# HEURISTIC SETS OVER A SPACE OF DISTRIBUTIONS: LLM-BASED EVOLUTION OF HEURISTICS BEYOND A SINGLE INSTANCE GENERATOR

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Large language models can design heuristics for combinatorial problems by evolving program code, but the resulting heuristics are designed and tested on instances from a single generator, and independent evaluations report that their advantage disappears, sometimes falling behind hand-written rules, once the instances come from elsewhere. Complementary heuristic sets, as in EoH-S, answer this with several heuristics, but they too are evolved on one generator. We study this problem on online bin packing. We first show that it is not only a weakness of the search: we construct two item-size distributions and compute their optimal policies exactly, and at the same packing state the two require different actions, so no single heuristic minimizes the expected number of bins on every distribution. This grounds the set in the problem and moves the question to what it is evolved on. We construct DistSpace-OBP, a family of 20,000 item-size distributions selected from 180,000 random candidates by farthest-point sampling in total variation distance, with measured coverage, and we propose DiSE, which extends EoH-S to evolve its set over this family: each generation is scored on a fresh mini-batch of distributions, on which surviving members are re-scored before comparison. To use the set online, a selector chooses a member from a prefix of the item stream, by retrieval under TV or $W_1$ or by lookahead simulation. On a BPPLIB suite of 6,064 instances, where six published heuristics designed on Weibull instances are all behind Best Fit, the DiSE sets have the lowest gap ($5.92\%$ against $7.11\%$ for Best Fit and $6.32\%$ for EoH-S on its fixed instance set), and on the Weibull benchmark they never saw they are ahead of Best Fit; a lookahead selector realizes part of this online, beating or matching the best fixed choice made in hindsight.

## 1 INTRODUCTION

LLM-based automated heuristic design (AHD) (Liu et al., 2026c) has produced new constructions for open mathematical problems (Romera-Paredes et al., 2024), improved production cluster scheduling at Google (Novikov et al., 2025), and found new best-known solutions for large-scale capacitated vehicle routing (Liu et al., 2024b). These advances involve three dimensions of AHD. *What it rewards* concerns the evaluation objective: which instances candidates are tested on and how their performance is scored. *What it designs* concerns the form and scope of the candidate programs, from single scoring functions (Liu et al., 2024a) to complementary heuristic sets (Liu et al., 2026b; Gungordu et al., 2026), multi-component algorithms (Lv et al., 2026), and entire codebases (Novikov et al., 2025). *How it searches* concerns how candidates are generated, revised, and selected, through reflective evolution (Ye et al., 2024), diversity-driven search (Dat et al., 2025), Monte Carlo tree search (Zheng et al., 2025), and meta-optimization (Shi et al., 2026).

Recent evaluations, however, reveal that evolved heuristics can overfit their design distributions. In online bin packing (OBP), independent benchmarking finds that the evaluated Weibull-trained FunSearch and EoH heuristics underperform Best Fit overall across heterogeneous datasets (Sim et al., 2025); RAISE likewise finds that EoH, ReEvo, and MoH can lose to Best Fit under distribution shifts (Liu et al., 2026a). Efforts to improve generalization span the same three dimensions. RAISE and ASRO adapt the instances used to evaluate candidates, exposing weaknesses that a fixed training set may miss (Liu et al., 2026a; Ke et al., 2026). EoH-S and DASH develop complementary sets



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

or specialist archives to serve different instances (Liu et al., 2026b; Wang et al., 2026). MOSAIC combines these two directions by co-evolving specialist heuristics and adversarial instances (Gungordu et al., 2026). MoH uses multi-task meta-optimization to learn heuristic-generation procedures that transfer across tasks and scales (Shi et al., 2026). Together, these efforts broaden the goal from strong performance on the design distribution to generalization across distributions.

Earlier work (Rice, 1976; Huberman et al., 1997; Gomes & Selman, 2001; Xu et al., 2008) and EoH-S et al. (Liu et al., 2026b; Gungordu et al., 2026) show that complementary heuristic sets can outperform individual heuristics in practice. However, this does not tell us whether a sufficiently effective design process could produce a single heuristic that replaces the set. The no-free-lunch theorem (Wolpert & Macready, 1997), invoked by Liu et al. (2026b); Gungordu et al. (2026), does not settle this question for a specific problem: as Kerschke et al. (2019) caution, performance complementarity is often incorrectly attributed to this theorem. We therefore ask whether a single distribution-unaware heuristic can be optimal across item-size distributions. Taking OBP as a concrete case, we construct two item-size distributions in a finite-horizon setting with i.i.d. arrivals and show that they require conflicting optimal decisions at the same reachable packing state. Consequently, no single policy in the considered class, without access to the generating distribution, minimizes expected bin count under both distributions. This gives a problem-specific motivation for complementary heuristic sets; their online performance must additionally be evaluated with a selector.

Learning complementary heuristics requires deciding which distributions should guide their evolution. Varying generator parameters (Liu et al., 2026b) and adaptively seeking hard or discriminative instances (Liu et al., 2026a; Ke et al., 2026; Gungordu et al., 2026) broaden training experience, but instance difficulty and feature-space coverage do not directly characterize how well the training distributions cover a specified space of input distributions. We therefore study a complementary approach: constructing a fixed training environment independently of solver performance, with coverage measured over a specified distribution pool. We introduce DistSpace-OBP, a reusable family of 20,000 OBP item-size distributions. Under i.i.d. arrivals, each distribution specifies the probability of each integer item size. We construct a diverse candidate pool using Gaussian Dirichlet process mixtures (Escobar & West, 1995) and mixtures of uniform intervals, then select representative distributions by farthest-point sampling under total variation (TV) distance. The resulting covering radius bounds the distance from every candidate to its nearest selected distribution. This gives an explicit, measurable scope for training and evaluating complementary heuristics, while keeping distribution selection independent of solver performance.

Evaluating every candidate on all training distributions is expensive. We introduce DiSE (Distribution-Space Set Evolution), which builds on EoH-S (Liu et al., 2026b) and evaluates candidates on a different mini-batch of distributions each generation. Existing members are also re-evaluated on that batch, so all candidates are compared on the same data. Each epoch visits every training distribution once. We test the resulting sets on the Weibull benchmark used in FunSearch and EoH (Romera-Paredes et al., 2024; Liu et al., 2024a), and the heterogeneous BPPLIB-based suite used to reveal overfitting in (Sim et al., 2025). Best-of-set evaluation chooses the best member in hindsight, while online packing must commit to placements before future items are known. We therefore select a member from an initial item prefix, using distribution retrieval or lookahead simulation, and evaluate the resulting online gains separately.

Our contributions are:

- **Theory.** We construct two item-size distributions in a finite-horizon i.i.d. OBP setting and prove that no single policy in the considered class, without access to the generating distribution, minimizes expected bin count under both. This gives a problem-specific theoretical motivation for complementary heuristic sets.
- **Training environment.** We introduce DistSpace-OBP, a reusable family of 20,000 OBP item-size distributions selected independently of solver performance from 172,796 filtered candidates, with a measured TV covering radius of $0.167$ over this pool.
- **Method.** We propose DiSE, extending EoH-S (Liu et al., 2026b) to evolve 10-member sets over up to 20,000 distributions in batches of 128. Re-scoring existing members on each batch keeps candidate comparisons on the same data.

- **Experiments.** We demonstrate generalization to external benchmarks: without benchmark-specific training, DiSE outperforms Best Fit on both benchmarks under best-of-set evaluation, whereas all six evaluated Weibull-designed heuristics fall behind Best Fit on BPPLIB. Its dataset-averaged BPPLIB gap is $5.92\%$, vs $6.32\%$ for EoH-S and $7.11\%$ for Best Fit. On Weibull($M = 200$), online lookahead selection beats the best fixed action chosen in hindsight, improving on Best Fit by $1.291$ versus $0.955$ percentage points.

## 2 Related Work

**LLM-based heuristic design and its distributional brittleness.** Automated heuristic design predates LLMs: hyper-heuristics and genetic programming have long selected or generated heuristics, including packing and dispatching rules (Burke et al., 2013; Drake et al., 2020; Branke et al., 2016; Nguyen et al., 2017; Đurasević & Jakobović, 2018; Zhang et al., 2024). Since FunSearch (Romera-Paredes et al., 2024), LLM-guided evolution has widened both what is searched, from single functions to whole algorithms and to online and dynamic problems (Liu et al., 2024a; Novikov et al., 2025; Huang et al., 2026; Zhou et al., 2026; Cao et al., 2026), and how it is searched, through reflection, diversity preservation, tree and neighborhood search, multi-objective evolution, and meta-optimization (Ye et al., 2024; Dat et al., 2025; Hu & Zhang, 2025; Zheng et al., 2025; Xie et al., 2025; Yao et al., 2025; Shi et al., 2026). In every case the evaluator decides which behaviors are rewarded, and independent evaluations show that what it rewards does not carry over: across twelve heterogeneous OBP datasets the Weibull-trained FunSearch heuristic incurs nearly twenty times the excess bins of Best Fit (Sim et al., 2025), capacity shifts alone separate robust from brittle evolved heuristics (Yao et al., 2025), and similar failures are reported for online optimization tasks and for GP-evolved dispatching rules (Liu et al., 2026a; Zhu et al., 2026). Section 3.2 shows that for OBP this is not only a weakness of the search.

**Complementary heuristic sets.** Algorithm selection and portfolios exploit complementary strengths across instances (Rice, 1976; Huberman et al., 1997; Gomes & Selman, 2001), from SATzilla, Hydra, ISAC, and AutoFolio (Xu et al., 2008; 2010; Kadioglu et al., 2010; Lindauer et al., 2015) to selectors that read code or raw item sequences (Wu et al., 2024; Alissa et al., 2023) and to selection that adapts during deployment (Gagliolo & Schmidhuber, 2010; Degroote et al., 2016; Tornede et al., 2022; van Rijn et al., 2014; Biedenkapp et al., 2020; Adriaensen et al., 2022; Fitzgerald et al., 2015; Pierreval & Mebarki, 1997; Priore et al., 2014; Đurasević & Jakobović, 2022; Sim et al., 2015; Hart & Sim, 2016). LLM-based methods now generate the members themselves: EoH-S optimizes per-instance best-of-set performance on a fixed instance set (Liu et al., 2026b); InstSpecHH, HeurAgenix, DASH, and quality-diversity libraries produce specialists and select among them while solving (Zhang et al., 2025; Yang et al., 2025; Wang et al., 2026; Chen & Liu, 2026); SLEEP and MOSAIC evolve portfolios for changing environments and against discriminative instances (Nguyen et al., 2026; Gungordu et al., 2026). Empirical complementarity among candidates does not show that no single optimal policy exists, and this does not follow from no-free-lunch results either (Kerschke et al., 2019); Section 3.2 establishes it for OBP directly.

**Training-distribution construction.** Training environments in AHD range from fixed instance sets, as in EoH-S and MoH (Liu et al., 2026b; Shi et al., 2026), to construction guided by solver performance: RAISE searches for hard instances near the nominal training set, ASRO co-evolves solvers and generators, SLEEP learns environments from evaluation feedback, and MOSAIC combines adversarial generation with coverage of an instance-feature grid (Liu et al., 2026a; Ke et al., 2026; Nguyen et al., 2026; Gungordu et al., 2026); AHD benchmarks broaden the tasks but evaluate on fixed collections (Sun et al., 2026; Chen et al., 2026; van Stein et al., 2025). DistSpace-OBP takes the complementary route: a fixed family constructed independently of any solver, with coverage measured over a specified pool of distributions (Section 3.3). Appendix A extends this discussion.

