.ucla.edu, {wyunzhe, ustun}@ict.usc.edu, {ellen.r.novoseller, benjamin.t.files}.civ@army.mil

Abstract

Preference-Conditioned Policy Learning (PCPL) in Multi-Objective Reinforcement Learning (MORL) approximates diverse Pareto-optimal solutions by conditioning a single policy on user-specified preferences, enabling run-time adaptation to arbitrary trade-offs without retraining. However, existing PCPL benchmarks are largely restricted to toy tasks and fixed environments, limiting their realism and scalability. To address this gap, we introduce GraphAllocBench, a flexible benchmark built on CityPlannerEnv, a novel graph-based resource allocation sandbox inspired by city management. GraphAllocBench provides a rich suite of problems with customizable objective functions, varying preference conditions, complex Pareto Fronts, and high-dimensional scalability. We further propose two supplementary metrics – Proportion of Non-Dominated Solutions (PNDS) and Ordering Score (OS) – that capture prediction reliability and preference consistency while complementing the widely used hypervolume metric. Through experiments with several state-of-the-art PCPL algorithms and our own MLP and graph-aware PCPL-PPO baseline, we show that GraphAllocBench exposes distinct failure modes that hypervolume alone does not capture but our supplementary metrics reveal, while motivating graph-based approaches such as Graph Neural Networks (GNNs) for scaling to complex, high-dimensional allocation tasks. By letting users freely vary objectives, preferences, and allocation rules, GraphAllocBench serves as a versatile and extensible testbed for advancing PCPL. Code: https://github.com/jzh001/GraphAllocBench

1 Introduction

Reinforcement learning (RL) has achieved remarkable progress in domains such as video games [Shao *et al.*, 2019], robotics [Labiosa *et al.*, 2025], and large language model fine-tuning [Ouyang *et al.*, 2022]. Much of this success is built on single-objective RL algorithms such as Proximal Policy Optimization (PPO) [Schulman *et al.*, 2017], which optimize a single scalar reward function.

However, many real-world and simulated environments involve multiple, often conflicting objectives rather than a single scalar reward. For example, in games and training simulations, Non-Player Characters (NPCs) must balance dealing damage, avoiding damage, and maximizing survival time [Schrum and Miikkulainen, 2008] to create engaging and adaptive experiences. In such cases, objectives carry different levels of importance; we term such weights *preferences* over objectives.

A common approach to multi-objective RL (MORL) is scalarization [Van Moffaert *et al.*, 2013] (See Appendix D.3) , where objectives are collapsed into a single reward using preference weights. While straightforward, using a predetermined scalarization ties the learned policy to one fixed preference and requires retraining for new ones.

Other existing approaches to MORL (See Section 2.1) include training populations of policies via evolutionary algorithms [Deb and Jain, 2014; Dixit and Tumer, 2025], particle swarm approaches [Becerril Torres *et al.*, 2025] or training multiple single-objective models to approximate the Pareto front [Xu *et al.*, 2020]. These methods are generally inefficient, as each new preference requires retraining from scratch, or requires keeping track of multiple concurrent policies. Gradient-based optimization with vector-valued rewards [Désidéri, 2012] has also been explored, but often suffers from convergence issues [Liu *et al.*, 2021].

Preference-conditioned Policy Learning (PCPL) in MORL, for instance PD-MORL [Basaklar *et al.*, 2023], offers a more scalable solution. Instead of training a separate model for each preference, a single preference-conditioned policy can adapt to arbitrary user-specified preferences. Compared to alternatives such as role embeddings [Long *et al.*, 2024] or policy pools [Garnelo *et al.*, 2021], preference-conditioned methods provide a more direct and flexible mechanism for handling diverse objectives.

Despite algorithmic progress in the field, current multi-objective benchmarks [Felten *et al.*, 2023] are insufficient for pushing progress forward for PCPL. Most existing testbeds utilize either continuous optimization problems [Deb *et al.*, 2005; Huband *et al.*, 2006] or simple 2D grid-

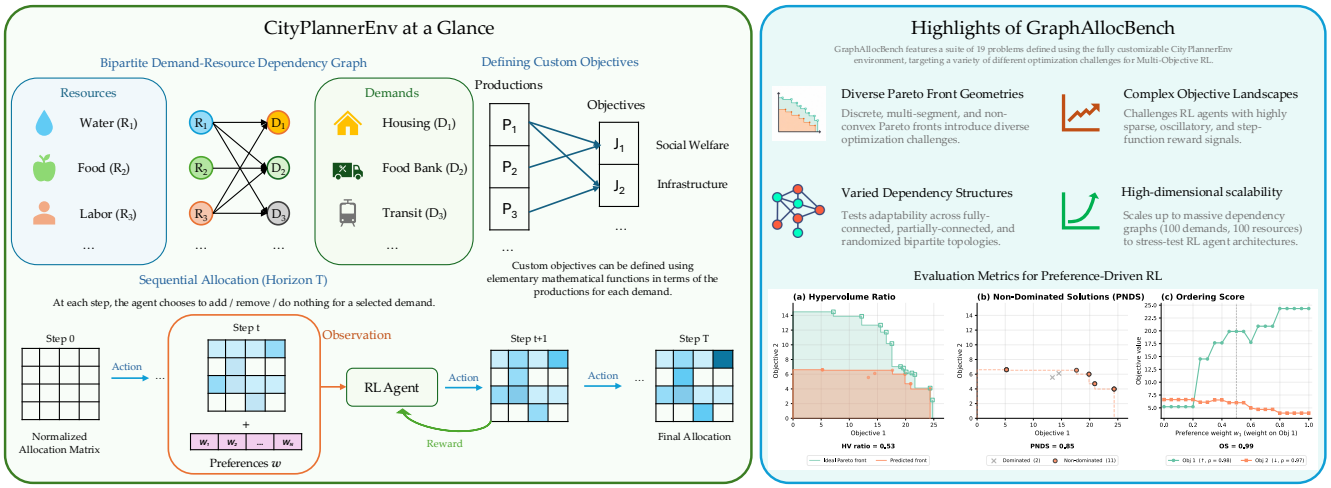


Figure 1: CityPlannerEnv models city resource allocation as sequential decision-making over a bipartite demand-resource graph, with composable multi-objective rewards and preference-conditioned policies. GraphAllocBench provides a diverse suite of problems spanning varied graph structures, Pareto front geometries, and scales, built on top of CityPlannerEnv.

based games [Yang *et al.*, 2019; Vamplew *et al.*, 2011]. These lack flexibility in scaling observations, actions, objectives and dependency structures, making them insufficient to evaluate the robustness of preference-conditioned methods in more complex problems. For instance, in Deep Sea Treasure [Vamplew *et al.*, 2011], the objectives are to maximize treasure and minimize time, as limited by the game mechanics.

To address this gap, we introduce **CityPlannerEnv**, a Gymnasium-based [Towers *et al.*, 2024] sandbox environment inspired by city-scale resource allocation that enables multi-objective planning in graph-based environments of arbitrary and scalable complexity. Cities must allocate limited resources across diverse and often conflicting demands – such as reducing congestion, fostering economic growth, and promoting sustainability – according to shifting preferences. The environment models this process as a sequential allocation problem, represented through bipartite resource-demand dependency graphs. Agents must not only balance competing objectives but also anticipate future events and adapt to varying risk profiles (e.g., conserving resources for emergencies versus aggressively pursuing short-term gains). Although framed around city planning, the bipartite demand–resource structure is the abstraction underlying many allocation problems such as transportation, human resources and supply-chain distribution.

In **CityPlannerEnv**, agents allocate resources by adding or removing productions for each demand at each step to pursue custom-defined objectives based on user preferences. This incremental allocation mechanism draws inspiration from the Multi-step Colonel Blotto Game (MCBG), a well-studied model for competitive resource allocation on graphs [An and Zhou, 2025b; An and Zhou, 2025a; Shishika *et al.*, 2022]. Dependencies between resources and demands are represented as bipartite graphs, and objectives can be defined over any subset of productions. Agents seek to plan actions that maximize the expected cumulative reward over time while respecting preference trade-offs. As the number of resources, demands, and objectives increases – and as conditions change over time – the allocation problem becomes increasingly challenging for PCPL.

We further introduce **GraphAllocBench**, a benchmark for PCPL that defines a suite of challenging problems designed using CityPlannerEnv. GraphAllocBench leverages the flexibility of CityPlannerEnv to create customizable problems by varying the numbers of demands, resources, objectives, and objective shapes, providing targeted stress tests for PCPL algorithms. For example, GraphAllocBench includes test problems spanning diverse optimization challenges, including difficult objective functions, diverse and non-convex Pareto fronts, sparse observation spaces, complex dependency structures, and high-dimensional graph observations. For performance characterization, the benchmark includes the hypervolume ratio along with two novel supplementary metrics – Proportion of Non-Dominated Solutions (PNDS) and Ordering Score (OS) – to assess how well policies satisfy varied preferences. Figure 1 shows an overview of CityPlannerEnv and GraphAllocBench.

