ACTOR SCORE COMPOSITION AND REFINEMENT FOR MULTI-OBJECTIVE FLEXIBLE JOB SHOP SCHEDULING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-objective flexible job shop scheduling has attracted growing research interest due to the need to balance competing objectives in production. Current neural approaches use preferences to condition model features, parameters, or computation paths, enabling a single model to construct schedules across different trade-offs. However, generating high-quality schedule sets is challenging because a single model needs not only to learn assignment and sequencing rules required across preferences but also to continually adapt its decisions to changing machine workloads and remaining operations. To address this challenge, we propose Actor Composition with Preference Correction (AC-PC), which constructs preference-conditioned scheduling policies by composing and refining independent actor scores. Specifically, a shared conditional encoder represents the current scheduling state at fixed preference anchors, and independent actors produce action scores that are combined using weights determined by the requested preference. Moreover, preference correction uses the requested preference and mixed actor features to apply an action-specific adjustment to the composed scores. To evaluate AC-PC, experiments are conducted on same-size synthetic instances, larger unseen instances, and public benchmarks. The results show that AC-PC outperforms baselines in most cases.

1 INTRODUCTION

Multi-objective flexible job shop scheduling assigns each operation to an eligible machine and orders operations on each machine (Dauzère-Pérès et al., 2024). Feasible schedules must respect job precedence and machine capacity while balancing objectives such as makespan, tardiness, and processing cost (Tamssaouet et al., 2022; Xiao et al., 2024). Different preferences favor different trade-offs, so the goal is to generate a high-quality schedule set that covers them.

Preference-conditioned neural policies learn from training instances to construct solutions for different trade-offs with a single model (Lin et al., 2022; Wang et al., 2024b). Graph-based scheduling methods encode operations, machines, and their relationships during construction (Song et al., 2023; Wang et al., 2024a; Smit et al., 2025); MOFJSP policies further condition this process on objective weights (Su et al., 2024; Smit et al., 2026). We study how action scoring can improve the quality of the generated schedule set, focusing on two challenges.

The first challenge is to accommodate preference-specific decisions within one policy. Machine assignment and operation sequencing are coupled: assigning an operation to a fast machine may shorten its processing time but delay another job competing for that machine, whereas a slower alternative may reduce cost while changing later completion times. Job precedence propagates these effects, and preferences change their relative importance. Different preferences may therefore require different scoring rules for the same candidate actions. Insufficient separation between these rules can limit the quality of the generated schedule set.

The second challenge is to make action-level trade-offs that account for both the requested preference and the current scheduling state. Even under a fixed preference, a fast machine may be attractive when available but cause long waits when heavily loaded. Changing machine workloads and available operations can therefore alter which assignment is desirable.

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.

To address these challenges, we propose Actor Composition with Preference Correction (AC-PC), which composes independent actor scores and refines them at each scheduling step. Actor Composition (AC) uses a shared conditional encoder to represent the partial schedule at fixed preference anchors. Independent actors learn separate scoring rules for candidate operation–machine pairs, and weights determined by the requested preference compose their scores.

Each actor already scores the current scheduling state, while the composition coefficients depend only on the requested preference. The same coefficients combine the actor scores for every candidate action. Preference correction adds a learned, action-specific adjustment using the requested preference and mixed actor features. It refines the composed scores while leaving the requested preference unchanged. A gate makes the adjustment vanish at preference endpoints or simplex vertices. Masking and softmax then convert the final scores into a distribution over feasible actions.

Our contributions are threefold. First, AC combines independently parameterized actor scores from fixed anchor-conditioned representations under a shared encoder, with all branches trained jointly through the composed policy. Second, preference correction refines the composed scores with an action-specific adjustment conditioned on the requested preference and mixed actor features, preserving the base scores at preference endpoints or simplex vertices. Third, we evaluate AC-PC on synthetic instances at the training sizes, larger unseen instances, and public benchmarks. Ablations and an actor-parameter-matched comparison assess independent scoring and score refinement; further analysis examines how correction changes scheduling decisions.

2 Related work

Scheduling methods must coordinate operation selection with machine assignment. Graph policies learn dispatching from the evolving partial schedule (Zhang et al., 2020; Park et al., 2021), while multi-action reinforcement learning separates operation and machine decisions into two sub-policies (Lei et al., 2022). Self-labeling trains from the best sampled schedule (Corsini et al., 2024). For multiple objectives, particle swarm optimization with local search (Moslehi & Mahnam, 2011) and decomposition-based evolutionary search (Xiao et al., 2024) obtain solution sets through repeated search; reinforcement learning has also addressed dynamic multi-objective flexible job shops (Luo et al., 2021). AC-PC constructs schedules directly for a requested preference.