## 3 Method

We study how to evolve complementary heuristics across input distributions and use the resulting set on a new online instance. The approach connects three components: DistSpace-OBP defines the training environment, DiSE evolves a set through distribution batches, and a prefix-based selector

chooses a member during execution. We first define the design objective (Section 3.1) and establish a problem-specific motivation for complementarity (Section 3.2). We then describe the distribution family (Section 3.3), then set evolution and online selection (Section 3.4).

### 3.1 Problem setting and design objective

**Online bin packing problem formulation.** Online bin packing asks how to pack a stream of items into as few fixed-capacity bins as possible when each item must be placed as soon as it arrives. Formally, an instance is a sequence $L = (a_1, \ldots, a_n)$ of item sizes with $0 < a_i \leq C$, and every bin has capacity $C$. Item $a_i$ is assigned to a bin whose remaining capacity is at least $a_i$, before $a_{i+1}$ is seen, and is never moved afterwards. The objective is to minimize the number of bins used. We write $A(L)$ for the number of bins used by an online algorithm $A$ on $L$, and $\mathrm{OPT}(L)$ for the offline minimum. In the offline problem the whole sequence is known before packing, so the difficulty is computational. Online, every decision is made from the items seen so far and the current packing state, while its cost depends on items that have not yet arrived. Which decision is good therefore depends on what the rest of the sequence will look like.

**Performance measure.** Computing $\mathrm{OPT}(L)$ is impractical during search, where every candidate heuristic is run on many sequences. Following EoH and EoH-S (Liu et al., 2024a; 2026b), we score a heuristic by its gap to a lower bound on the optimal number of bins,

$$\mathrm{gap}(A, L) = \frac{A(L) - \mathrm{lb}(L)}{\mathrm{lb}(L)}, \qquad \mathrm{lb}(L) = \left\lfloor \frac{1}{C} \sum_{i=1}^{n} a_i + \tfrac{1}{2} \right\rfloor \leq \mathrm{OPT}(L), \tag{1}$$

where $\mathrm{lb}$ is the rounded total-size bound used in the EoH-S implementation (Appendix B.1). Lower is better, and experiments report the gap in percent. Since $\mathrm{lb}(L) \leq \mathrm{OPT}(L)$, this value overstates the gap to OPT.

**Heuristics and heuristic sets.** A heuristic is a rule that, at each arrival, chooses a feasible bin from the current item and the state of the bins. It is fixed before packing begins and is not told the generating distribution. The heuristics evolved in this paper use the interface of FunSearch and EoH (Romera-Paredes et al., 2024; Liu et al., 2024a): a program receives the item size and the remaining capacities of the bins and returns a priority for every bin, and the item is placed in the feasible bin with the highest priority (Appendix B.2).

Following EoH-S (Liu et al., 2026b), a heuristic set $S$ of $k$ heuristics is scored on an instance by its best member, $\min_{A \in S} \mathrm{gap}(A, L)$. We consider i.i.d. arrivals from an item-size distribution $D$, and score a set over a finite family $\mathcal{D}$ of such distributions by taking the expectation on each distribution and averaging over the family, with every distribution weighted equally:

$$\mathrm{gap}_{\mathcal{D}}(S) = \frac{1}{|\mathcal{D}|} \sum_{D \in \mathcal{D}} \mathbb{E}_{L \sim D^n} \left[ \min_{A \in S} \mathrm{gap}(A, L) \right], \qquad |S| = k. \tag{2}$$

The minimum is *inside* the expectation: different members may win on different instances, including instances from the same distribution. For $k = 1$ this is the objective for a single heuristic, and a set never scores worse than any of its members taken alone, $\mathrm{gap}_{\mathcal{D}}(S) \leq \min_{A \in S} \mathrm{gap}_{\mathcal{D}}(\{A\})$; a strict inequality is the benefit of complementarity over the best fixed member (Appendix B.3).

### 3.2 Why a heuristic set rather than a single heuristic

Empirical complementarity alone cannot rule out a better single heuristic. If one heuristic minimized the expected number of bins under every item-size distribution, there would be no reason to design a set. A heuristic in the sense of Section 3.1 is fixed before packing begins and is not told which distribution generates the items (Appendix B.2). We show that two distributions already rule this out, and that a set of two heuristics then beats every single one.

**Theorem 1** (Incompatible distributional optima)**.** *For horizon $n = 6$ there exist two item-size distributions $D_1, D_2$ on a common finite support such that no heuristic is optimal under both: writing $\mathcal{A}_D$ for the set of heuristics that minimize $\mathbb{E}_{L \sim D^6}[A(L)]$,*

$$\mathcal{A}_{D_1} \cap \mathcal{A}_{D_2} = \emptyset. \tag{3}$$

Table 1: Exact expected final bin counts for the two decisions in the witness prefix $(0.4, 0.4, 0.29)$, with optimal continuation under the indicated distribution. Smaller values are better. Identical bins are grouped into one action.

| Current decision | Action | Under $D_1$ | Under $D_2$ |
|---|---|---|---|
| Second item: $0.4$ | Use the existing bin | $3.2048$ | $3.0038$ |
| | Open a new bin | $\mathbf{3.0256}$ | $\mathbf{2.5913}$ |
| Third item: $0.29$ | Use either $0.6$ residual | $3.037$ | $\mathbf{2.489}$ |
| | Open a new bin | $\mathbf{3}$ | $3$ |

Table 2: Candidate generators and their contributions to DistSpace-OBP. Candidate counts precede filtering; selected counts sum to 20,000.

| Generator | Typical profiles | Candidates | Selected |
|---|---|---|---|
| DPM, $\alpha = 0.15$ | Smooth, mostly unimodal | 100,000 | 2,922 |
| DPM, $\alpha = 3$ | Smooth, multiple modes | 40,000 | 6,393 |
| Uniform-interval mixtures | Sharp boundaries | 40,000 | 10,685 |

**Corollary 1.** *Let $A_1 \in \mathcal{A}_{D_1}$ and $A_2 \in \mathcal{A}_{D_2}$ for the distributions of Theorem 1. For every heuristic $A$,*

$$\sum_{D \in \{D_1, D_2\}} \mathbb{E}_{L \sim D^6}\big[\min\{A_1(L), A_2(L)\}\big] < \sum_{D \in \{D_1, D_2\}} \mathbb{E}_{L \sim D^6}\big[A(L)\big]. \tag{4}$$

The left side has the form of the set objective equation 2 on the family $\{D_1, D_2\}$, with the number of bins in place of the gap: on each sequence the set $\{A_1, A_2\}$ is scored by its better member, and so scored it beats every single heuristic.

*Proof of Theorem 1.* Take capacity $C = 1$ and item sizes $(0.29, 0.4, 0.6)$. Under $D_1 = (0.3, 0.3, 0.4)$ large items are common; under $D_2 = (0.8, 0.1, 0.1)$ small items dominate. For each distribution, Table 1 reports the exact expected number of bins that follows from each action when the remaining items are packed optimally; the values are computed by backward induction over the six arrivals (Appendix B.4). The same appendix shows that the optimal way of packing under a fixed distribution can be written as a heuristic, so $\mathcal{A}_D$ is nonempty.

An optimal heuristic must take the better action at every state it reaches with positive probability. Let the first two items both be $0.4$. Under both distributions the second $0.4$ is better placed in a new bin (Table 1, first two rows), so any heuristic in $\mathcal{A}_{D_1}$ or $\mathcal{A}_{D_2}$ now has two bins with residual capacity $0.6$ each. Let the third item be $0.29$. Here the two distributions disagree: under $D_1$ it is better to open a new bin and keep both $0.6$ residuals for the $0.6$ items that are likely to come, whereas under $D_2$ it is better to put it into one of the two bins, since the items that follow are mostly small (last two rows). Each action is strictly better than the other under its own distribution.

A heuristic sees only the item $0.29$ and the two residuals $0.6$. It does not know which distribution produced them, so it makes the same choice under both. Opening a new bin is not optimal under $D_2$, and using an existing bin is not optimal under $D_1$. Since this prefix has positive probability under both distributions, no heuristic belongs to both $\mathcal{A}_{D_1}$ and $\mathcal{A}_{D_2}$, which is equation 3. □

*Proof of Corollary 1.* Under $D_1$, $\min\{A_1(L), A_2(L)\} \le A_1(L)$ for every $L$, and $A_1$ attains the minimum expected bin count; likewise under $D_2$ with $A_2$. The left side of equation 4 is therefore at most the sum of the two minima. By Theorem 1, $A$ lies outside $\mathcal{A}_{D_1}$ or $\mathcal{A}_{D_2}$, so it exceeds the minimum under that distribution and cannot fall below it under the other. □

### 3.3 DistSpace-OBP: a training family with measured coverage

Section 3.2 shows that which heuristics are optimal depends on the item-size distribution, so the family $\mathcal{D}$ in the set objective equation 2 decides what a set is optimized for. We construct DistSpace-

OBP to make this choice explicit and reusable: a large candidate pool supplies item-size profiles of different shapes, and a distance criterion selects representatives whose coverage of the pool is measured.

**Item-size distributions and the candidate pool.** An item-size distribution is the rule that draws the size of each arriving item. LLM-based design for OBP has used a single one, Weibull$(3, 45)$ (Romera-Paredes et al., 2024; Liu et al., 2024a; 2026b), so heuristics are evolved and tested at one point of the space of distributions; DistSpace-OBP is a family of such points. At reference capacity $C = 100$, a distribution is a probability vector $p = (p_1, \ldots, p_{100})$ on the integer sizes, with continuous densities truncated, discretized, and normalized on this grid. The candidate pool $\mathcal{P}$ is a large random sample of such vectors covering different shapes (Table 2): smooth profiles of varying location, spread, and number of modes from a Dirichlet process mixture (DPM) of Gaussians (Escobar & West, 1995), mostly unimodal at concentration $\alpha = 0.15$ and multimodal at $\alpha = 3$, and sharp-edged profiles from mixtures of 1–3 uniform intervals (parameters in Appendix B.5). Of 180,000 candidates, those with $\max_x p_x > 0.5$ or item-size standard deviation below 1 are excluded, leaving 172,796.

**Selecting the family.** From $\mathcal{P}$ we select 20,000 distributions such that every candidate is close to a selected one. Closeness is total variation distance,

$$\mathrm{TV}(p, q) = \frac{1}{2} \sum_{x=1}^{100} |p_x - q_x|, \tag{5}$$

the probability mass on which two distributions differ: at TV distance $\varepsilon$, sequences drawn from the two agree in all but an $\varepsilon$ fraction of items under the best coupling. Selection is farthest-point sampling. Starting with the candidate closest to the uniform distribution, at each step we add the unselected candidate $p$ whose distance to the selected set, that is, its TV distance to the nearest selected distribution $q$, is largest:

$$p_{m+1} \in \arg \max_{p \in \mathcal{P} \setminus \mathcal{D}_m} \min_{q \in \mathcal{D}_m} \mathrm{TV}(p, q), \qquad \mathcal{D}_{m+1} = \mathcal{D}_m \cup \{p_{m+1}\}. \tag{6}$$

The resulting ordered family $\mathcal{D} = \mathcal{D}_{20,000}$ is DistSpace-OBP. Its covering radius

$$r(\mathcal{D}; \mathcal{P}) = \max_{p \in \mathcal{P}} \min_{q \in \mathcal{D}} \mathrm{TV}(p, q) \simeq 0.167 \tag{7}$$

is the largest distance from any candidate to the family: no distribution in $\mathcal{P}$ is farther than $0.167$ from some selected one. Farthest-point sampling also spreads the family out, since every selected distribution was, when added, the farthest candidate from those before it; any two selected distributions are therefore at least $0.167$ apart, and the median distance from a selected distribution to its nearest other is $0.191$ (against $0.017$ for 20,000 independent DPM draws). Because selection only adds, the first $m$ selected distributions form a farthest-point family of size $m$, so a smaller family is a prefix of the same list, which we write $\mathcal{D}_m$. How far the family lies from the external benchmarks is measured in Section 4.3.