To establish baseline performance and highlight how the benchmark challenges existing methods, we demonstrate a PPO-based [Schulman *et al.*, 2017] PCPL solution (PCPL-PPO) with Multi-Layer Perceptron (MLP) and Heterogeneous Graph Neural Network (HGNN) [Hu *et al.*, 2020] feature extractors. We further employ Smooth Tchebycheff Scalarization [Lin *et al.*, 2024] to approximate challenging non-convex Pareto fronts and demonstrate how graph neural networks can learn preference-conditioned, graph-aware policies in environments with complex dependency structures, going beyond prior work that primarily focused on 2D grids [Basaklar *et al.*, 2023] or simpler graph topologies [Lin *et al.*, 2022; Fu and Gu, 2024].

Our contributions are as follows:

- **GraphAllocBench benchmark:** We introduce *GraphAllocBench*, a flexible and challenging graph-based benchmark for preference-conditioned policy learning, based on our novel, customizable sandbox environment *CityPlannerEnv*, designed to capture complex resource allocation scenarios with complex bipartite dependency structures, discrete non-convex Pareto Fronts, and difficult objective functions.
- **Novel evaluation metrics:** Beyond the standard hypervolume metric, we introduce the *ordering score* (OS) and the *proportion of non-dominated solutions* (PNDS) to evaluate preference-consistency and robustness, addressing limitations of existing multi-objective evaluation practices for PCPL algorithms (see Section 2).
- **PCPL-PPO with MLPs and HGNNs:** We use MLP-based PCPL-PPO as a strong baseline and introduce a preference-conditioned HGNN-based PCPL-PPO that substantially improves hypervolume coverage on high-dimensional graph observations, while exposing a measurable trade-off between coverage and per-objective preference alignment that remains an open challenge.

2 Background

2.1 Multi-Objective RL

Many solutions in the multi-objective optimization space have aimed to tackle the issue of learning Pareto Fronts (See Appendix A). Evolutionary methods such as NSGA-III [Deb and Jain, 2014] have shown success, focusing on generating increasingly Pareto-optimal populations of solutions with each generation by performing operations such as selection, crossover and mutation. However, while these evolutionary methods are effective for simpler multi-objective optimization tasks, they are less adaptable towards more complex and high-dimensional tasks, such as graphs, and struggle to maintain a set of diverse solutions [Gu *et al.*, 2022].

More importantly, popular multi-objective optimization algorithms including evolutionary methods often aim to produce a set of solutions on the Pareto Front, making it difficult to specify a preference for a particular solution directly as input. Learning and maintaining a large population of Pareto-optimal solutions can be computationally and memory intensive, especially for higher-dimensional objective spaces. Hence for a large number of objectives, it becomes difficult to maintain a good resolution of different solutions on the Pareto Front.

For Pareto-Optimal Policy Set Learning in RL, Multiple-Gradient Descent Algorithms (MGDA) [Désidéri, 2012] have been introduced to approximate the Pareto Front. These algorithms involve iteratively updating variables such that all objectives are improved at the same time. However, these methods do not allow us to easily specify preferences for solutions with different tradeoffs, and face convergence issues for some segments on the Pareto Front [Liu *et al.*, 2021].

Some work has been conducted to learn preference-aware Pareto-optimal policies using RL. One such approach is to train multiple single-objective RL models on a set of fixed preference vectors, and select a corresponding model to use during inference, based on a specified preference. A variant of this method uses an evolutionary algorithm with a population of different RL policies [Xu *et al.*, 2020]. A limitation of this method is that we can only learn a discrete set of preferences, as we need to train a separate RL model for each fixed preference vector. Additionally, these methods struggle with similar challenges to traditional multi-objective optimization tasks, as they still involve generating a population of Pareto-optimal solutions instead of an individual preference-aware solution.

Unlike conventional multi-objective approaches that generate a complete Pareto Front, Preference-Conditioned Policy Learning (PCPL) methods [Basaklar *et al.*, 2023; Lin *et al.*, 2022; Liu *et al.*, 2025; Yang *et al.*, 2019] use a continuous, user-defined preference as input to a single policy. This allows them to produce a specific, preference-aware solution on the Pareto Front via a series of sequential decisions within the environment. The final reward can be computed using the scalarization (See Appendix D.3) of the components of the objective function, based on the preference provided during input. This allows us to learn a preference-conditioned policy, where during inference, we can specify any preference vector within a continuous range and obtain an approximation of the Pareto-optimal solution on the Pareto Front.

2.2 Benchmarking MORL

Several benchmarks have been proposed for multi-objective optimization and reinforcement learning. For multi-objective optimization problems, examples include DTLZ [Deb *et al.*, 2005] and WFG [Huband *et al.*, 2006], which are primarily traditional multi-objective optimization problems, where we are varying decision variables to approximate mostly continuous objective functions and Pareto Fronts. These benchmarks offer expandable and flexible testcases for multi-objective optimization. However, these benchmarks are not easily adaptable for MORL tasks involving sequential decision making, non-continuous Pareto Fronts, or high dimensional observation spaces such as graphs.

Common benchmarks for MORL include game environments, for instance those in Multi-Objective Gymnasium [Felten *et al.*, 2023] such as Deep Sea Treasure (DST) [Vamplew *et al.*, 2011] and Fruit Tree Navigation (FTN) [Yang *et al.*, 2019]. These are game environments, e.g. where an agent tries to accomplish multiple objectives in a 2D grid world or MuJoCo task with continuous action spaces [Zhu *et al.*, 2023]. However, these benchmarks are still limited to the mechanics of the game environment itself (limited action and observation spaces), and it is difficult to design custom scenarios and objective functions to challenge the RL model. Even though variants of these problems [Cassimon *et al.*, 2021] have also been introduced to increase the size of the observation space and number of objectives, these variants are still non-expandable and normally focus on small action and observation spaces, with limited number of objectives, often just 2-3 objectives [Zhu *et al.*, 2023].

GraphAllocBench addresses these gaps along three axes that existing benchmarks lack: (1) customizable objectives, enabling deliberate design of Pareto Front geometry; (2) scalable bipartite graph structure, exposing the limits of MLP-

based approaches; and (3) complementary evaluation metrics (PNDS and OS) that go beyond hypervolume: PNDS measures prediction reliability, while OS is a preference-specific metric that measures preference alignment.

3 CityPlannerEnv

Our sandbox environment **CityPlannerEnv** is modeled as a city that has a set of resources R and a set of demands D (Figure 1). At each step, the RL agent receives the current resource allocation state as an observation, takes an action to alter the allocation, and receives a multi-objective reward signal.

3.1 Definitions

Resource This refers to a particular factor of production, denoted by R_i, which can be combined with other resources to create productions (e.g. water, food, labor). The environment starts with a predefined number of units for each resource.

Demand A type of need fulfilled by resources (e.g. food bank), denoted by D_i.

Requirements Unweighted dependency set $R'(D_i) \subseteq R$ of the resources needed by each demand D_i.

Allocation Assignment of resources to demands, represented by allocation matrix A, where $A_{i,j}$ is the amount of R_i given to D_j. The observation returned by the environment is the normalized allocation matrix, concatenated with the current preferences vector for each rollout.

Production Actual output for each demand, determined by the limiting allocated resources. Production $\mathbf{P}$ changes when an action is taken to add or remove a production, where $|\mathbf{P}|$ is the number of demands.

Objective Function Multi-objective reward signal at the end of each step (i.e. after each action), defined by $\mathbf{J}(\mathbf{P}) : \mathbb{R}^{|\mathbf{P}|} \to \mathbb{R}^N$, mapping $|\mathbf{P}| = |D|$ productions to N objective values.

Preferences Importance assigned to each objective, represented by preferences vector $\mathbf{w} = [w_1, \ldots, w_N]$, $\sum_{n=1}^{N} w_n = 1$, $0 \leq w_i \leq 1$ for each $i \in \{1, \ldots, N\}$.

3.2 Action Space

Custom action spaces can be defined within the CityPlannerEnv environment to move resources between demands. In our solution, we define a single action as the selection of 2 sub-actions. Firstly, the agent will choose to either add or remove a production or not to modify the current allocation state. Secondly, the agent will choose a specific demand among the demands D for which to create 1 unit of production, remove 1 unit of production, or do nothing, depending on the first action chosen. When we add or remove a production, the corresponding resources tied to that demand will also be removed or added respectively from the unallocated resources pile.

CityPlannerEnv is implemented using Gymnasium [Towers *et al.*, 2024]. Custom environment configurations can be designed for the number of resource and demand types, the number of available resources for each type of resource, resource-demand dependency graphs, and objective functions using common mathematical functions (e.g. polynomial, sinusoidal, logarithmic).

3.3 Custom Objectives

Objective functions in CityPlannerEnv are fully user-definable: each component J_i can depend on any subset of productions P_j and can be composed from elementary functions (linear, quadratic, logarithmic, logistic, Gaussian, ceiling, and floor, or sums thereof). This composability allows practitioners to deliberately engineer Pareto Front geometries, including smooth convex fronts, discontinuous multi-segment fronts, and non-convex fronts with oscillatory structure. Table 3 in Appendix E illustrates representative objective functions spanning these cases. This design flexibility is a key differentiator from existing benchmarks, where objectives are fixed by game mechanics.