Neural multi-objective optimization methods either construct solutions directly or use learned decisions within iterative search. Pareto Improver learns preference-conditioned 2-opt moves (Zheng et al., 2024); WRPN guides evolutionary improvement (Ye et al., 2025); and Neuro-PLS learns solution selection and neighborhood exploration in Pareto local search (Zhang et al., 2026).

Representation and generation. Graph policies for flexible job shop scheduling encode operations and machines at each construction step (Song et al., 2023; Wang et al., 2024a). Multi-objective extensions add preferences and objective-specific scheduling information (Su et al., 2024; Smit et al., 2026), providing the state information needed to evaluate candidate actions.

Preference information can act on features, parameters, or computation paths. P-MOCO generates preference-specific decoder parameters with a hypernetwork (Lin et al., 2022), and WSDAM uses a weight-specific attention decoder (Ye et al., 2022). CNH combines preference–instance attention with a size-aware decoder (Fan et al., 2025a), while Neat-Weights learns weight-specific representations for weight–instance interactions (Chen et al., 2025). CFAN forms preference-dependent context through cross-fusion attention (Fu et al., 2025), WeCon conditions both encoding and decoding (Wu et al., 2026), and POCCO routes subproblems to specialized architectures (Fan et al., 2025b). AC-PC combines independently parameterized actor scores from fixed anchor-conditioned representations for the requested preference, then refines the result before action selection.

Sharing representations while retaining separate output mappings is common in multi-task learning (Caruana, 1997; Ruder, 2017). In AC-PC, the encoder supplies anchor-conditioned scheduling features, independent actors provide distinct scoring mappings, and all branches train jointly through the composed policy.

Training and adaptation. Multi-objective sequential decision making studies how utility specifications determine useful policies or policy sets (Roijers et al., 2013; Hayes et al., 2022). Weight-

conditioned value networks support changing objective weights (Abels et al., 2019), and generalized value learning adapts policies across linear preferences (Yang et al., 2019). In our setting, the preference is fixed within a rollout, but the partial schedule changes after every assignment; action scores must therefore depend on both.

Other methods change how decision rules are learned and reused. DRL-MOA transfers parameters between neighboring scalarized subproblems (Li et al., 2021); EMNH learns a shared meta-model and derives subproblem models through fine-tuning (Chen et al., 2023a); and MODRL/D-EL refines a population of policy parameters through evolutionary learning (Zhang et al., 2021). Robust training instead targets difficult preference regions with adversarial instances (Liu et al., 2026). AC-PC trains all actors jointly through one composed PPO policy and reuses their scoring mappings across requested preferences.

Pareto set coordination. Solving each preference independently can produce repeated solutions and gaps in the approximate front. NHDE combines the instance graph, current Pareto-front graph, and indicator information to promote diversity (Chen et al., 2023b). CDE combines node- and solution-level context with a hypervolume residual update strategy (Lu et al., 2025), while Wang et al. (2023) learn which preference weights to query. These methods coordinate across generated solutions or select trade-offs. AC-PC refines candidate action scores at each construction step; evaluation uses fixed preference grids followed by Pareto filtering.

3 Problem formulation

Let $\mathcal{J}$ be a set of jobs and $\mathcal{M}$ a set of machines. Job i contains n_i operations $O_{i1}, \ldots, O_{in_i}$ that must be processed in order. Operation O_{ij} can run on a machine $k \in \mathcal{M}_{ij} \subseteq \mathcal{M}$ and takes time $p_{ij}^k > 0$. A feasible schedule x assigns an eligible machine and a nonnegative start time to each operation, respecting job precedence, non-preemption, and machine capacity. Processing durations are fixed by the selected machines, and operations assigned to the same machine cannot overlap.