**From distributions to training instances.** The family specifies input distributions; the evaluator runs on item sequences. A run uses a subset $\mathcal{D}_{\mathrm{tr}} \subseteq \mathcal{D}$ and draws one sequence $L_D$ from each distribution $D \in \mathcal{D}_{\mathrm{tr}}$, so that variety comes from the number of distributions rather than from repeated draws of one. The default instantiation, *C100-n1000*, uses capacity $C = 100$ and length $n = 1{,}000$; a *mixed* variant cycles through $C \in \{100, 1000\}$ and $n \in \{200, 1000, 5000\}$ along the distribution order, with sizes rescaled to the capacity as described in Appendix B.5. Once drawn, the sequences are fixed and shared by all candidate heuristics, as the training instances of EoH-S are, so each evaluation costs one packing run per sequence.

### 3.4 DiSE: evolving the set over distribution batches, and using it online

DiSE keeps the machinery of EoH-S (Liu et al., 2026b), its thought-and-code representation, search operators, and greedy complementary selection of $k$ members, and changes only what candidates are scored on. A run minimizes the set gap equation 2 over $\mathcal{D}_{\mathrm{tr}}$ with the one fixed sequence per distribution of Section 3.3. Instead of scoring every candidate on all of $\mathcal{D}_{\mathrm{tr}}$, generation $g$ scores

on a batch $\mathcal{B}_g$ of at most $B$ distributions, cut from a fresh seeded permutation of $\mathcal{D}_{\mathrm{tr}}$ in every epoch, so that each distribution is visited once per epoch. Gap vectors from different batches are not comparable, so survivors, and any offspring scored on an earlier batch, are re-scored on the active batch before they enter a comparison. A generation thus costs $(Q + k)|\mathcal{B}_g|$ packing runs for $Q$ offspring instead of $(Q + k)|\mathcal{D}_{\mathrm{tr}}|$; Appendix C gives the protocol.

DiSE returns a set, and the set gap scores it by its best member in hindsight. To use the set online, a selector packs the first $M$ items with Best Fit and then, from the prefix and the current packing state, with capacity and length known, chooses one member, or stays with Best Fit, for the remaining items, with no further switch. We use two selectors: retrieval, which adopts the action that an offline table ranks best for the reference distribution nearest to the prefix, and lookahead, which simulates every action from the current state on bootstrap suffixes of the prefix and picks the lowest mean bin count. Appendix D defines both; Section 4.4 measures the gain of selection separately from the set's hindsight performance.

# 4 Experiments

We evaluate whether heuristic sets evolved over DistSpace-OBP generalize to external benchmarks. We first introduce the two benchmarks (Section 4.1) and examine how close their item-size distributions are to DistSpace-OBP (Section 4.2; Figure 1). We then compare DiSE with Best Fit and state-of-the-art methods on both benchmarks (Section 4.3). Finally, we assess the gains from prefix-based selection during online packing (Section 4.4).

## 4.1 Benchmarks

We evaluate on two benchmarks; neither took part in constructing DistSpace-OBP or in any design run. The first is the Weibull$(3, 45)$ test data of EoH (Liu et al., 2024a), drawn from the distribution on which FunSearch, EoH and EoH-S design their heuristics: sizes are $45 \cdot \mathrm{Weibull}(3)$, rounded and clipped to $[1, 100]$, so at $C = 100$, half the items lie between $0.3C$ and $0.5C$ and few exceed $0.65C$. It has 1,000, 5,000 and 10,000 items, five instances each, at the design capacity $C = 100$ and at $C = 500$.

The second, which we call the BPPLIB suite, is the benchmark of Sim et al. (2025), assembled to test whether LLM-evolved heuristics overfit their generator: 12 datasets with 6,064 instances, eleven from BPPLIB (Delorme et al., 2018) and one with Weibull-drawn sizes, capacities from 50 to $1.5 \times 10^6$, and items in the shuffled order the authors published. Their size profiles differ from Weibull and from one another: Schwerin items all lie between $0.15C$ and $0.2C$; Scholl 2 and Wäscher consist mostly of small items, with median sizes near $C/6$ and $C/10$; Irnich items lie between $2C/15$ and $2C/3$; and AI, ANI, Hard28, Scholl 1 and the randomly generated instances spread items over most of the capacity.

## 4.2 Distance from the benchmarks to DistSpace-OBP

To relate the family to the distributions on which OBP heuristics are actually tested, Figure 1 projects DistSpace-OBP and its nested prefixes $\mathcal{D}_m$ together with the nine benchmark targets onto the first two principal components of their CDFs. The first benchmark is represented by Weibull$(3, 45)$. For the second, we group the eleven BPPLIB datasets by generator family, pooling Scholl and Schwerin into one target each to obtain eight empirical item-size distributions. The Weibull-drawn Angelopoulos dataset shares the Weibull target. Sizes are normalized by capacity and mapped to the grid of Section 3.3. Distances are TV equation 5 in the original probability space; the projection serves only for display. Appendix B.5 compares each target with its nearest distribution and gives the distances for individual benchmark instances.

The targets lie inside the family: each has a distribution in DistSpace-OBP within TV $0.103$ to $0.213$, and six of the nine are within the covering radius $0.167$ of Section 3.3, that is, as close to the family as any candidate of the pool it was selected from. The prefixes approach the targets as $m$ grows: the nearest distribution to a target lies within TV $0.235$ to $0.485$ in $\mathcal{D}_{128}$, $0.235$ to $0.404$ in $\mathcal{D}_{1,024}$ and $0.197$ to $0.317$ in $\mathcal{D}_{4,096}$ (Table 6). The family also extends far beyond the targets on both sides of

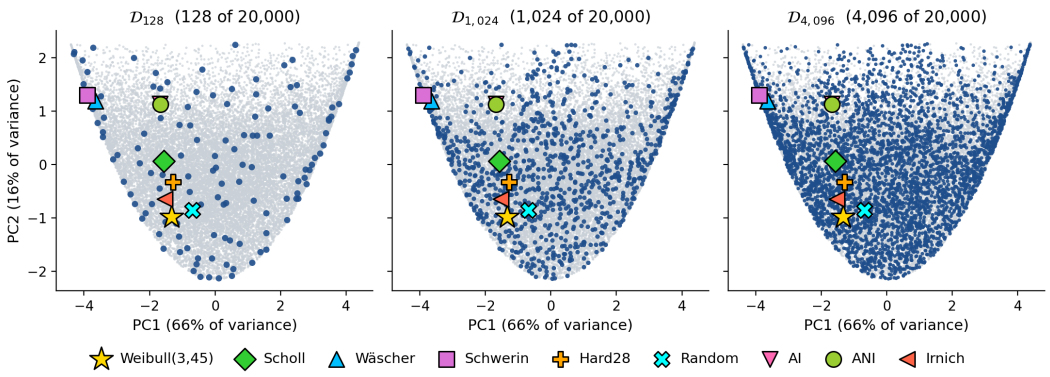


Figure 1: DistSpace-OBP (grey, 20,000 distributions), its nested prefixes $\mathcal{D}_{128}$, $\mathcal{D}_{1,024}$ and $\mathcal{D}_{4,096}$ (blue), on which the DiSE sets of Tables 3 and 4 are evolved, and the nine benchmark targets (markers) on the first two principal components of their CDF vectors ($83\%$ of the variance).

Table 3: Gap (%) on the Weibull$(3, 45)$ test instances of Liu et al. (2024a), 5 per column (items/capacity); lower is better. Sets: per-instance best member; EoH-S: mean over three seeds; DiSE: mean over two. Bold: best per column, within the published methods and within the DiSE family separately.

| Method | 1k/100 | 5k/100 | 10k/100 | 1k/500 | 5k/500 | 10k/500 | avg |
|---|---|---|---|---|---|---|---|
| Best Fit | 4.51 | 4.21 | 4.02 | 1.24 | 0.50 | 0.52 | 2.50 |
| First Fit | 5.01 | 4.55 | 4.30 | 1.24 | 0.60 | 0.52 | 2.70 |
| FunSearch (Romera-Paredes et al., 2024) | 3.86 | 0.72 | 0.41 | 67.00 | 21.76 | 9.21 | 17.16 |
| EoH (Liu et al., 2024a) | 2.67 | 0.70 | 0.56 | 1.24 | 0.50 | 0.37 | 1.01 |
| ReEvo (Ye et al., 2024) | **1.93** | **0.61** | 0.44 | **0.75** | 0.25 | 0.25 | **0.70** |
| HSEvo (Dat et al., 2025) | 2.92 | 0.65 | **0.32** | 67.07 | 67.98 | 67.42 | 34.39 |
| MCTS-AHD (Zheng et al., 2025) | 2.18 | 0.67 | 0.50 | 0.99 | 0.35 | 0.15 | 0.81 |
| MEoH (Yao et al., 2025) | 2.53 | 0.68 | 0.50 | 1.49 | 0.25 | 0.15 | 0.93 |
| EoH-S (Liu et al., 2026b) | 2.08 | 0.80 | 0.68 | **0.75** | **0.22** | **0.10** | 0.77 |
| DiSE-128 | 3.12 | 2.48 | 2.31 | **0.99** | 0.53 | 0.42 | 1.64 |
| DiSE-1,024 | 2.65 | 1.48 | 1.40 | 1.24 | 0.50 | 0.42 | 1.29 |
| DiSE-4,096 | 2.83 | 1.62 | 1.51 | 1.24 | 0.50 | 0.48 | 1.36 |
| DiSE | **2.35** | **0.81** | **0.62** | 1.11 | **0.47** | **0.41** | **0.97** |

the projection, and more than half of its distributions are farther than TV $0.4$ from every benchmark instance (Appendix B.5): it covers the benchmarks without being built around them.

### 4.3 Comparison with the state of the art

To compare DiSE with the state of the art, we evaluate three groups of methods on the two benchmarks. Hand-written rules: Best Fit and First Fit. Single heuristics of LLM-based design, the one published with each method, all designed on Weibull$(3, 45)$ instances: FunSearch (Romera-Paredes et al., 2024), EoH (Liu et al., 2024a), ReEvo (Ye et al., 2024), HSEvo (Dat et al., 2025), MCTS-AHD (Zheng et al., 2025) and MEoH (Yao et al., 2025). Heuristic sets: EoH-S (Liu et al., 2026b), run three times on its original 128 training instances for 2,000 samples, and DiSE, run on the full family and three nested prefixes $\mathcal{D}_m$ of DistSpace-OBP (Figure 1), $m \in \{128,\ 1{,}024,\ 4{,}096\}$, with batch size $B = 128$ and $k = 10$ and reporting the mean over the two runs; we write DiSE for the set evolved on the full family and DiSE-$m$ for the set evolved on the prefix $\mathcal{D}_m$. EoH-S and DiSE share prompts, operators and the language model, DeepSeek-flash. Tables 3 and 4 report the gap equation 1 in percent on the Weibull benchmark and on the BPPLIB suite.

Three conclusions follow. On the Weibull benchmark (Table 3) the heuristics designed on Weibull instances are strongest, as expected: ReEvo reaches $0.70\%$ and EoH-S $0.77\%$, and EoH, MCTS-AHD and MEoH lie between $0.81$ and $1.01\%$, against $2.50\%$ for Best Fit; FunSearch and HSEvo, ahead of Best Fit at the design capacity $C = 100$, fall behind it at $C = 500$ (9–68%). The DiSE sets, evolved over DistSpace-OBP, achieve gaps of $0.97$–$1.64\%$ on the Weibull benchmark.