4 GraphAllocBench

GraphAllocBench comprises a set of 19 challenging evaluation problems designed using our sandbox environment CityPlannerEnv, across six challenge categories: Problem 0 serves as a baseline with a smooth, convex Pareto front; Problems 1a–1c introduce oscillatory and multi-segment objective landscapes with sparse rewards; Problems 2a–2c increase the number of demands, expanding the action and observation spaces; Problems 3a–3b scale to five objectives with highly sparse reward signals; Problems 4a–4b use non-fully-connected dependency graphs; and Problems 5a–5e feature randomly sampled dependency structures and objective functions at extended planning horizons. Three large-scale problems (6a–6c) further stress-test scalability on graphs with 100 demands. Full problem specifications are provided in Appendix E (Table 2).

This benchmark evaluates the quality of solutions using the hypervolume (HV) and Proportion of Non-Dominated Solutions (PNDS) using preferences sampled with the Das and Dennis method [Das and Dennis, 1998]. The solutions evaluated via the Ordering Score (OS) are sampled separately (See Algorithm 1 in Appendix C).

For each sampled preference, we roll out the agent's policy deterministically for T steps to obtain the final allocation of resources to demands. The Das and Dennis method [Das and Dennis, 1998] generates a set of evenly distributed preferences by constructing a structured grid of points on an N-dimensional simplex (for N objectives). Hence, the number of preferences sampled increases combinatorially with N. For scenarios where it is computationally feasible to compute the allocations that lead to solutions on the analytical Pareto Front, we compare the hypervolume of the set of predicted solutions for the sampled preferences with the ideal Pareto Front. The evaluation metrics are defined respectively in the following sections.

4.1 Hypervolume

For each sampled preference $\mathbf{w}$, we obtain a final allocation of resources, which allows us to calculate the final production vector $\mathbf{P}$ for a given allocation. Hence, we can use the final production state to calculate the value of each objective component, which are collectively represented by $\mathbf{J}(\mathbf{P})$. By varying our preferences, we obtain sets of objective vectors $\mathbf{J}$, which form a set of predicted optimal solutions (i.e. the predicted Pareto Front). We can evaluate this solution set using the hypervolume metric based on the maximization variant of the PyMOO [Blank and Deb, 2020] implementation (See Appendix B).

For simpler solutions, we can also calculate the ideal hypervolume of the environmental setup by enumerating all possible allocations. We can then calculate how well the predicted Pareto Front estimates the true Pareto Front using $\text{HV}_{\text{ratio}} = \text{HV}_{\text{predicted}}/\text{HV}_{\text{ideal}}$.

HV_{ratio} measures agent performance against the best possible solution and, being normalized, allows comparison across problems with differing objective counts (raw HV values are not comparable across dimensions). We prefer HV over metrics such as Generational Distance because it is more robust to the discontinuous Pareto Fronts common in our benchmark. For scenarios where the ideal hypervolume is not available due to computational complexity (high number of demands or objectives), we use the raw hypervolume to compare across different algorithms or architectures for each fixed task.

While hypervolume-based metrics can evaluate the overall quality and diversity of solutions, measuring the hypervolume becomes computationally expensive for high numbers of objectives, and may not provide the full picture in evaluating multi-objective problems [Ibrahim *et al.*, 2024], especially for PCPL. In particular, an agent may predict solutions close to the ideal Pareto Front some of the time, resulting in a high hypervolume, but the hypervolume metric does not indicate the reliability of single-shot predictions.

4.2 Proportion of Non-Dominated Solutions (PNDS)

To address the issue of prediction reliability, we also calculate the proportion of the predicted solutions from Section 4.1 that are non-dominated. This metric is independent of the ideal Pareto Front solution, which provides two main advantages:

1. It can be computed even when the ideal Pareto Front is intractable (e.g. many production combinations), and unlike purity [Custódio *et al.*, 2011], which scores against an external reference pooled across solvers, PNDS uses only a single policy's own predictions.

2. High PNDS with low hypervolume can indicate the agent is trapped in local optima.

While this metric is beneficial as a supplement to the hypervolume, it still struggles when the number of objectives is high due to sampling inefficiency and requires sampling exponentially more preferences using [Das and Dennis, 1998] with increasing N to ensure fair comparisons.

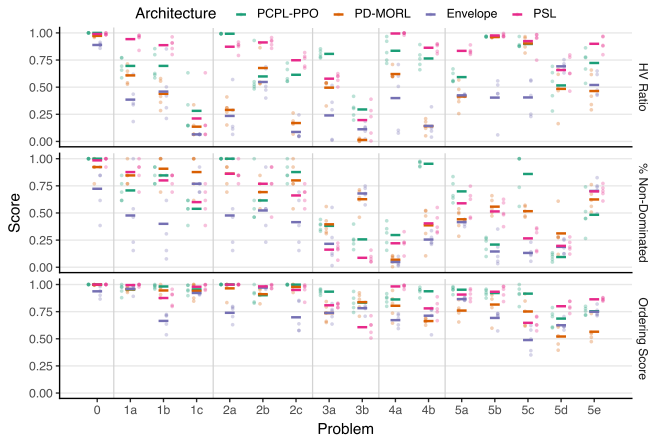


Figure 2: Performance comparison of PCPL-PPO, PD-MORL, Envelope Q-Learning and PSL policies over 5 random seeds at up to 1M steps for HV Ratio, PNDS and OS. Horizontal bars show the mean, and dots show the outcome for each seed.

4.3 Ordering Score (OS)

The ordering score measures the rank correlation between the preference weight w_i on each production i and the value of the objective function $\mathbf{J}_i(\mathbf{P})$. This evaluates how well the agent obeys the preference input, instead of whether the agent generates solutions close to the Pareto Front, which is measured by the hypervolume and PNDS. For traditional multi-objective optimization algorithms, we are often concerned with generating an unordered population of solutions which approximates the Pareto Front. However, for PCPL, the ordering score becomes an important way to evaluate how well an agent follows the preference it is given.

To compute the ordering score detailed in Algorithm 1 in Appendix C, we sample preferences that sweep along each objective and evaluate the corresponding objective values. An agent has performed well if the reward received in a particular objective increases with the preference weight in that component. We quantify this relationship using the Spearman Rank Correlation, normalized between 0 and 1 and averaged across all objective dimensions. The final ordering score is computed as the average Spearman Rank Correlation across all samples and all objectives.

The ordering score is a low-cost, scalable, magnitude-agnostic metric: it rewards the ordering of objective values rather than their levels. Metrics such as expected utility [Zintgraf *et al.*, 2015] account for preferences only indirectly, as they remain sensitive to objective magnitudes and the choice of scalarization; OS instead isolates preference alignment via a magnitude- and scalarization-free rank correlation. Being distribution-free, the Spearman correlation is robust to outliers and complements the magnitude-focused HV. Unlike the directional-angle training signal in PD-MORL, OS evaluates alignment at inference time. We thus view it as a supplementary preference-awareness diagnostic, used alongside HV and PNDS.

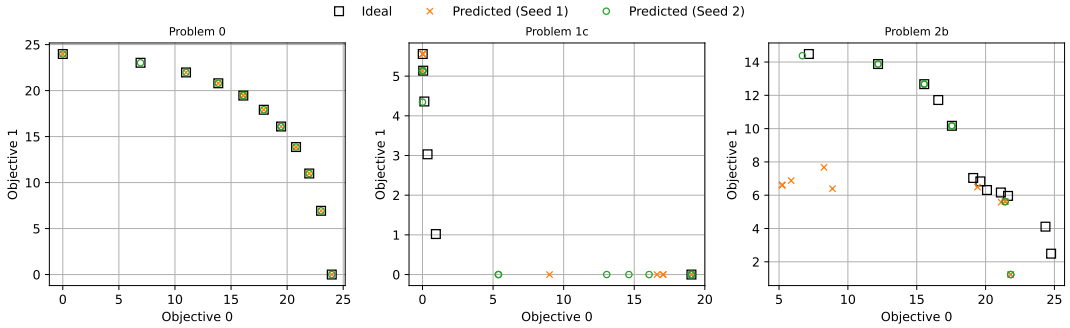


Figure 3: Selected 2-Objective Pareto Fronts for PCPL-PPO at 1M steps over 2 random seeds: Compared to the Problem 0 baseline, the PCPL-PPO agent struggles with non-convex (Problem 1c) and multi-segment (Problem 2b) Pareto Fronts.

5 Results and Discussion

5.1 Main Problem Set

Using the set of objective functions as our multi-objective reward function $\mathbf{J}$, we trained a preference-conditioned policy using Proximal Policy Optimization (PPO) on GraphAllocBench to convergence (See Figure 1). We will refer to our implementation as PCPL-PPO. We then employ the methods in Section 4 to evaluate the trained model. During training, preferences are sampled from a flat Dirichlet distribution for each rollout, and the PPO agent is trained to maximize the scalarized reward (using Smooth Tchebycheff Scalarization [Lin *et al.*, 2024]), given the sampled preference vector and current normalized allocation matrix as input. Each rollout lasts for a total of T steps. See Appendix D for more information on our PCPL-PPO implementation. Figure 2 compares PCPL-PPO against three other PCPL algorithms on problem categories 0-5: PD-MORL [Basaklar *et al.*, 2023], Envelope Q-Learning [Yang *et al.*, 2019] and Pareto Set Learning (PSL) [Liu *et al.*, 2025]. The benchmark's diverse problem set elicits distinct failure modes across methods, and the supplementary PNDS and OS metrics reveal preference-alignment gaps that hypervolume alone does not capture. PSL attains the strongest hypervolume across most problems, while PCPL-PPO leads on PNDS and is most robust on the hardest sparse and non-convex problems; the two are comparable on OS. Full results are in Appendix F.

These metrics are most informative where they disagree with hypervolume. On Problem 5c, PSL attains the highest hypervolume yet PCPL-PPO leads on both PNDS and ordering score; and on the five-objective problems 3a, 3b, and 4a, PSL's high-coverage sets yield few non-dominated solutions (PNDS 0.16, 0.09, 0.22). Strong hypervolume thus does not guarantee reliable, preference-aligned predictions — directly motivating PNDS and OS alongside HV.

Sharp Changes in Rewards (e.g. Problems 1a-c)
These sudden changes in reward signals make it difficult for an agent to learn a smooth approximation of the Pareto Front, and also lead to greater variance in predictions. From Figure 2, compared to the baseline (Problem 0) with a smooth objective function, Problems 1a-c report significantly higher variance and worse approximations of the Pareto Front despite sharing a dependency structure with the baseline, since sharp objective functions can result in large jumps between adjacent discrete points on the Pareto Front, which can be difficult to predict based on a smooth preference vector input.

Sparse Rewards (e.g. Problem 1c)
A PCPL agent may find it difficult to deal with sparse rewards, especially when optimizing a continuous preference vector. As example of this would be the $\mathbf{J}_i(\mathbf{P}) = \max(0, \mathbf{P}_j - 5)$ floor function, where we receive no reward for Objective i until we produce more than 5 units for Demand j. Instead of deciding whether to a) optimize a particular objective by creating productions until the reward spikes or to b) allocate no resources to the given production, the agent might create insufficient production, such that resources are wasted but no additional reward is received. This results in a lower proportion of non-dominated solutions, as observed for Problem 1c in Figure 2. In Figure 3, we also observe many predicted solutions along the horizontal axis where the reward component for Objective 1 is 0. However, there should only be one Pareto-optimal solution along the horizontal axis that maximizes Objective 0.

Non-convex Pareto Front (e.g. Problems 1c, 3b)
Non-convex Pareto Fronts are difficult to represent in general for multi-objective optimization or RL problems, and are highly dependent on the scalarization function used. Smooth Tchebycheff Scalarization is more robust to non-convexity than weighted sums, but still has limitations when the Pareto front is non-convex and discontinuous, as shown in Problem 1c and 3b (See Figures 2 and 3).

Unbalanced Objectives (e.g. Problem 2c)
Objectives with sparse reward signal can be difficult to optimize jointly with other concurrent objectives, which can result in failure to reach the sparser objective's extreme values and achieving a lower-quality approximation of intermediate values on the Pareto Front (See Problem 2c in Figure 2).

Local Optima Traps (e.g. Problem 2b)
When approximating a continuous Pareto front from smooth preference inputs, the PCPL agent can fail to capture certain regions of the front in the presence of local optima (Problem 2b). As shown in Figure 3, the agent collapses to local solutions and cannot reliably generalize to the global Pareto front, highlighting a key failure mode of existing approaches.

6 Heterogeneous Graph Neural Network

For problems 6a-c with a large number of resources and productions, we used a Heterogeneous Graph Neural Network (HGNN) [Hu *et al.*, 2020] to represent the complex bipartite structure of the demand-resource dependencies. Extracting global information from Graph Neural Networks to use as feature input to RL models is challenging, because global pooling methods can potentially dilute the quantity and quality of information available in the original environment. To tackle this issue, past works have used specific node embeddings [Wang *et al.*, 2025] and node embedding concatenation [Sharma and Kunkel, 2025]; thus, we hope to show that flexible pooling methods like Mean/Max Pooling and Attention Pooling can still capture global information more effectively than MLPs. We replace the common MLP feature extractor with a HGNN-based feature extractor by using multiple Heterogeneous Graph Attention layers and performing global pooling on the node embeddings for different types of nodes separately. Each HGNN layer is comprised of several Graph Attention Networks [Veličković *et al.*, 2018], one for each type of node (Demand / Resource / Unallocated). Two HGNN layers are stacked with residual connections in between.

6.1 HGNN Preference Conditioning

To increase the preference-awareness of the policy, we incorporate preference conditioning at multiple stages within the HGNN feature extractor and training pipeline, enhancing the model's robustness to varying preference inputs.

Concatenating Preferences to Features

We concatenate the preference vector to both the initial node embeddings and the final pooled node embeddings. Additionally, residual connections between HGNN layers are employed to preserve preference information across layers throughout the forward pass.

Preference-conditioned Attention Pooling

We use preference-conditioned multi-head attention pooling, where each head computes attention scores over the nodes by jointly considering the node features and the preference vector $\mathbf{w}$. For each head, the node embeddings are transformed together with $\mathbf{w}$, weighted by the attention scores, and aggregated into a pooled representation. The outputs of all heads are then concatenated to form the final preference-aware global graph representation.

6.2 MLP vs HGNN Model Comparison

For small graphs, MLP-based models are preferred for their stability and rate of convergence. However, Graph Neural Networks have the potential to better generalize to large graphs with large dependency structures and eventually outperform MLPs. We investigated this behavior for Problems 6a-c by increasing the number of demands and resources to 100 each, with a total of 201 nodes (including "Unallocated" resource node). From Table 1, the HGNN-based approaches achieve significantly and consistently better hypervolume scores than MLPs. The two pooling variants (Mean+Max vs. preference-conditioned Attention Pooling) span fixed versus learned aggregation. Attention Pooling attains the highest HV but the lowest OS, indicating that learned aggregation maximizes global coverage at the expense of per-objective preference alignment. Both HGNN methods have fewer parameters than the MLP-based PPO model, as the number of MLP parameters scales quadratically with the size of the allocation matrix, due to the flattening of the allocation matrix and the initial fully connected feedforward neural network. The MLP outperforms the HGNN versions on ordering score, which may indicate that the MLP finds more preference-aligned local solutions than the HGNN-based models, but struggles to find solutions closer to the global solution (See Section 4).

Table 1: Metrics for Problem 6 (100x100 graph) using PCPL-PPO with MLP (1.58M params), HGNN+MeanMaxPool (635K params), and HGNN+AttentionPool (1.15M params) at 3M steps across 5 random seeds. The hypervolume (HV) measures overall policy performance (mean $\pm$ std). The mean scores for PNDS and Ordering Score measure consistency and preference alignment.

Problem 6a	HV $\times 10^6$	PNDS	Order
MLP	3.6 ± 0.6	**0.20**	**0.86**
HGNN+MeanMaxPool	9.3 ± 2.1	0.13	0.72
HGNN+AttentionPool	$\mathbf{14.2 \pm 2.8}$	0.16	0.67
Problem 6b	HV $\times 10^6$	PNDS	Order
MLP	4.2 ± 0.6	0.16	**0.87**
HGNN+MeanMaxPool	7.9 ± 2.4	0.11	0.74
HGNN+AttentionPool	$\mathbf{14.6 \pm 2.4}$	**0.17**	0.70
Problem 6c	HV $\times 10^6$	PNDS	Order
MLP	7.9 ± 1.1	0.13	**0.82**
HGNN+MeanMaxPool	25.2 ± 3.1	0.14	0.63
HGNN+AttentionPool	$\mathbf{33.7 \pm 4.1}$	**0.15**	0.54

7 Conclusion

We introduced **GraphAllocBench**, a flexible benchmark for preference-aware RL and combinatorial resource allocation powered by our custom **CityPlannerEnv** sandbox. GraphAllocBench's diverse problem set exposes distinct failure modes across both SOTA methods and our proposed PCPL-PPO approach, demonstrating that no single method dominates across all settings. Crucially, our supplementary metrics PNDS and Ordering Score reveal preference-alignment gaps that hypervolume alone cannot capture: a solution with lower HV may better reflect user preferences as measured by OS, underscoring the importance of multi-dimensional evaluation for PCPL. Our experiments further show that equipping PPO with an HGNN feature extractor outperforms standard MLP-based baselines on large graphs, though trade-offs in preference consistency remain an open challenge. Overall, GraphAllocBench serves as a versatile and challenging testbed for advancing research in preference-aware many-objective policy learning, and future work will extend the benchmark with richer dependency structures and environmental uncertainty to evaluate risk-aware decision-making.

Acknowledgments

The project or effort depicted was or is sponsored by the U.S. Army Combat Capabilities Development Command – Soldier Center under contract number W912CG-24-D-0001. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. Distribution Statement A. Approved for public release: distribution is unlimited.

References

[An and Zhou, 2025a] Zijian An and Lifeng Zhou. Double oracle algorithm for game-theoretic robot allocation on graphs. *IEEE Transactions on Robotics*, 41:3244–3259, 2025.

[An and Zhou, 2025b] Zijian An and Lifeng Zhou. Reinforcement learning for game-theoretic resource allocation on graphs. *IEEE transactions on automation science and engineering*, 2025.