Table 4: Gap (%) on the 12 datasets (6,064 instances) of Sim et al. (2025); avg: mean over the 12 datasets. Methods, citations, scoring and marks as in Table 3.

| Method | AI | ANI | Irnich | Hard28 | Scholl1 | Scholl2 | SchollH | Schw1 | Schw2 | Wäscher | Random | Angelop. | avg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Best Fit | **2.55** | **2.65** | 3.69 | 5.97 | 10.31 | 6.89 | 11.64 | 13.65 | 11.86 | **6.26** | 7.31 | 2.53 | 7.11 |
| First Fit | 3.70 | 3.77 | 4.03 | 7.32 | 11.40 | 6.95 | 11.64 | 13.65 | 11.86 | 6.52 | 8.47 | 2.87 | 7.68 |
| FunSearch | 34.63 | 34.88 | 52.11 | 48.29 | 26.52 | 142.31 | 58.57 | 470.38 | 466.67 | 111.42 | 25.84 | 18.61 | 124.19 |
| EoH | 13.38 | 13.59 | 39.48 | 30.88 | 18.86 | 12.73 | 20.00 | 14.41 | 13.24 | 16.61 | 18.29 | 11.02 | 18.54 |
| ReEvo | 28.75 | 29.10 | 69.01 | 41.97 | 20.51 | 21.87 | 52.73 | 24.65 | 23.95 | 38.53 | 20.93 | 23.25 | 32.94 |
| HSEvo | 31.56 | 31.35 | 49.82 | 40.64 | 24.32 | 57.93 | 60.36 | 56.17 | 54.38 | 69.10 | 24.21 | 25.06 | 43.74 |
| MCTS-AHD | 25.68 | 25.86 | 51.69 | 38.58 | 20.34 | 19.20 | 34.91 | 20.44 | 19.90 | 30.72 | 20.30 | 20.66 | 27.36 |
| MEoH | 22.10 | 22.10 | 51.88 | 36.62 | 21.53 | 37.58 | 105.26 | 62.82 | 62.10 | 30.63 | 19.96 | 15.33 | 40.66 |
| EoH-S | 2.64 | 2.76 | **2.37** | **5.80** | **9.44** | **6.12** | **8.55** | **12.92** | **11.44** | 6.44 | **6.35** | **0.94** | **6.32** |
| DiSE-128 | 2.44 | 2.56 | 2.68 | 5.58 | 9.35 | 6.42 | 7.92 | **11.23** | **10.12** | 6.39 | 6.29 | 1.77 | 6.06 |
| DiSE-1,024 | 2.47 | 2.57 | **2.26** | **5.46** | 9.36 | **6.14** | 8.19 | 11.54 | 10.31 | **6.26** | 6.18 | 1.33 | 6.01 |
| DiSE-4,096 | **2.42** | **2.54** | 2.33 | 5.51 | **9.23** | 6.15 | 7.55 | 12.12 | 11.00 | **6.26** | **6.15** | 1.53 | 6.07 |
| DiSE | 2.49 | 2.60 | 2.44 | 5.66 | 9.32 | 6.16 | **6.83** | 11.52 | 10.50 | **6.26** | 6.23 | **1.06** | **5.92** |

Table 5: Improvement over Best Fit, mean over the 30 Weibull instances in percentage points, with the DiSE set (seed 1 of Table 3); bold marks the best prefix selector per column.

| Strategy | Information used | $M = 50$ | $M = 100$ | $M = 200$ |
|---|---|---|---|---|
| Best fixed action | full benchmark, in hindsight | $+1.065$ | $+1.004$ | $+0.955$ |
| Retrieval, TV nearest neighbour | prefix | $+0.906$ | $+1.055$ | $+1.081$ |
| Retrieval, $W_1$ nearest neighbour | prefix | $+0.964$ | $\mathbf{+1.077}$ | $+1.081$ |
| Lookahead simulation | prefix | $\mathbf{+1.035}$ | $+1.028$ | $\mathbf{+1.291}$ |
| Oracle | entire instance | $+1.441$ | $+1.463$ | $+1.439$ |

On the BPPLIB suite (Table 4) the order changes: all six single heuristics fall behind Best Fit (18.54–124.19% against 7.11% over datasets), the EoH-S set is ahead of it (6.32% over datasets), and the DiSE sets are best, 5.92–6.07%. Across the four family sizes, the full family gives the lowest average on the Weibull benchmark and the lowest dataset average on the suite; the three prefixes are close to each other on the suite and farther apart on the Weibull data. Since DiSE and EoH-S share every part of the search and differ only in what candidates are scored on, the gain on unseen data comes from the diversity of the training distributions.

### 4.4 Prefix-based selection

We evaluate the selectors on the 30 Weibull test instances at switch points $M \in \{50, 100, 200\}$, with the DiSE members and Best Fit as actions and DistSpace-OBP as library. Table 5 reports the mean improvement over Best Fit in percentage points. All selectors improve on Best Fit at every switch point, by 0.9 to 1.3, and at $M = 50$ and $M = 100$ they match the best fixed action chosen in hindsight. At $M = 200$ lookahead gains 1.291 points, against 1.081 for either retrieval and 0.955 for the best fixed action, and comes within 0.15 points of the per-instance oracle (1.439), against 0.41 and 0.44 points at $M = 50$ and $M = 100$: the more of the stream the selector has seen, the less it leaves to hindsight. On the BPPLIB suite, whose instances differ in capacity, length and size profile, the same selectors stay farther from the oracle: at $M = 100$ lookahead recovers 0.236 of the oracle's 0.626 points (Table 8), so most of the headroom there lies in selection.

## 5 Conclusion

LLM-based heuristic design has largely relied on instances from one generator, varied or hardened but not replaced. For online bin packing we showed that the loss away from that generator is not only a matter of search: two item-size distributions can require different actions at the same packing state, so a set is needed, and what it is evolved on is the question. DistSpace-OBP makes that space explicit and measurable, DiSE evolves over it at one batch per generation, and the sets lead on the heterogeneous suite and stay ahead of other heuristics evolved on the Weibull on BPPLIB; a prefix selector turns part of that into online gains. None of this is specific to bin packing: any online problem with a generator of instances, dynamic scheduling among them, admits the same construction. Selection holds the remaining headroom; non-stationary streams are the next test.

### AI USE STATEMENT

In this work, we used generative AI tools to aid and polish writing, and to support research ideation and execution: surveying related work, discussing the problem formulation, and debugging experiment. This is separate from the LLM that DiSE and the baselines use as a heuristic generator by design. The authors edited all AI-assisted text, read every cited reference in the original, reviewed and tested AI-assisted code, and checked all reported numbers against the experiment logs. We have reviewed all AI-assisted work and take responsibility for the final content of this work, including text, claims and artifacts produced with the aid of generative AI.

## REFERENCES

Steven Adriaensen, André Biedenkapp, Gresa Shala, Noor Awad, Theresa Eimer, Marius Lindauer, and Frank Hutter. Automated dynamic algorithm configuration. *Journal of Artificial Intelligence Research*, 75:1633–1699, 2022. doi: 10.1613/jair.1.13922.

Mohamad Alissa, Kevin Sim, and Emma Hart. Automated algorithm selection: from feature-based to feature-free approaches. *Journal of Heuristics*, 29(1):1–38, 2023. doi: 10.1007/s10732-022-09505-4. preprint arXiv:2203.13392.

André Biedenkapp, H. Furkan Bozkurt, Theresa Eimer, Frank Hutter, and Marius Lindauer. Dynamic algorithm configuration: Foundation of a new meta-algorithmic framework. In *ECAI 2020*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 427–434. IOS Press, 2020. doi: 10.3233/FAIA200122. page numbers unverified.

Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Marius Lindauer, Yuri Malitsky, Alexandre Fréchette, Holger H. Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. ASlib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016. doi: 10.1016/j.artint.2016.04.003.

Jürgen Branke, Su Nguyen, Christoph W. Pickardt, and Mengjie Zhang. Automated design of production scheduling heuristics: A review. *IEEE Transactions on Evolutionary Computation*, 20(1): 110–124, 2016. doi: 10.1109/TEVC.2015.2429314. URL https://doi.org/10.1109/TEVC.2015.2429314.

Edmund K. Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013. doi: 10.1057/jors.2013.71.

Shijie Cao, Yuan Yuan, and Jing Liu. Harmonizing real-time constraints and long-horizon reasoning: An asynchronous agentic framework for dynamic scheduling, 2026. URL https://arxiv.org/abs/2605.29262. v1, 28 May 2026; venue unverified.

Hongzheng Chen, Yingheng Wang, Yaohui Cai, Hins Hu, Jiajie Li, Shirley Huang, Chenhui Deng, Rongjian Liang, Shufeng Kong, Haoxing Ren, Samitha Samaranayake, Carla P. Gomes, and Zhiru Zhang. HeuriGym: An agentic benchmark for LLM-crafted heuristics in combinatorial optimization. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Shikun Chen and Yangguang Liu. Illuminating heuristic design: Language-model-driven quality-diversity evolution for combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 2026. doi: 10.1109/TEVC.2026.3721299. Early Access.

Pham Vu Tuan Dat, Long Doan, and Huynh Thi Thanh Binh. HSEvo: Elevating automatic heuristic design with diversity-driven harmony search and genetic algorithm using LLMs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 26931–26938, 2025. doi: 10.1609/aaai.v39i25.34898.

Hans Degroote, Bernd Bischl, Lars Kotthoff, and Patrick De Causmaecker. Reinforcement learning for automatic online algorithm selection — an empirical study. In *ITAT 2016: Information Technologies — Applications and Theory*, volume 1649 of *CEUR Workshop Proceedings*, pp. 93–101, 2016. URL http://ceur-ws.org/Vol-1649.

Maxence Delorme, Manuel Iori, and Silvano Martello. BPPLIB: A library for bin packing and cutting stock problems. *Optimization Letters*, 12(2):235–250, 2018. doi: 10.1007/s11590-017-1192-z.

John H. Drake, Ahmed Kheiri, Ender Özcan, and Edmund K. Burke. Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, 285(2):405–428, 2020. doi: 10.1016/j.ejor.2019.07.073.

Michael D. Escobar and Mike West. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association*, 90(430):577–588, 1995. doi: 10.1080/01621459.1995.10476550.

Tadhg Fitzgerald, Yuri Malitsky, and Barry O'Sullivan. ReACTR: Realtime algorithm configuration through tournament rankings. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pp. 304–310, 2015. URL https://www.ijcai.org/Proceedings/15/Papers/049.pdf.

Matteo Gagliolo and Jürgen Schmidhuber. Algorithm selection as a bandit problem with unbounded losses. In *Learning and Intelligent Optimization (LION 2010)*, volume 6073 of *Lecture Notes in Computer Science*, pp. 82–96. Springer, 2010. doi: 10.1007/978-3-642-13800-3_7. preprint arXiv:0807.1494 / IDSIA-07-08.

Carla P. Gomes and Bart Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1-2):43–62, 2001. doi: 10.1016/S0004-3702(00)00081-3.

Oguzhan Gungordu, Siheng Xiong, and Faramarz Fekri. MOSAIC: Adversarial co-evolution of specialist heuristics and problem instances for LLM-based automated heuristic design, 2026. URL https://arxiv.org/abs/2608.07544.