[Basaklar *et al.*, 2023] Toygun Basaklar, Suat Gumussoy, and Umit Ogras. PD-MORL: Preference-driven multi-objective reinforcement learning algorithm. In *The Eleventh International Conference on Learning Representations*, 2023.

[Becerril Torres *et al.*, 2025] Teresa Becerril Torres, Carlos Ignacio Hernández Castellanos, and Roberto Santana Hermida. Particle swarm algorithm for multi-objective reinforcement learning. In *Proceedings of the 4th Workshop on Multi-Objective Decision Making (MODeM 2025)*, Bologna, Italy, 2025.

[Blank and Deb, 2020] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[Cassimon *et al.*, 2021] Amber Cassimon, Reinout Eyckerman, Siegfried Mercelis, Steven Latré, and Peter Hellinckx. A review of the deep sea treasure problem as a multi-objective reinforcement learning benchmark. *CoRR*, abs/2110.06742, 2021.

[Custódio *et al.*, 2011] A. L. Custódio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente. Direct multisearch for multiobjective optimization. *SIAM Journal on Optimization*, 21(3):1109–1140, 2011.

[Das and Dennis, 1998] Indraneel Das and J. E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

[Deb and Jain, 2014] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

[Deb *et al.*, 2005] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. *Scalable Test Problems for Evolutionary Multiobjective Optimization*, pages 105–145. Springer London, London, 2005.

[Dixit and Tumer, 2025] Gaurav Dixit and Kagan Tumer. Preference-conditioned evolutionary reinforcement learning for multi-objective continuous control. In *Proceedings of the 4th Workshop on Multi-Objective Decision Making (MODeM 2025)*, Bologna, Italy, 2025.

[Désidéri, 2012] Jean-Antoine Désidéri. Multiple-gradient descent algorithm (mgda) for multiobjective optimization. *Comptes Rendus Mathematique*, 350(5):313–318, 2012.

[Felten *et al.*, 2023] Florian Felten, Lucas N. Alegre, Ann Nowé, Ana L. C. Bazzan, El Ghazali Talbi, Grégoire Danoy, and Bruno C. da Silva. A toolkit for reliable benchmarking and research in multi-objective reinforcement learning. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023.

[Fu and Gu, 2024] Xiaoyu Fu and Shenshen Gu. Deep reinforcement learning algorithm based on graph weight multi-pointer network for solving multiobjective traveling salesman problem. *IEEE Access*, 12:179091–179103, 2024.

[Garnelo *et al.*, 2021] Marta Garnelo, Wojciech Marian Czarnecki, Siqi Liu, Dhruva Tirumala, Junhyuk Oh, Gauthier Gidel, Hado van Hasselt, and David Balduzzi. Pick your battles: Interaction graphs as population-level objectives for strategic diversity. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, page 1501–1503, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems.

[Gu *et al.*, 2022] Qinghua Gu, Qingsong Xu, and Xuexian Li. An improved nsga-iii algorithm based on distance dominance relation for many-objective optimization. *Expert Systems with Applications*, 207:117738, 2022.

[Hu *et al.*, 2020] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. *CoRR*, abs/2003.01332, 2020.

[Huband *et al.*, 2006] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

[Ibrahim *et al.*, 2024] Amin Ibrahim, Azam Asilian Bidgoli, Shahryar Rahnamayan, and Kalyanmoy Deb. A novel pareto-optimal ranking method for comparing multi-objective optimization algorithms, 2024.

[Ishibuchi *et al.*, 2016] Hisao Ishibuchi, Ryo Imada, Yu Setoguchi, and Yusuke Nojima. Performance comparison of nsga-ii and nsga-iii on various many-objective test problems. *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 3045–3052, 2016.

[Labiosa *et al.*, 2025] Adam Labiosa, Zhihan Wang, Siddhant Agarwal, William Cong, Geethika Hemkumar, Abhinav Narayan Harish, Benjamin Hong, Josh Kelle, Chen Li, Yuhao Li, Zisen Shao, Peter Stone, and Josiah P. Hanna. Reinforcement learning within the classical robotics stack: A case study in robot soccer, 2025.

[Lin *et al.*, 2022] Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. In *International Conference on Learning Representations*, 2022.

[Lin *et al.*, 2024] Xi Lin, Xiaoyuan Zhang, Zhiyuan Yang, Fei Liu, Zhenkun Wang, and Qingfu Zhang. Smooth tchebycheff scalarization for multi-objective optimization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[Liu *et al.*, 2021] Xingchao Liu, Xin Tong, and Qiang Liu. Profiling pareto front with multi-objective stein variational gradient descent. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 14721–14733. Curran Associates, Inc., 2021.

[Liu *et al.*, 2025] Erlong Liu, Yu-Chang Wu, Xiaobin Huang, Chengrui Gao, Ren-Jian Wang, Ke Xue, and Chao Qian. Pareto set learning for multi-objective reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 18789–18797, 2025.

[Long *et al.*, 2024] Weifan Long, Wen Wen, Peng Zhai, and Lihua Zhang. Role play: Learning adaptive role-specific strategies in multi-agent interactions, 2024.

[Ouyang *et al.*, 2022] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744, 2022.

[Raffin *et al.*, 2021] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[Schrum and Miikkulainen, 2008] Jacob Schrum and Risto Miikkulainen. Constructing complex npc behavior via multi-objective neuroevolution. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2008)*, pages 108–113, Stanford, California, 2008.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[Shao *et al.*, 2019] Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao. A survey of deep reinforcement learning in video games. *CoRR*, abs/1912.10944, 2019.

[Sharma and Kunkel, 2025] Aasish Kumar Sharma and Julian Kunkel. Grapheon rl: A graph neural network and reinforcement learning framework for constraint and data-aware workflow mapping and scheduling in heterogeneous hpc systems. In *2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 489–494, 2025.

[Shishika *et al.*, 2022] Daigo Shishika, Yue Guan, Michael Dorothy, and Vijay Kumar. Dynamic defender-attacker blotto game. In *2022 American Control Conference (ACC)*, pages 4422–4428, 2022.

[Steuer and Choo, 1983] Ralph E. Steuer and Eng-Ung Choo. An interactive weighted tchebycheff procedure for multiple objective programming. *Math. Program.*, 26(3):326–344, October 1983.

[Towers *et al.*, 2024] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

[Vamplew *et al.*, 2011] Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Mach. Learn.*, 84(1–2):51–80, July 2011.

[Van Moffaert *et al.*, 2013] Kristof Van Moffaert, Madalina M. Drugan, and Ann Nowé. Scalarized multi-objective reinforcement learning: Novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 191–199, 2013.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[Wang *et al.*, 2025] Jiaxing Wang, Yifeng Yu, Jiahan Song, Bin Cao, Jing Fan, and Ji Zhang. Rlhgnn: Reinforcement learning-driven heterogeneous graph neural network for next activity prediction in business processes, 2025.

[Xu *et al.*, 2020] Jie Xu, Yunsheng Tian, Pingchuan Ma, Daniela Rus, Shinjiro Sueda, and Wojciech Matusik. Prediction-guided multi-objective reinforcement learning for continuous robot control. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[Yang *et al.*, 2019] Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A generalized algorithm for multi-objective reinforcement learning and policy adaptation. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[Zhu *et al.*, 2023] Baiting Zhu, Meihua Dang, and Aditya Grover. Scaling pareto-efficient decision making via offline multi-objective rl. In *International Conference on Learning Representations*, 2023.

[Zintgraf *et al.*, 2015] Luisa M. Zintgraf, Timon V. Kanters, Diederik M. Roijers, Frans A. Oliehoek, and Philipp Beau. Quality assessment of MORL algorithms: A utility-based approach. In *Benelearn 2015: Proceedings of the 24th Annual Machine Learning Conference of Belgium and the Netherlands*, 2015.

Appendices

A Pareto Front

In multi-objective optimization, a solution is Pareto optimal if it cannot be improved in any objective without degrading at least one other objective. Let's consider a multi-objective maximization problem with N objectives, represented by a vector function:

$$\mathbf{F}(x) = (f_1(x), f_2(x), \ldots, f_N(x)),$$

where x is a solution in the decision space X.

A solution $x_1 \in X$ is said to **dominate** another solution $x_2 \in X$, denoted as $x_1 \succ x_2$, if and only if:

1. $f_i(x_1) \geq f_i(x_2)$ for all objectives $i = 1, \ldots, N$.
2. $f_j(x_1) > f_j(x_2)$ for at least one objective $j \in \{1, \ldots, N\}$.

A solution $x^* \in X$ is called **Pareto optimal** if there is no other solution $x \in X$ that dominates it. The set of the objective values of all Pareto-optimal solutions form the **Pareto Front**.

B Hypervolume Definition

The hypervolume of a set of solutions generated from a particular policy is defined as follows:

$$\mathrm{HV}(X, \mathbf{r}) = \lambda \left(\bigcup_{\mathbf{x} \in X} \left([r_0, x_0] \times \cdots \times [r_{N-1}, x_{N-1}] \right) \right), \tag{1}$$

where:

- $X \subset \mathbb{R}^N$ is a finite set of objective vectors (solutions in the objective space),
- $\mathbf{x} = (x_0, x_1, \ldots, x_{N-1}) \in X$ is an individual objective vector,
- $\mathbf{r} = (r_0, r_1, \ldots, r_{N-1}) \in \mathbb{R}^N$ is the reference point, chosen such that $x_i \geq r_i$ for all i (maximization case),
- N is the number of objectives,
- $\lambda(\cdot)$ denotes the Lebesgue measure (i.e. the volume) of the dominated region.

The reference point is standardized to be at the origin for all problems, since all objective functions are designed to be non-negative.

C Ordering Score Computation

Refer to Algorithm 1 for Ordering Score computation.

Algorithm 1 Ordering Score Computation

Require: Environment env with N objectives, policy π_θ, number of samples n_{samp}, steps n_{step} in each parameter sweep
Ensure: Ordering score $\mathcal{O} \in [0, 1]$

Generate set of preferences $\mathbf{W}$ by sweeping each objective i from 0 to 1 in n_{step} steps, sampling other components uniformly via random Dirichlet samples (repeat n_{samp} times).
Evaluate π_θ $\forall \mathbf{w} \in \mathbf{W}$ to obtain a list of objective vectors.
for each objective i and repetition j **do**
    Extract reward sequence $\mathbf{J}_i^{(j)}$ across n_{step} steps.
    if all values in $\mathbf{J}_i^{(j)}$ are equal **then**
        $s_{i,j} \leftarrow 1$
    else
        $s_{i,j} \leftarrow \frac{1}{2}\left(\text{Spearman}(\mathbf{J}_i^{(j)}, \text{sorted}(\mathbf{J}_i^{(j)})) + 1\right)$
    end if
end for
return $\mathcal{O} = \frac{1}{N \cdot n_{\text{samp}}} \sum_{i=1}^{N} \sum_{j=1}^{n_{\text{samp}}} s_{i,j}$

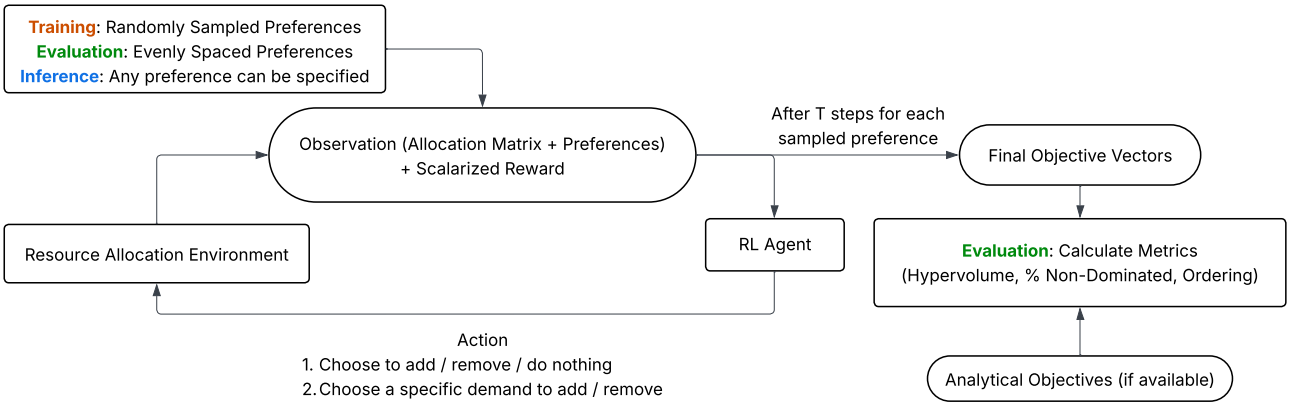


Figure 4: Training and evaluation pipeline using the CityPlannerEnv Gymnasium environment and Proximal Policy Optimization (PPO) agent from Stable Baselines3 [Raffin *et al.*, 2021].

D Training Methodology

D.1 PCPL-PPO Solution

Using the set of objective functions as our multi-objective reward function $\mathbf{J}$, we train a preference-conditioned policy using Proximal Policy Optimization (PPO) to convergence. We then employ the methods in Section 4 to evaluate the trained model. Figure 4 illustrates our training and evaluation pipeline. During training, preferences are sampled from a flat Dirichlet distribution for each rollout, and the PPO agent is trained to maximize the scalarized reward based on different preferences for each rollout, given the preference vector as input. Each rollout lasts for a total of T steps, which is part of the environment configuration set prior to training.

D.2 Normalization

Firstly, we normalized each component of the objective function to between 0 and 1, by taking the worst values of each component (i.e. the nadir point) to be at the origin (since objectives are defined to be always non-negative), and setting the best values (i.e. ideal point) to the current maximum reward obtained based on all prior rollouts. Even though using a moving ideal point may lead to some instability in training when discovering new ideal points, the agent eventually learns the optimal behavior over a large number of training steps.

D.3 Scalarization

Most existing RL-based scalarization methods use weighted sum scalarizations (i.e. Reward $= \mathbf{J} \cdot \mathbf{w}$) because of their simplicity. However, weighted-sum methods are ineffective in approximating non-convex Pareto Fronts. Therefore, we experimented with other scalarization methods, such as Tchebycheff scalarization [Steuer and Choo, 1983] and Penalty Boundary Intersection (PBI) [Deb and Jain, 2014]. We found that Smooth Tchebycheff Scalarization [Lin *et al.*, 2024] performed the most robustly in terms of hypervolume achieved when the agent encountered highly non-convex Pareto fronts. Additionally, the Smooth Tchebycheff Scalarization includes a smoothness hyperparameter that can be tuned for different problems with varying Pareto Front landscapes.

Unlike weighted sums, methods such as the Smooth Tchebycheff Scalarization require an ideal point to compute the scalarized reward, and this information is not readily available in an RL environment. Hence we use the moving ideal point [Van Moffaert *et al.*, 2013] introduced in Section D.2 as an estimate of the true ideal point.

D.4 Model Architecture

We used a Stable Baselines3 Proximal Policy Optimization (PPO) Agent with a Multi-Layer Perceptron (MLP) feature extractor (See Figure 5). This feature extractor can be easily replaced with alternate architectures such as Graph Neural Networks (Section 6), while maintaining the same feature vector dimension.

E Objective Function

See Table 3 for examples of objective functions defined in GraphAllocBench using CityPlannerEnv. Each component of the objective function $\mathbf{J}_i$ can depend on any production $\mathbf{P}_j$, and comprises elementary mathematical functions such as linear, quadratic, logarithmic, logistic, Gaussian, ceiling and floor functions. We can obtain complex objective functions based on the sum of elementary functions to design particular challenges (e.g. Gaussian with ceiling, sinusoidal increasing). All objectives

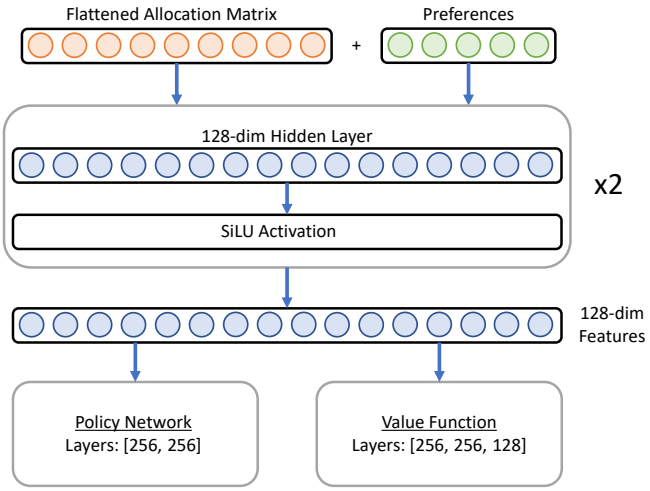


Figure 5: PPO architecture with a shared 2-layer MLP feature extractor (128 units, SiLU) for policy and value networks, implemented with Stable Baselines3.

Table 2: Evaluation Problem Set Description

Problems	$\|P\|$	N	**FC**	**Description**
0	2	2	Yes	Baseline simple logarithmic objective functions with smooth convex Pareto Fronts. The baseline problem uses a Fully-Connected (FC) dependency graph, where every demand depends on every resource.
1a-c	2	2	Yes	Difficult objective functions, including oscillatory behavior, stationary rewards, and spikes.
2a-c	5	2	Yes	More demands, which increases the action and observation spaces.
3a-b	5	5	Yes	5 objectives with sparse rewards, building on difficult objective functions from previous testcases.
4a-b	5	5	No	Varied dependencies and resources, instead of Fully-Connected (FC) dependency graphs.
5a-e	5	3-5	No	Testcases with dependencies, resources, and objectives sampled randomly. Extended horizon for allocating more available resources.
6a-c	100	5	No	Random testcases with simple convex functions similar to baseline, but with more complex graph structures with 100 demands and 100 resources. The ideal Pareto Front is not computed due to computational complexity.

*For more detailed problem definitions, refer to Appendix E.

are non-negative (i.e. $\mathbf{J}_i = \max(f(\mathbf{P}), 0)$) so that reward is always non-negative. For more detailed problem definitions on the full problem set, refer to https://github.com/jzh001/GraphAllocBench.