Emma Hart and Kevin Sim. A hyper-heuristic ensemble method for static job-shop scheduling. *Evolutionary Computation*, 24(4):609–635, 2016. doi: 10.1162/EVCO_a_00183.

Qinglong Hu and Qingfu Zhang. Partition to evolve: Niching-enhanced evolution with LLMs for automated algorithm discovery. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Jin Huang, Qihao Liu, Xinyu Li, Liang Gao, and Yue Teng. Automatic programming via large language models with population self-evolution for dynamic fuzzy job shop scheduling problem. *IEEE Transactions on Fuzzy Systems*, 34(3):896–908, 2026. doi: 10.1109/TFUZZ.2025.3650586. URL https://arxiv.org/abs/2410.22657. arXiv:2410.22657; v1 (Oct 2024) titled "...for dynamic job shop scheduling problem" with author order Huang, Li, Gao, Liu, Teng.

Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg. An economics approach to hard computational problems. *Science*, 275(5296):51–54, 1997. doi: 10.1126/science.275.5296.51.

Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. ISAC — instance-specific algorithm configuration. In *ECAI 2010*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pp. 751–756. IOS Press, 2010. doi: 10.3233/978-1-60750-606-5-751.

Xinyi Ke, Kai Li, Junliang Xing, Yifan Zhang, and Jian Cheng. Game-theoretic co-evolution for LLM-based heuristic discovery. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research. PMLR, 2026. URL https://openreview.net/forum?id=z9x30tqLUV.

Pascal Kerschke, Holger H. Hoos, Frank Neumann, and Heike Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary Computation*, 27(1):3–45, 2019. doi: 10.1162/evco_a_00242.

Marius Lindauer, Holger H. Hoos, Frank Hutter, and Torsten Schaub. AutoFolio: An automatically configured algorithm selector. *Journal of Artificial Intelligence Research*, 53:745–778, 2015. doi: 10.1613/jair.4726.

Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32201–32223, 2024a.

Fei Liu, Rui Zhang, Zhuoliang Xie, Rui Sun, Kai Li, Qinglong Hu, Ping Guo, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. LLM4AD: A platform for algorithm design with large language model, 2024b. URL https://arxiv.org/abs/2412.17287.

Fei Liu, Alessio Figalli, Patrick Owen, and Nicola Serra. RAISE: LLM-based automated heuristic design with robust adversary instance search, 2026a. URL https://arxiv.org/abs/2606.31801. v1, 30 Jun 2026; venue unverified; tasks OBP, Online JSP, Online VRP under five distribution families.

Fei Liu, Yilu Liu, Qingfu Zhang, Xialiang Tong, and Mingxuan Yuan. EoH-S: Evolution of heuristic set using LLMs for automated heuristic design. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 37090–37098, 2026b. doi: 10.1609/aaai.v40i43.41038.

Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Xi Lin, Zhe Zhao, Xialiang Tong, Kun Mao, Zhichao Lu, Zhenkun Wang, Mingxuan Yuan, and Qingfu Zhang. A systematic survey on large language models for algorithm design. *ACM Computing Surveys*, 58(8):1–32, 2026c. doi: 10.1145/3787585.

Haoze Lv, Ning Lu, Shengcai Liu, Shaofeng Zhang, and Ke Tang. MuEvo: LLM-driven evolution of multi-heuristic ensemble, 2026. URL https://arxiv.org/abs/2608.03636.

Su Nguyen, Yi Mei, and Mengjie Zhang. Genetic programming for production scheduling: a survey with a unified framework. *Complex & Intelligent Systems*, 3(1):41–66, 2017. doi: 10.1007/s40747-017-0036-x. URL https://doi.org/10.1007/s40747-017-0036-x.

Viet Tuan Kiet Nguyen, Huu Duc Nguyen, Cong Dao Tran, Trong Duc Bui, and Thi Thanh Binh Huynh. SLEEP: Simulated future learning environments for automated evolution of heuristic portfolios. ICML 2026 Workshop, 2026. URL https://openreview.net/forum?id=sVYB5IihO2.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Henri Pierreval and Nasser Mebarki. Dynamic scheduling selection of dispatching rules for manufacturing system. *International Journal of Production Research*, 35(6):1575–1591, 1997. doi: 10.1080/002075497195137.

Paolo Priore, Alberto Gómez, Raúl Pino, and Rafael Rosillo. Dynamic scheduling of manufacturing systems using machine learning: An updated review. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 28(1):83–97, 2014. doi: 10.1017/S0890060413000516.

John R. Rice. The algorithm selection problem. In *Advances in Computers*, volume 15, pp. 65–118. Elsevier, 1976. doi: 10.1016/S0065-2458(08)60520-3.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024.

Yiding Shi, Jianan Zhou, Wen Song, Jieyi Bi, Yaoxin Wu, Zhiguang Cao, and Jie Zhang. Generalizable heuristic generation through LLMs with meta-optimization. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Kevin Sim, Emma Hart, and Ben Paechter. A lifelong learning hyper-heuristic method for bin packing. *Evolutionary Computation*, 23(1):37–67, 2015. doi: 10.1162/EVCO_a_00121.

Kevin Sim, Quentin Renau, and Emma Hart. Beyond the hype: Benchmarking LLM-evolved heuristics for bin packing. In *Applications of Evolutionary Computation (EvoApplications 2025)*, volume 15613 of *Lecture Notes in Computer Science*, pp. 386–402. Springer, 2025. doi: 10.1007/978-3-031-90065-5_24.

Kate Smith-Miles, Davaatseren Baatar, Brendan Wreford, and Rhyd Lewis. Towards objective measures of algorithm performance across instance space. *Computers & Operations Research*, 45: 12–24, 2014. doi: 10.1016/j.cor.2013.11.015.

Weiwei Sun, Shengyu Feng, Shanda Li, and Yiming Yang. CO-Bench: Benchmarking language model agents in algorithm search for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33126–33134, 2026. doi: 10.1609/aaai.v40i39.40596.

Alexander Tornede, Viktor Bengs, and Eyke Hüllermeier. Machine learning for online algorithm selection under censored feedback. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2022)*, volume 36, pp. 10370–10380, 2022. doi: 10.1609/aaai.v36i9.21279.

Marko Đurasević and Domagoj Jakobović. A survey of dispatching rules for the dynamic unrelated machines environment. *Expert Systems with Applications*, 113:555–569, 2018. doi: 10.1016/j.eswa.2018.06.053. URL https://doi.org/10.1016/j.eswa.2018.06.053.

Marko Đurasević and Domagoj Jakobović. Selection of dispatching rules evolved by genetic programming in dynamic unrelated machines scheduling based on problem characteristics. *Journal of Computational Science*, 61:101649, 2022. doi: 10.1016/j.jocs.2022.101649. URL https://doi.org/10.1016/j.jocs.2022.101649.

Jan N. van Rijn, Geoffrey Holmes, Bernhard Pfahringer, and Joaquin Vanschoren. Algorithm selection on data streams. In *Discovery Science (DS 2014)*, volume 8777 of *Lecture Notes in Computer Science*, pp. 325–336. Springer, 2014. doi: 10.1007/978-3-319-11812-3_28.

Niki van Stein, Anna V. Kononova, Haoran Yin, and Thomas Bäck. BLADE: Benchmark suite for LLM-driven automated design and evolution of iterative optimisation heuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO)*, pp. 2336–2344, 2025. doi: 10.1145/3712255.3734347.

Rongzheng Wang, Yihong Huang, Muquan Li, Jiakai Li, Di Liang, Bob Simons, Pei Ke, Shuang Liang, and Ke Qin. Rethinking LLM-driven heuristic design: Generating efficient and specialized solvers via dynamics-aware optimization, 2026. URL https://arxiv.org/abs/2601.20868.

David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. doi: 10.1109/4235.585893.

Xingyu Wu, Yan Zhong, Jibin Wu, Bingbing Jiang, and Kay Chen Tan. Large language model-enhanced algorithm selection: Towards comprehensive algorithm representation. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 5235–5244, 2024. doi: 10.24963/ijcai.2024/579.

Zhuoliang Xie, Fei Liu, Zhenkun Wang, and Qingfu Zhang. LLM-driven neighborhood search for efficient heuristic design. In *2025 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8. IEEE, 2025. doi: 10.1109/CEC65147.2025.11043025.

Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606, 2008. doi: 10.1613/jair.2490.

Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. Hydra: Automatically configuring algorithms for portfolio-based selection. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI 2010)*, volume 24, pp. 210–216, 2010. doi: 10.1609/aaai.v24i1.7565.

Xianliang Yang, Ling Zhang, Haolong Qian, Lei Song, and Jiang Bian. Heuragenix: Leveraging llms for solving complex combinatorial optimization challenges. *arXiv preprint arXiv:2506.15196*, 2025.

Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. Multi-objective evolution of heuristic using large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 27144–27152, 2025. doi: 10.1609/aaai.v39i25.34922.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. ReEvo: Large language models as hyper-heuristics with reflective evolution. In *Advances in Neural Information Processing Systems*, volume 37, pp. 43571–43608, 2024.

Fangfang Zhang, Yi Mei, Su Nguyen, and Mengjie Zhang. Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling. *IEEE Transactions on Evolutionary Computation*, 28(1):147–167, 2024. doi: 10.1109/TEVC.2023.3255246. URL https://doi.org/10.1109/TEVC.2023.3255246.

Shaofeng Zhang, Shengcai Liu, Ning Lu, Jiahao Wu, Ji Liu, Yew-Soon Ong, and Ke Tang. Llm-driven instance-specific heuristic generation and selection. *arXiv preprint arXiv:2506.00490*, 2025.

Zhi Zheng, Zhuoliang Xie, Zhenkun Wang, and Bryan Hooi. Monte carlo tree search for comprehensive exploration in LLM-based automatic heuristic design. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*. PMLR, 2025.

XinLei Zhou, Jin Huang, Jie Yang, Xinyu Li, and Liang Gao. DSevolve: Enabling real-time adaptive scheduling on dynamic flexible job shop with LLM-evolved heuristic portfolios, 2026. URL https://arxiv.org/abs/2603.27628. v2, 28 Jul 2026; venue unverified.

Luyao Zhu, Fangfang Zhang, Yi Mei, and Mengjie Zhang. Investigation of the generalisation ability of genetic programming-evolved scheduling rules in dynamic flexible job shop scheduling, 2026. URL https://arxiv.org/abs/2601.15717. v1, 22 Jan 2026; venue unverified.

# A EXTENDED RELATED WORK

**LLM-based automated heuristic design.** Automated heuristic design predates LLMs: hyper-heuristics and genetic programming have long selected or generated heuristics, including packing and dispatching rules (Burke et al., 2013; Drake et al., 2020; Branke et al., 2016; Nguyen et al., 2017; Đurasević & Jakobović, 2018; Zhang et al., 2024). Since FunSearch (Romera-Paredes et al., 2024) demonstrated that LLM-guided evolution can discover heuristics that outperform human-designed baselines, automated heuristic design has expanded both the scope of the programs it searches and the mechanisms used to search them. The search space now extends from individual heuristic functions to larger algorithmic implementations (Liu et al., 2024a; Novikov et al., 2025), and applications now include online and dynamic problems (Huang et al., 2026; Zhou et al., 2026; Cao et al., 2026). Search strategies have likewise diversified, incorporating reflection (Ye et al., 2024), diversity preservation and niching (Dat et al., 2025; Hu & Zhang, 2025), tree and neighborhood search (Zheng et al., 2025; Xie et al., 2025), multi-objective evolution (Yao et al., 2025), and meta-optimization (Shi et al., 2026). Across these approaches, the evaluator supplies the performance signal that guides discovery: its choice of training instances and scoring rule specifies which behaviors are rewarded. This makes the evaluation distribution a central consideration when asking whether improvements achieved during search extend to a broader range of instances.

**Distributional brittleness of evolved heuristics.** Independent evaluations show that strong performance within a design distribution can coexist with severe failures beyond it. Across twelve heterogeneous OBP datasets, the Weibull-trained FunSearch heuristic evaluated by Sim et al. (2025) incurs nearly twenty times the average excess bins of Best Fit. Capacity-shift experiments further reveal that generalization can fail even within the same parametric benchmark family, with substantial differences in robustness across evolved heuristics (Yao et al., 2025). The contrast shows that the same shift can expose weaknesses in some evolved heuristics while leaving others effective. Related distribution-shift failures are documented across online optimization tasks (Liu et al., 2026a); in genetic-programming-based scheduling, generalization is associated with similarity between training and test decision-point distributions (Zhu et al., 2026). These findings distinguish success on new instances of a familiar distribution from robustness across distributions, and motivate examining where individual heuristics succeed, where they fail, and whether their strengths are complementary.

**Complementary heuristic sets.** Classical algorithm selection and portfolios exploit performance differences across instances by selecting or combining solvers with complementary strengths (Rice, 1976; Huberman et al., 1997; Gomes & Selman, 2001), as realized in SATzilla, Hydra, ISAC, and AutoFolio (Xu et al., 2008; 2010; Kadioglu et al., 2010; Lindauer et al., 2015) and benchmarked by ASlib (Bischl et al., 2016); instance space analysis maps where each algorithm performs well (Smith-Miles et al., 2014). Learned selectors now use code semantics (Wu et al., 2024) or raw item sequences, including for bin packing (Alissa et al., 2023). Selection can also adapt during deployment, through bandit-based online selection (Gagliolo & Schmidhuber, 2010; Degroote et al., 2016; Tornede et al., 2022), selection on data streams (van Rijn et al., 2014), dynamic algorithm configuration (Biedenkapp et al., 2020; Adriaensen et al., 2022; Fitzgerald et al., 2015), real-time selection among dispatching rules (Pierreval & Mebarki, 1997; Priore et al., 2014; Đurasević & Jakobović, 2022), and lifelong hyper-heuristics that grow their repertoire over time (Sim et al., 2015; Hart & Sim, 2016). LLM-based methods extend this approach to generating the heuristics themselves. EoH-S directly optimizes per-instance best-of-set performance and establishes monotonicity and supermodularity of its objective (Liu et al., 2026b). InstSpecHH generates specialists for instance subclasses, HeurAgenix selects among LLM-evolved heuristics during solving, and DASH maintains group-specific solver archives for profile-based retrieval (Zhang et al., 2025; Yang et al., 2025; Wang et al., 2026). Quality-diversity approaches also maintain structured heuristic libraries with instance-dependent selection (Chen & Liu, 2026). SLEEP extends portfolio evolution to changing environments, and MOSAIC co-evolves discriminative instances and regional specialists before extracting a complementary portfolio (Nguyen et al., 2026; Gungordu et al., 2026). However, complementarity among existing candidates does not establish that a universally optimal policy cannot exist within a fixed problem class. Nor does this follow directly from generic no-free-lunch results: as Kerschke et al. (2019) emphasize, empirical performance complementarity is often incorrectly attributed to these theorems. Our analysis addresses this distinction within OBP. For finite-horizon

OBP with i.i.d. item sizes, we consider policies that are not given the generating distribution and show that no single policy minimizes the expected bin count under every item-size distribution. This provides a problem-specific theoretical motivation for studying complementary heuristic sets. The same analytical approach can be applied to other online optimization problems to examine whether a single policy can be optimal across different input distributions.

**Training-distribution construction and coverage.** Training environments in AHD range from predefined instances or task distributions, as in EoH-S and MoH (Liu et al., 2026b; Shi et al., 2026), to adaptive construction guided by solver performance. RAISE searches for hard instances within a prescribed neighborhood of the nominal training set (Liu et al., 2026a); ASRO co-evolves solvers and instance generators (Ke et al., 2026); and SLEEP learns training environments from subsequent evaluation feedback (Nguyen et al., 2026). MOSAIC further combines adversarial instance generation with explicit coverage of an instance-feature grid (Gungordu et al., 2026). These developments highlight two related aspects of evaluator design: dynamically exposing weaknesses of current solvers and statically characterizing the range of inputs represented during training. Recent AHD benchmarks broaden the range of problems and tasks but evaluate on fixed instance collections (Sun et al., 2026; Chen et al., 2026; van Stein et al., 2025). Alongside adaptive instance generation, fixed, reusable datasets with a clearly defined distributional scope remain valuable for establishing common training conditions and evaluating generalization under controlled distribution shifts.

Taken together, these studies motivate a joint view of heuristic search, complementarity, and evaluation distributions. A central question is how the distributions used to score and select candidates shape the generalization of individual heuristics and the complementarity of heuristic sets. Explicitly characterizing training coverage and testing on distributions withheld from search and selection provide a basis for investigating this relationship.

# B Problem setting details

## B.1 Lower bound used in the gap

The bound in equation 1 is the total item size divided by the capacity and rounded to the nearest integer. It is the bound computed in the EoH-S evaluation code, and we keep it so that our gaps are comparable with values reported for EoH-S. Rounding to the nearest integer never exceeds the ceiling, so $\mathrm{lb}(L) \le \lceil \sum_i a_i/C \rceil \le \mathrm{OPT}(L)$, and the gap of equation 1 is an upper bound on the gap to $\mathrm{OPT}(L)$. The bound ignores how item sizes combine, so on instances with many large items it can lie well below $\mathrm{OPT}(L)$; gaps of different heuristics on the same instance are therefore comparable with each other, but not with the true distance to optimality. All instances used in this paper have $\mathrm{lb}(L) > 0$.

## B.2 Packing loop and decision-function class

Algorithm 1 is the packing loop behind every heuristic in this paper. Bins are allocated in advance: $n$ items can never need more than $n$ bins, so $n$ bins of capacity $C$ are opened at the start, and a bin that still has its full capacity at the end was never used. This lets one rule express both placing an item in a partially filled bin and opening a new one, without a separate open-bin decision.

**Algorithm 1** Online packing with decision function $f$

**Require:** Item sequence $L = (a_1, \ldots, a_n)$; capacity $C$; decision function $f$
1: $\mathbf{b} \leftarrow (C, \ldots, C) \in \mathbb{R}^n$
2: **for** $i = 1$ **to** $n$ **do**
3: $\quad j \leftarrow f(a_i, \mathbf{b})$ ▷ choose a bin with $b_j \ge a_i$
4: $\quad b_j \leftarrow b_j - a_i$
5: **end for**
6: **return** $|\{j : b_j < C\}|$ ▷ number of used bins

Formally, a heuristic is a deterministic decision function $f(a, \mathbf{b})$ that returns a feasible bin index from the current item size $a$ and the bin state $\mathbf{b}$, here the vector of remaining capacities. The

function is fixed before packing begins and is not told the generating distribution. We write $A_f$ for the packing algorithm induced by $f$ and identify a heuristic with $A_f$ when discussing performance, so that $A_f(L)$ is the number of bins returned by Algorithm 1. Theorem 1 quantifies over this class.

The evolved programs realize $f$ through the FunSearch/EoH interface (Romera-Paredes et al., 2024; Liu et al., 2024a): a Python function `priority(item, bins)` receives the item size and the remaining capacities of the feasible bins (those with $b_j \geq a$, in index order) and returns one score per bin; the item goes to the feasible bin with the highest score, ties broken by the lowest index. Any decision function in the class above can be written in this form by scoring the chosen bin higher than the others, so the priority interface and the decision-function class describe the same set of rules.

### B.3 Remarks on the set objective

For any set $S$ and any instance $L$, $\min_{A \in S} \mathrm{gap}(A, L) \leq \mathrm{gap}(A', L)$ for every $A' \in S$. Taking expectations and averaging over the family gives

$$\mathrm{gap}_{\mathcal{D}}(S) \leq \min_{A \in S} \mathrm{gap}_{\mathcal{D}}(\{A\}), \tag{8}$$

which is the inequality quoted in Section 3.1. The difference between the two sides is the amount by which the set, scored by its per-instance best member, improves on its best single member; it is zero when one member is best on every instance.

### B.4 Proof details for Theorem 1

**Value recursion.** Write $\mathbf{b}$ for the vector of remaining capacities and $t$ for the number of items still to arrive. For a distribution $D$ with probabilities $p_D(a)$ on the support, the minimum expected final bin count satisfies

$$V_D(0, \mathbf{b}) = |\{j : b_j < C\}|, \qquad V_D(t, \mathbf{b}) = \sum_a p_D(a) \min_{j:\, b_j \geq a} V_D\big(t-1, \mathbf{b} - a\mathbf{e}_j\big), \tag{9}$$

where $\mathbf{e}_j$ is the $j$-th unit vector. Bins with the same remaining capacity lead to the same continuation and are treated as one action. For the construction of Theorem 1 the recursion visits 527 states $(t, \mathbf{b})$ from the empty packing, whose optimal values are 3.220233 under $D_1$ and 2.48615 under $D_2$. Every minimizing action at every state is retained, so the disjointness at the witness state is between the full sets of optimal actions, not between particular tie-breaking choices.

**The optimal continuation is a heuristic.** A minimizer of equation 9 may depend on $t$, whereas a heuristic chooses from the item and the bin state alone. For the support $(0.29, 0.4, 0.6)$ at capacity $C = 1$, every attainable fill of a bin determines how many items it contains, so the residual vector $\mathbf{b}$ determines the number of items placed so far and hence $t$. A $t$-dependent optimal policy is therefore a function of the item and $\mathbf{b}$ alone, that is, a heuristic in the class of Appendix B.2. Hence $\mathcal{A}_D$ in Theorem 1 is nonempty and consists of heuristics.

**Cost of the wrong distribution.** Running the heuristic optimal for $D_1$ under $D_2$ gives an expected bin count of 2.51051 against the optimum 2.48615; running the heuristic optimal for $D_2$ under $D_1$ gives 3.2249 against 3.220233. The same witness prefix and the same disjoint optimal actions are obtained when the terminal cost in equation 9 is replaced by the gap of equation 1, so the conflict does not depend on scoring by the number of bins.