F Results and Comparison with PD-MORL, Envelope Q-Learning, and PSL

We ran PD-MORL [Basaklar *et al.*, 2023] (MO-DDQN-HER), Envelope Q-Learning [Yang *et al.*, 2019], and Pareto Set Learning (PSL) [Liu *et al.*, 2025] on all GraphAllocBench problems and compare against our PPO-based PCPL (PCPL-PPO) in Table 4. For each baseline we conducted a per-problem random hyperparameter search (e.g. learning rate, discount factor, batch size, soft-update rate, replay buffer size, exploration schedule, and training step budget) and used the best configuration found per problem to train 5 independent seeds at up to 1M steps.

Table 4: Comparison of PD-MORL, Envelope Q-Learning, PSL, and PCPL-PPO across all benchmark problems (mean $\pm$ std, n=5 seeds). The baselines use per-problem hyperparameters selected via random search. Bold indicates the best result per metric per problem.

Problem	**Method**	**Norm. HV** $\uparrow$	**PNDS** $\uparrow$	**Ordering** $\uparrow$
0	PD-MORL	0.973 ± 0.012	0.923 ± 0.094	0.997 ± 0.003
	Envelope	0.889 ± 0.020	0.723 ± 0.222	0.937 ± 0.054
	PSL	0.990 ± 0.009	0.985 ± 0.034	$\mathbf{1.000 \pm 0.000}$

Continued on next page

Table 4: Comparison of PD-MORL, Envelope Q-Learning, PSL, and PCPL-PPO across all benchmark problems (continued).

Problem	Method	Norm. HV ↑	PNDS ↑	Ordering ↑
	PCPL-PPO	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$
1a	PD-MORL	0.608 ± 0.078	0.846 ± 0.094	0.956 ± 0.023
	Envelope	0.385 ± 0.159	0.477 ± 0.333	0.961 ± 0.041
	PSL	$\mathbf{0.943 \pm 0.059}$	$\mathbf{0.877 \pm 0.117}$	$\mathbf{0.995 \pm 0.010}$
	PCPL-PPO	0.696 ± 0.079	0.708 ± 0.100	0.954 ± 0.047
1b	PD-MORL	0.439 ± 0.126	$\mathbf{0.908 \pm 0.138}$	0.945 ± 0.052
	Envelope	0.459 ± 0.246	0.400 ± 0.324	0.665 ± 0.080
	PSL	$\mathbf{0.887 \pm 0.061}$	0.800 ± 0.069	0.876 ± 0.069
	PCPL-PPO	0.696 ± 0.122	0.846 ± 0.077	$\mathbf{0.982 \pm 0.017}$
1c	PD-MORL	0.136 ± 0.118	$\mathbf{0.877 \pm 0.177}$	0.942 ± 0.075
	Envelope	0.065 ± 0.000	0.769 ± 0.094	0.922 ± 0.015
	PSL	0.211 ± 0.239	0.600 ± 0.148	$\mathbf{0.980 \pm 0.025}$
	PCPL-PPO	$\mathbf{0.282 \pm 0.213}$	0.538 ± 0.094	0.955 ± 0.040
2a	PD-MORL	0.290 ± 0.093	0.862 ± 0.126	0.965 ± 0.074
	Envelope	0.235 ± 0.205	0.477 ± 0.333	0.739 ± 0.071
	PSL	0.873 ± 0.047	0.862 ± 0.084	0.999 ± 0.001
	PCPL-PPO	$\mathbf{0.992 \pm 0.002}$	$\mathbf{1.000 \pm 0.000}$	$\mathbf{1.000 \pm 0.000}$
2b	PD-MORL	0.676 ± 0.191	0.692 ± 0.122	0.911 ± 0.056
	Envelope	0.548 ± 0.126	0.523 ± 0.257	0.972 ± 0.010
	PSL	$\mathbf{0.912 \pm 0.038}$	$\mathbf{0.769 \pm 0.188}$	$\mathbf{0.983 \pm 0.018}$
	PCPL-PPO	0.599 ± 0.187	0.615 ± 0.224	0.902 ± 0.070
2c	PD-MORL	0.169 ± 0.070	0.800 ± 0.177	0.977 ± 0.030
	Envelope	0.087 ± 0.088	0.415 ± 0.235	0.698 ± 0.141
	PSL	$\mathbf{0.748 \pm 0.061}$	0.662 ± 0.088	0.950 ± 0.066
	PCPL-PPO	0.614 ± 0.075	$\mathbf{0.877 \pm 0.088}$	$\mathbf{0.999 \pm 0.003}$
3a	PD-MORL	0.495 ± 0.094	$\mathbf{0.395 \pm 0.102}$	0.737 ± 0.083
	Envelope	0.240 ± 0.221	0.216 ± 0.209	0.738 ± 0.071
	PSL	0.578 ± 0.048	0.162 ± 0.059	0.809 ± 0.020
	PCPL-PPO	$\mathbf{0.807 \pm 0.035}$	0.381 ± 0.043	$\mathbf{0.934 \pm 0.023}$
3b	PD-MORL	0.013 ± 0.012	0.627 ± 0.101	$\mathbf{0.838 \pm 0.088}$
	Envelope	0.112 ± 0.060	$\mathbf{0.680 \pm 0.067}$	0.782 ± 0.047
	PSL	0.197 ± 0.156	0.087 ± 0.044	0.607 ± 0.078
	PCPL-PPO	$\mathbf{0.295 \pm 0.083}$	0.257 ± 0.087	0.833 ± 0.075
4a	PD-MORL	0.621 ± 0.049	0.069 ± 0.043	0.804 ± 0.117
	Envelope	0.399 ± 0.292	0.048 ± 0.035	0.672 ± 0.059
	PSL	$\mathbf{0.994 \pm 0.009}$	0.221 ± 0.080	$\mathbf{0.982 \pm 0.016}$
	PCPL-PPO	0.835 ± 0.107	$\mathbf{0.297 \pm 0.107}$	0.862 ± 0.045
4b	PD-MORL	0.141 ± 0.061	0.386 ± 0.179	0.663 ± 0.026
	Envelope	0.143 ± 0.105	0.255 ± 0.052	0.713 ± 0.102
	PSL	$\mathbf{0.863 \pm 0.042}$	0.404 ± 0.098	0.780 ± 0.082
	PCPL-PPO	0.764 ± 0.088	$\mathbf{0.953 \pm 0.023}$	$\mathbf{0.938 \pm 0.037}$
5a	PD-MORL	0.414 ± 0.117	0.442 ± 0.105	0.760 ± 0.075
	Envelope	0.425 ± 0.009	0.415 ± 0.029	0.866 ± 0.012
	PSL	$\mathbf{0.834 \pm 0.033}$	0.589 ± 0.129	0.907 ± 0.050
	PCPL-PPO	0.593 ± 0.052	$\mathbf{0.699 \pm 0.100}$	$\mathbf{0.952 \pm 0.028}$
5b	PD-MORL	0.962 ± 0.005	$\mathbf{0.558 \pm 0.077}$	0.815 ± 0.132
	Envelope	0.404 ± 0.199	0.145 ± 0.130	0.692 ± 0.068
	PSL	$\mathbf{0.976 \pm 0.013}$	0.514 ± 0.100	$\mathbf{0.934 \pm 0.060}$

Continued on next page

Table 4: Comparison of PD-MORL, Envelope Q-Learning, PSL, and PCPL-PPO across all benchmark problems (continued).

Problem	Method	Norm. HV ↑	PNDS ↑	Ordering ↑
	PCPL-PPO	0.966 ± 0.004	0.209 ± 0.075	0.922 ± 0.032
5c	PD-MORL	0.903 ± 0.062	0.516 ± 0.050	0.751 ± 0.132
	Envelope	0.405 ± 0.161	0.132 ± 0.064	0.488 ± 0.117
	PSL	$\mathbf{0.924 \pm 0.099}$	0.266 ± 0.109	0.647 ± 0.056
	PCPL-PPO	0.898 ± 0.039	$\mathbf{0.859 \pm 0.202}$	$\mathbf{0.916 \pm 0.076}$
5d	PD-MORL	0.484 ± 0.196	$\mathbf{0.312 \pm 0.180}$	0.521 ± 0.098
	Envelope	$\mathbf{0.692 \pm 0.047}$	0.188 ± 0.066	0.625 ± 0.047
	PSL	0.658 ± 0.135	0.197 ± 0.060	$\mathbf{0.800 \pm 0.047}$
	PCPL-PPO	0.516 ± 0.166	0.094 ± 0.034	0.686 ± 0.080
5e	PD-MORL	0.464 ± 0.164	0.625 ± 0.098	0.565 ± 0.099
	Envelope	0.520 ± 0.105	$\mathbf{0.703 \pm 0.125}$	0.749 ± 0.044
	PSL	$\mathbf{0.900 \pm 0.070}$	0.699 ± 0.060	$\mathbf{0.864 \pm 0.014}$
	PCPL-PPO	0.723 ± 0.126	0.483 ± 0.171	0.754 ± 0.036

The results illustrate how GraphAllocBench differentiates algorithm behavior across diverse problem settings. PSL achieves the highest normalized hypervolume on 10 of 16 problems and beats PCPL-PPO head-to-head on hypervolume on 11 of 16, making it the strongest method for overall Pareto-front coverage. PCPL-PPO instead leads on PNDS (7 of 16 problems), and the two methods are comparable on ordering score, each leading on 8 problems (tying on Problem 0). Notably, PCPL-PPO remains the most robust method on the hardest sparse and non-convex problems (1c, 3a, 3b), where it still attains the best hypervolume. PD-MORL does not lead hypervolume on any problem but stays competitive on PNDS (5 wins), particularly on the sharp and oscillatory objectives of Problems 1b and 1c. Envelope Q-Learning leads hypervolume only on Problem 5d, and generally underperforms the other methods across the broader problem set, particularly on problems with sparse rewards (Problems 1c, 3a) and high-dimensional observations (5b–5c).