### B.5 DistSpace-OBP construction details

**Candidate generators.** Each DPM candidate independently draws truncated Gaussian components with

$$\mu_j \sim U[1, 100], \qquad \log \sigma_j \sim U[\log 2, \log 50],$$

and stick-breaking weights

$$\pi_j = V_j \prod_{l<j} (1 - V_l), \qquad V_j \sim \mathrm{Beta}(1, \alpha),$$

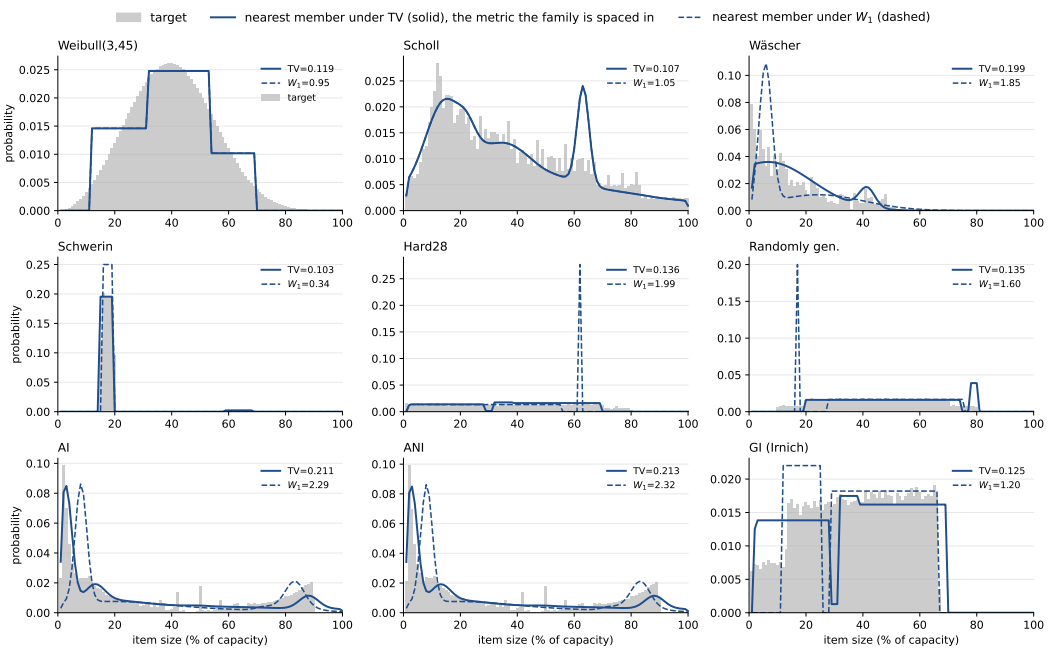


Figure 2: External targets (grey bars) and their nearest distributions in DistSpace-OBP under TV (solid) and $W_1$ (dashed). The two criteria select the same distribution for Weibull and Scholl. TV measures differing probability mass; $W_1$ measures displacement in percent of capacity, so their numerical values are not directly comparable.

truncated once the expected remaining mass falls below $10^{-9}$. The concentration parameter sets how weight is spread over components: the first component has expected weight $1/(1+\alpha)$, approximately $87\%$ for $\alpha = 0.15$ and $25\%$ for $\alpha = 3$. A uniform-interval candidate mixes 1–3 intervals whose lengths are log-uniform on $[1, 100]$, with weights drawn from $\mathrm{Dirichlet}(1, \ldots, 1)$. Candidates with $\max_x p_x > 0.5$ or item-size standard deviation below 1 are excluded. Farthest-point sampling starts from the candidate nearest the uniform distribution and uses seed 0. Every pair of selected distributions is at least the covering radius apart; their nearest-neighbor TV has median $0.191$ and 5th–95th percentiles $0.169$–$0.325$.

**Wasserstein-1 diagnostic.** As a secondary diagnostic, the Wasserstein-1 distance measures displacement along the size axis,

$$W_1(p, q) = \sum_{x=1}^{99} |F_p(x) - F_q(x)|, \qquad F_p(x) = \sum_{y \le x} p_y, \tag{10}$$

whose unit on the reference grid is one percent of bin capacity. Each of the nine external targets of Section 3.3 has a distribution in DistSpace-OBP within $0.3$–$2.3$ in $W_1$. TV and $W_1$ select the same nearest distribution for two of the nine targets, Weibull and Scholl; Figure 2 shows both choices for every target.

**Prefix coverage of the targets.** Table 6 lists, for each target of Section 3.3, the TV distance to its nearest distribution in the three nested prefixes $\mathcal{D}_m$ on which the DiSE-$m$ sets of Section 4.3 are evolved, and in the full family; the last row is the covering radius of each prefix over the candidate pool. Scholl and Hard28 have their nearest distribution already in $\mathcal{D}_{128}$; the other targets approach as $m$ grows.

**Instance-level distances.** The target distances of Section 3.3 are computed on pooled size distributions. Among the 6,035 individual BPPLIB instances, the TV distance to the nearest distribution

Table 6: TV distance from each benchmark target to its nearest distribution in the nested prefixes $\mathcal{D}_m$ and in DistSpace-OBP, and the covering radius of each over the candidate pool.

| Target | $\mathcal{D}_{128}$ | $\mathcal{D}_{1,024}$ | $\mathcal{D}_{4,096}$ | DistSpace-OBP |
|---|---|---|---|---|
| Weibull$(3, 45)$ | 0.425 | 0.374 | 0.236 | 0.119 |
| Scholl | 0.248 | 0.248 | 0.248 | 0.107 |
| Wäscher | 0.417 | 0.387 | 0.282 | 0.199 |
| Schwerin | 0.485 | 0.404 | 0.197 | 0.103 |
| Hard28 | 0.235 | 0.235 | 0.235 | 0.136 |
| Random | 0.340 | 0.340 | 0.265 | 0.135 |
| AI | 0.403 | 0.343 | 0.317 | 0.211 |
| ANI | 0.402 | 0.344 | 0.317 | 0.213 |
| Irnich | 0.371 | 0.365 | 0.235 | 0.125 |
| Covering radius | 0.664 | 0.473 | 0.324 | 0.167 |

in DistSpace-OBP has median $0.253$ and maximum $0.609$; the per-family medians range from $0.150$ (Schwerin) to $0.358$ (Wäscher). In the other direction, $57\%$ of the family's distributions are farther than $0.4$ in TV from every benchmark instance, so the family contains many profiles that no benchmark instance resembles.

**Sequence generation.** To instantiate a distribution at capacity $C$, each item draws a size cell of the reference grid from $p$, then a value uniform within that cell, which is scaled to $C$, rounded, and clipped to $\{1, \ldots, C\}$. Sequences use seeded common random numbers across distributions, and a shorter sequence is a prefix of the longer one for the same distribution. The mixed variant of Section 3.3 assigns capacity and length by cycling through $C \in \{100, 1000\}$ and $n \in \{200, 1000, 5000\}$ along the greedy order.

## C DiSE DETAILS

With the training environment fixed, the remaining task is to search for a small set under a manageable evaluation budget. DiSE builds on EoH-S (Liu et al., 2026b): it retains the thought-and-code representation, complementary search operators, and greedy population management, while replacing repeated evaluation on one fixed instance set with successive distribution batches. The central requirement is that all performance vectors used in a comparison refer to the same batch.

**Training objective and batch schedule.** Let $\mathcal{D}_{\mathrm{tr}} \subseteq \mathcal{D}$ contain the $N$ distributions used in a run, with one fixed sequence $L_D$ per distribution as described in Section 3.3. The empirical design objective is

$$\widehat{F}_{\mathrm{tr}}(S) = \frac{1}{N} \sum_{D \in \mathcal{D}_{\mathrm{tr}}} \min_{A \in S} \mathrm{gap}(A, L_D). \tag{11}$$

At fixed capacity and length, this is a one-sequence-per-distribution estimate of equation 2 on the training subset. In the mixed setting, each $L_D$ instead uses its assigned capacity and length.

A seeded random permutation of $\mathcal{D}_{\mathrm{tr}}$ is partitioned into batches of at most $B$ distributions, and a fresh permutation is used for each epoch. Generation $g$ uses batch $\mathcal{B}_g$ and its associated sequences. Retaining the final, possibly smaller batch visits every training distribution once per epoch; a run that drops that batch omits the shuffled remainder and must report this setting. The sequence attached to each distribution stays fixed across epochs. The changing batch therefore changes which distributions guide selection, without resampling their instances.

**Candidate generation and evaluation.** Initialization prompts the LLM with the task description and function interface to obtain an initial population of $k$ valid heuristics. Each candidate consists of a natural-language description and executable code, as in EoH and EoH-S (Liu et al., 2024a; 2026b). During evolution, the EoH-S operators generate offspring either from parents with different performance profiles or by revising a single parent. Each valid candidate is evaluated on the current

**Algorithm 2** DiSE: generation-level evolution protocol

**Require:** Training sequences $\{L_D : D \in \mathcal{D}_{\mathrm{tr}}\}$; population size $k$; batch schedule $(\mathcal{B}_g)$; search budget

Generate $k$ valid initial heuristics with the EoH-S initialization prompt
Evaluate them on $\mathcal{B}_0$ to obtain population $P$
**for** each subsequent generation $g$ within the search budget **do**
    Activate $\mathcal{B}_g$ and re-evaluate every member of $P$ on it
    Generate offspring $O$ using the EoH-S search operators
    Evaluate $O$ on $\mathcal{B}_g$; discard invalid evaluations
    Ensure every vector in $P \cup O$ refers to $\mathcal{B}_g$
    $P \leftarrow$ greedy selection of $k$ members from $P \cup O$ using $\widehat{F}_g$
**end for**
**return** $P$

batch, yielding the gap vector

$$\mathbf{v}_g(A) = \big(\mathrm{gap}(A, L_D)\big)_{D \in \mathcal{B}_g}.$$

These vectors provide both the performance information used in parent selection and the instance-level comparisons used to select survivors. The algorithm retains the EoH-S prompts and operators; its change is in the evaluation schedule and the handling of scores when that schedule advances.

**Selection by contribution to the set.** After collecting offspring, we combine them with the current population and apply EoH-S's complementary population management. For any candidate set $U$, define its set gap on the active batch as

$$\widehat{F}_g(U) = \frac{1}{|\mathcal{B}_g|} \sum_{D \in \mathcal{B}_g} \min_{A \in U} \mathrm{gap}(A, L_D). \tag{12}$$

Selection starts with the candidate having the lowest mean gap. It then repeatedly adds the remaining candidate that maximally decreases $\widehat{F}_g$, until $k$ members are selected. This rewards a candidate for improving instances not already handled well by selected members. It does not require each survivor to be best on some instance: marginal improvements can be zero, in which case additional members may still be retained.

**Re-scoring at a batch change.** A score vector from $\mathcal{B}_g$ cannot be compared coordinate-wise with one from $\mathcal{B}_{g+1}$, because their coordinates refer to different instances. Before the next generation uses the new batch, DiSE re-evaluates every survivor on it. Offspring evaluated on an earlier batch must likewise be re-evaluated on the active batch before entering its candidate pool. Algorithm 2 summarizes this generation-level protocol; batch identity must be preserved when implementing evaluations asynchronously.

**Evaluation cost and scope.** Each new candidate costs $|\mathcal{B}_g|$ packing runs rather than $N$ for full-training-set evaluation. Refreshing the population adds $k|\mathcal{B}_g|$ runs per generation, giving a basic cost of $(Q + k)|\mathcal{B}_g|$ for $Q$ evaluated offspring, plus re-evaluations for offspring that cross a batch change. Batching limits the cost of each comparison while exposing the search to successive parts of the training family.

## D PREFIX-BASED SELECTION DETAILS

DiSE returns a set of heuristics, while the best-of-set objective measures its potential with hindsight. To use this set online, we need a rule for choosing a member from information available before its decisions are made. We use an *indicator*, or selector, that observes an initial item prefix and chooses one rule for the remaining sequence. This separates the quality of the evolved set from the ability to identify a useful member during execution.

**A single-switch protocol.** Let $H_M = (a_1, \ldots, a_M)$ be a prefix of fixed length $M$, and let $\mathbf{b}_M$ be the residual-capacity vector after packing it with Best Fit. A selector chooses an action from $\mathcal{A} = S \cup \{\mathrm{BF}\}$, where an action means a heuristic for packing the suffix, not a single bin assignment:

$$A^\star = \sigma(H_M, \mathbf{b}_M; C, n) \in \mathcal{A}.$$

The selected heuristic continues from $\mathbf{b}_M$ in the true arrival order, with no repacking or further switches. Choosing Best Fit corresponds to *stay*. In the reported experiments, $|S| = 10$, giving 11 actions, and instances with $n \leq M + 1$ stay on Best Fit throughout; switching therefore requires at least two remaining items.