As discussed in Section 5.1, these supplementary metrics expose preference-alignment weaknesses on the five-objective problems (3a, 3b, 4a) and Problem 5c that hypervolume alone would miss; full per-problem values appear in Table 4.

The benchmark's diversity also reveals where all methods struggle most. On the hardest non-convex and sparse problems (1c, 3b), every method attains low hypervolume (best HV 0.282 and 0.295, both from PCPL-PPO), reflecting the difficulty of recovering discontinuous Pareto fronts. The value-based baselines PD-MORL and Envelope Q-Learning see substantially degraded performance on problems with sparse rewards, complex dependency structures, or unbalanced objectives (Problems 2c, 3a, 3b, 4b), whereas PSL and PCPL-PPO remain comparatively strong. These results demonstrate that GraphAllocBench's varied problem set and complementary metrics can expose distinct failure modes in preference-conditioned policy learning.

We note that these comparisons are based on five seeds with substantial variance on the hardest problems (e.g. Problem 1c, 0.282 ± 0.213 for PCPL-PPO), so per-problem head-to-head counts should be read as trends rather than statistically separated rankings.

G Effect of Increasing Number of Objectives

We study the effect of increasing the number of objectives (Figure 6) by modeling each objective as dependent on a unique production (i.e. $N = |\mathbf{P}|$) and using objective functions that monotonically increase in each production. As the true Pareto Front is computationally intensive to calculate for high-dimensional objectives, we use the ratio of the quantity of allocated resources to total resources available to estimate the proximity of the predicted solutions to the true Pareto Front; intuitively, if objective functions are always monotonically increasing, it is always better to use more resources to get more reward, regardless of preference. Hence, the Pareto Front consists of outcomes that use all available resources in the environment. In Figure 6, results suggest that the agent maintains relatively stable performance even as the objective space dimensionality quadruples for simple objective functions, given sufficient model capacity and number of training steps. Additionally, a relatively high 20-objective ordering score (0.89) can still be reached at 1M steps. This restricted setup enables us to conclude that increasing the number of objectives (and productions) with a simple objective function and dependency setup may not necessarily lead to a significant reduction in overall performance, as measured by the proportion of allocated resources as an estimate of the closeness to the ideal solution. This result concurs with the findings of similar tests [Ishibuchi *et al.*, 2016] conducted using evolutionary methods for multi-objective optimization on benchmarks such as DTLZ.

H Ordering Score Sensitivity

To test the sensitivity of the ordering score, we calculated the ordering score for Problems 5d and 5e across 5 random seeds with different sampling patterns by varying values of α for the Dirichlet Distribution sampling in Algorithm 1. For simple problems

Table 3: Objective functions for selected problems (zero lower bound assumed for all functions).

Problem	Objective Function $\mathbf{J}$
0	$\mathbf{J}_0 = 10\log(\mathbf{P}_0 + \epsilon + 1)$ $\mathbf{J}_1 = 10\log(\mathbf{P}_1 + \epsilon + 1)$
1a	$\mathbf{J}_0 = \min(0.6\mathbf{P}_0^2, 18) + \max(5\exp(-0.9(\mathbf{P}_1 - 7)^2), 0.1)$ $\mathbf{J}_1 = -1.5\mathbf{P}_0^2 + 0.8\mathbf{P}_0 + 12 + 5\exp(-0.1(\mathbf{P}_1 - 3)^2) + 0.2\mathbf{P}_1\log(2\mathbf{P}_1 + \epsilon + 1)$
1b	$\mathbf{J}_0 = 0.5\mathbf{P}_0 + 3 + 3\sin(3\mathbf{P}_0) - \log(1.5\mathbf{P}_1 + \epsilon + 2) + 2.5\exp(-1.2(\mathbf{P}_1 - 7)^2)$ $\mathbf{J}_1 = -1.5\log(5.5\mathbf{P}_0 + \epsilon + 1) + 9\exp(-0.5(\mathbf{P}_0 - 2)^2) + 5 + (0.5\mathbf{P}_1 + 1.2)\sin(0.9\mathbf{P}_1 + 0.6)$
1c	$\mathbf{J}_0 = -\mathbf{P}_0 + \frac{20}{1+\exp(\mathbf{P}_1 - 3)}$ $\mathbf{J}_1 = \frac{1}{1+\exp(\mathbf{P}_0 - 6)} + 5 - \frac{15}{1+\exp(0.7(\mathbf{P}_1 - 5))}$

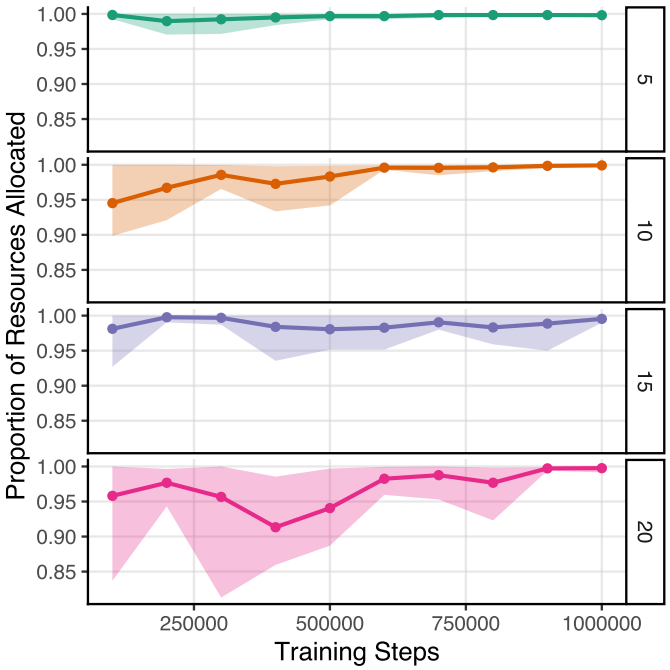
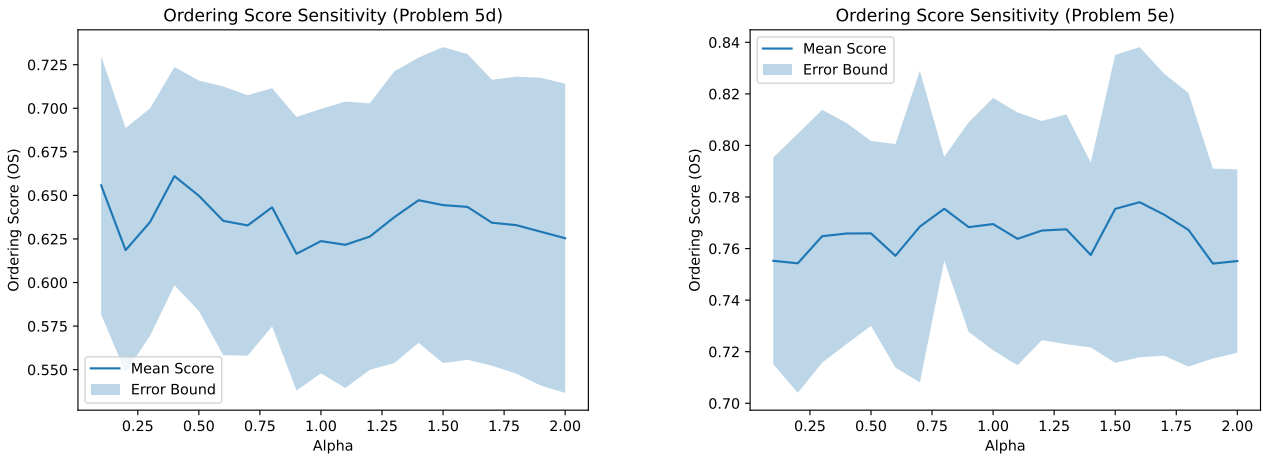


Figure 6: Performance over different number of objectives (5, 10, 15, 20) for a simple increasing objective function, across 5 random seeds. The points show the mean performance, and the shaded regions show the range over 5 seeds.



(a) Problem 5d

(b) Problem 5e

Figure 7: Order Sensitivity (Mean + Standard Deviation) across Dirichlet Distribution α parameter.

(e.g. Problem 0), the ordering score remained relatively constant. For more complex problems, ordering scores may fluctuate with different sampling methods. To investigate the sensitivity of Ordering Score with respect to sampling, Problem 5d and 5e were selected for their higher number of objectives (5), varied dependencies and low ordering score. The variation in Ordering Score is shown in Figures 7a and 7b.

In Figures 7a and 7b, although there are fluctuations in ordering score due to the variations in policies obtained, the mean ordering score remains relatively stable over different alpha parameters for the Dirichlet distribution, indicating resilience over different sampling methods.