Capacity $C$ and total length $n$ are treated as known instance metadata. The selector cannot read future item sizes or the generating distribution. The evolved members retain the fixed interface of Section 3.1; prefix-based selection adds a history-dependent choice over those members. It is therefore a separate policy component from the individual decision functions analyzed in Section 3.2.

**Retrieval from a reference family.** The first selector represents the prefix by its empirical histogram $\hat{p}_M$ on the reference grid of Section 3.3. Given a finite reference library $\mathcal{R}$ and an offline cost table $T(D, A)$, it retrieves a nearby distribution and adopts that row's preferred action:

$$D^\star \in \arg\min_{D \in \mathcal{R}} d(\hat{p}_M, p_D), \qquad A^\star_{\mathrm{ret}} \in \arg\min_{A \in \mathcal{A}} T(D^\star, A). \tag{13}$$

We consider TV equation 5 and $W_1$ equation 10 as the retrieval distance $d$. Each table entry estimates the final bin count when Best Fit packs a reference prefix and action $A$ packs the remainder, averaged over independent reference sequences. Table construction uses no items from the test instance. The reference library can be instantiated with a distribution family such as DistSpace-OBP; its version, capacities, sequence lengths, switch point, and replication count are part of the table's specification.

Retrieval pays for action evaluation offline. At deployment, it estimates a distribution and looks up an action. The retrieved ranking averages over reference packing states and horizons, so it need not match the ranking from the actual state $\mathbf{b}_M$.

**Lookahead from the observed packing state.** The second selector evaluates the actions directly from $\mathbf{b}_M$. Let $\hat{D}_M = M^{-1} \sum_{i=1}^{M} \delta_{a_i}$ be the empirical distribution of the observed sizes. We sample $R$ synthetic suffixes $\widetilde{L}^{(r)} \sim \hat{D}_M^{n-M}$ by bootstrap, retaining the original observed sizes rather than their reference-grid bins. For every suffix, each action starts from a copy of the same state $\mathbf{b}_M$. Writing $c(A; \mathbf{b}_M, \widetilde{L})$ for the resulting final bin count, including bins already used by the prefix, the selector is

$$A^\star_{\mathrm{sim}} \in \arg\min_{A \in \mathcal{A}} \frac{1}{R} \sum_{r=1}^{R} c(A; \mathbf{b}_M, \widetilde{L}^{(r)}). \tag{14}$$

All actions use the same simulated suffixes, making their cost comparisons paired. Only the selected action is then applied to the actual suffix. Simulated alternatives do not change the real packing state. This selector needs no reference library or offline action-cost table; the experiments use $R = 10$.

The simulation model treats the prefix as representative of future arrivals and draws future items independently. With a short prefix, unobserved sizes cannot appear in the bootstrap, and a finite simulation budget adds selection noise. Its advantage over retrieval is that it evaluates the actual residual state and remaining horizon, without replacing the empirical distribution by a library member.

**Information, commitment, and computation.** A longer prefix can improve distribution estimation, but it also commits more items to Best Fit and leaves fewer decisions for the chosen heuristic. Retrieval relies on the coverage and calibration of its reference table; lookahead instead pays for $R|\mathcal{A}|$ simulated continuations online. Both can choose *stay*.

# E Additional experimental results

## E.1 A heuristic designed on one distribution does not carry over

**Question.** Theorem 1 states that no single heuristic is optimal for every item distribution. How does this show for heuristics produced by LLM-based design on a single generator?

Table 7: Gap (%) of heuristics designed on Weibull$(3, 45)$ when the item-size distribution changes; 5 instances of 5,000 items, $C = 100$. The first column is the design distribution.

| Heuristic | Weibull | Uniform | Normal | Bimodal | Skew-S | Skew-L | Small | Large | Const. |
|---|---|---|---|---|---|---|---|---|---|
| Best Fit | 4.21 | **2.16** | **2.10** | 0.47 | **0.36** | 18.29 | **0.08** | **25.84** | **11.11** |
| First Fit | 4.55 | 3.19 | 3.37 | 0.76 | 0.48 | 18.43 | **0.08** | **25.84** | **11.11** |
| EoH (published) | **0.70** | 2.40 | 2.68 | 1.18 | 0.73 | 18.29 | 0.80 | **25.84** | 122.22 |
| EoH-1 | 0.74 | 2.36 | 2.62 | **0.45** | 0.52 | **18.28** | **0.08** | **25.84** | 122.22 |
| EoH-2 | 0.83 | 2.74 | 2.86 | 1.36 | 0.56 | 18.29 | 33.62 | **25.84** | 122.22 |
| EoH-3 | 0.97 | 2.26 | 2.31 | 0.75 | 2.28 | 18.29 | 0.27 | **25.84** | 122.22 |

Table 8: Improvement over Best Fit, $\mathrm{gap}(\text{Best Fit}) - \mathrm{gap}(\sigma)$, mean over the $6{,}064$ instances of the BPPLIB suite in percentage points; larger is better; bold marks the best prefix selector per column. Computed with an earlier DiSE run and library, see the text.

| Strategy | Information used | $M = 50$ | $M = 100$ | $M = 200$ |
|---|---|---|---|---|
| Best fixed action | full suite, in hindsight | $+0.087$ | $+0.127$ | $+0.102$ |
| Retrieval, TV nearest neighbour | prefix | $+0.061$ | $+0.171$ | $+0.154$ |
| Retrieval, $W_1$ nearest neighbour | prefix | $+0.125$ | $+0.186$ | $+0.148$ |
| Lookahead simulation | prefix | $\mathbf{+0.212}$ | $\mathbf{+0.236}$ | $\mathbf{+0.175}$ |
| Oracle | entire instance | $+0.834$ | $+0.626$ | $+0.402$ |

**Protocol.** We take four heuristics designed on Weibull$(3, 45)$—the heuristic published with EoH and the best final member of each of our three EoH runs (DeepSeek-v4-pro on the original design set of EoH, five Weibull$(3, 45)$ instances of 5,000 items at $C = 100$)—and evaluate them, together with Best Fit and First Fit, on nine item-size distributions: the design distribution and eight others (uniform, normal, bimodal, two skewed Beta shapes, small items only, large items only, and a constant size of 45). Each distribution contributes five instances of 5,000 items at $C = 100$.

**Results.** On the design distribution every evolved heuristic is far ahead of Best Fit ($0.70$–$0.97\%$ against $4.21\%$; Table 7). On the other eight distributions the advantage is gone: no evolved heuristic improves on Best Fit by more than $0.02$ points anywhere, and all four are behind it on the uniform and normal distributions. Two failures are qualitative. With a constant item size of 45, all four evolved heuristics place every item in its own bin ($122.22\%$, against $11.11\%$ for Best Fit, which pairs the items). On small items one of our runs reaches $33.62\%$ where Best Fit is at $0.08\%$. The two columns with items larger than half the capacity (Skew-L in part, Large entirely) do not separate the heuristics, because almost every item needs its own bin whatever the rule.

**Conclusion.** The advantage that LLM-based design obtains on its design distribution does not transfer, and which distribution is fatal differs from one evolved heuristic to the next. This is the empirical counterpart of Theorem 1: the theorem rules out a single rule optimal across distributions, and here a single evolved rule does not transfer between them.

### E.2 Prefix-based selection: references and discussion

**BPPLIB suite.** Table 8 reports the same comparison on all $6{,}064$ instances of the BPPLIB suite. All selectors improve on Best Fit at every switch point, with lookahead consistently strongest. At $M = 100$, lookahead gains $0.236$ points, compared with $0.171$ for TV retrieval and $0.186$ for $W_1$ retrieval, and exceeds the best fixed action chosen in hindsight ($0.127$), showing the benefit of instance-dependent selection. The per-instance oracle gains $0.626$ points at $M = 100$, indicating substantial room for better selection.

**Reference library and cost table.** The retrieval results of Table 5 use DistSpace-OBP itself as library. Its cost table averages ten independently generated sequences per distribution at $C = 100$, $n = 1{,}000$, and reference switch point $M_{\mathrm{ref}} = 500$; a prefix at $C = 500$ is mapped to the same grid by dividing sizes by the capacity. Both nearest-neighbour selectors use this same table; their only difference is the retrieval metric. The table's scale and switch point therefore differ from those

Table 9: Table 5 at $M = 200$ by group (items/capacity, 5 instances each), in percentage points; larger is better.

| Strategy | 1k/100 | 5k/100 | 10k/100 | 1k/500 | 5k/500 | 10k/500 |
|---|---|---|---|---|---|---|
| Best fixed action | $-0.396$ | $+3.267$ | $+3.431$ | $-0.250$ | $-0.050$ | $-0.273$ |
| Retrieval, TV nearest neighbour | $+1.887$ | $+2.373$ | $+2.229$ | $+0.000$ | $+0.000$ | $+0.000$ |
| Retrieval, $W_1$ nearest neighbour | $+1.887$ | $+2.373$ | $+2.229$ | $+0.000$ | $+0.000$ | $+0.000$ |
| Lookahead simulation | $+1.887$ | $+2.719$ | $+3.187$ | $+0.000$ | $-0.050$ | $+0.000$ |
| Oracle | $+1.887$ | $+3.267$ | $+3.431$ | $+0.000$ | $+0.000$ | $+0.050$ |

of the evaluated instance, and this mismatch is part of what retrieval pays for evaluating actions offline. The BPPLIB results of Table 8 were computed with an earlier DiSE run, evolved on the $4{,}096$-distribution prefix under $W_1$, and with a library of $20{,}000$ distributions built from the same candidate pool as DistSpace-OBP by the same farthest-point procedure with $W_1$ in place of TV.

**Switch rule.** Instances with $n \leq M + 1$ items stay on Best Fit throughout, and in the means of Table 8 every unswitched instance contributes zero; every Weibull instance is longer than $M + 1$.

**Hindsight references.** The fixed-action reference chooses one suffix action for all instances, selected by average improvement across the full evaluation set and the three switch points; it is therefore an offline reference and not the best fixed action separately at every switch point. The oracle chooses, for each instance, the lowest gap among the 11 suffix actions continued from the same Best Fit prefix state. It is specific to the Best Fit prefix and the switch point, and differs from choosing the best set member to run from the start.

**Per-group results on the Weibull benchmark.** Table 9 splits Table 5 at $M = 200$ by instance length and capacity. The gains come from the three groups at $C = 100$, where Best Fit has a gap of $4.0$ to $4.5\%$; at $C = 500$ its gap is $0.5$ to $1.2\%$ and even the oracle removes at most $0.05$ points at any switch point, so every method is close to zero there. The best fixed action loses on the $1{,}000$-item instances and at $C = 500$, where the selectors keep Best Fit or find another member.

**Significance and what the comparison does not show.** On the Weibull benchmark, under a paired bootstrap over instances, lookahead exceeds both retrieval selectors by $0.209$ points at $M = 200$ (95% interval $0.061$–$0.377$); at $M = 50$ and $M = 100$ no difference between the three selectors is significant. On the BPPLIB suite lookahead exceeds TV retrieval by $0.173$ points at $M = 50$ (95% interval $0.132$–$0.215$) and by $0.089$ at $M = 100$ ($0.059$–$0.120$), and exceeds $W_1$ retrieval by $0.100$ ($0.062$–$0.137$) and $0.068$ ($0.042$–$0.095$), on the instances where a switch is possible. The comparison does not isolate the cause of the advantage: the selectors differ in their distribution approximation, in conditioning on the actual packing state, in simulation noise, and in the switch-point mismatch of the reference table. The decrease of the oracle gain from $M = 50$ to $M = 200$ is an observed result rather than a general property of switching: different prefix lengths induce different packing states, and a longer prefix leaves fewer decisions to the chosen member.