With job completion time $C_i(x)$, job start time $S_i(x)$, due date D_i, and assigned processing time $p_{ij}(x)$, the objectives are

$$f_{\mathrm{mk}}(x) = \max_i C_i(x), \qquad f_{\mathrm{td}}(x) = \sum_i \max\{C_i(x) - D_i, 0\}, \tag{1}$$

$$f_{\mathrm{fl}}(x) = \frac{1}{|\mathcal{J}|} \sum_i \big(C_i(x) - S_i(x)\big), \qquad f_{\mathrm{co}}(x) = \sum_{i,j} \big(p_{\max} - p_{ij}(x)\big). \tag{2}$$

Here $p_{\max} = \max_{i,j} \max_{k \in \mathcal{M}_{ij}} p_{ij}^k$ is the largest eligible processing time in the instance. This cost proxy assigns higher cost to faster processing choices. With $S_i(x)$ denoting the first operation's start, f_{fl} measures mean in-process duration, excluding waiting before first dispatch. Due dates use $D_i = 1.5 \sum_j \min_{k \in \mathcal{M}_{ij}} p_{ij}^k$. We consider makespan–costs, tardiness–costs, and makespan–flowtime–costs, with every objective minimized.

For d objectives, a preference w lies in $\Delta^{d-1} = \{w \in \mathbb{R}^d : w_i \geq 0, \sum_i w_i = 1\}$. Following the benchmark, we minimize $\sum_i w_i \widetilde{f}_i(x)$, where $\widetilde{f}_i(x) = \alpha_i f_i^{\mathrm{norm}}(x)$. The environment normalizes processing times and costs, and $\alpha_i > 0$ is fixed for each training size (Appendix A); w therefore weights these scaled objectives. Weighted-sum optima cover supported trade-offs, leaving unsupported regions of non-convex fronts outside this scalarization (Ehrgott, 2005; Marler & Arora, 2010). At inference, schedules from a finite preference set are pooled and Pareto filtered.

3.1 Scheduling state and actions

At construction step t, state s_t contains the partial schedule and operation, machine, and pair features. An action $a_t = (O_{ij}, k)$ selects a candidate operation and an eligible machine. Let $\mathcal{A}(s_t)$ denote the feasible action set. The policy selects an action from this set, and the environment schedules the operation and updates the state.

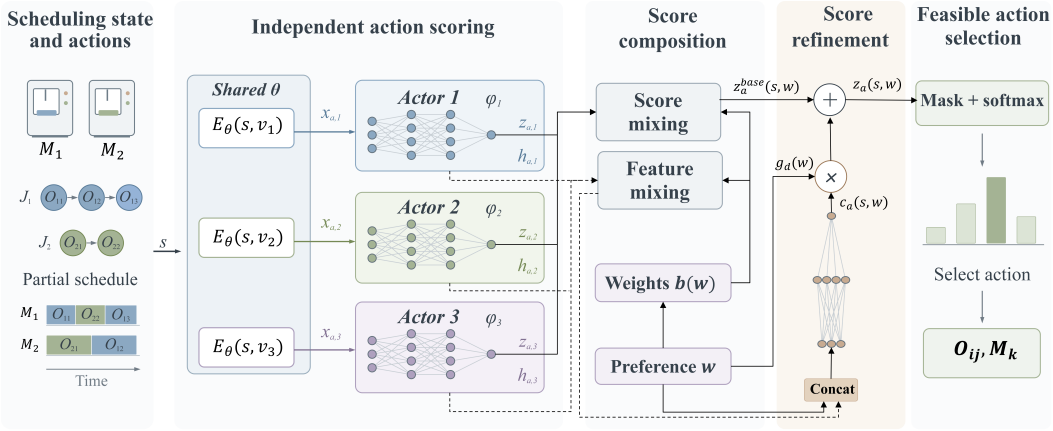


Figure 1: AC-PC architecture. A shared encoder represents the scheduling state at fixed preference anchors; independent actors output action scores and hidden features. Preference-dependent weights mix both. The correction network concatenates mixed features with the requested preference to produce $c_a(s, w)$. The $\times$ node applies the gate $g_d(w)$, and the $+$ node adds the gated correction to the composed score before masking and softmax. Dashed paths carry actor features.

4 AC-PC: Composing and Refining Actor Scores

AC-PC (Figure 1) uses DCAN's objective definitions, conditional encoder, scheduling environment, feasibility constraints, reward formulation, and PPO training framework (Smit et al., 2026). It composes independent actor scores at fixed preference anchors and refines them through preference correction (PC). For candidate action a in state s and requested preference w, the final score is

$$z_a(s, w) = \underbrace{\sum_{k=1}^{3} b_k(w) z_{a,k}(s)}_{\text{score composition}} + \underbrace{g_d(w) c_a(s, w)}_{\text{score refinement}}.$$

Here $z_{a,k}$ is an actor score, $b_k(w)$ is a preference-dependent composition coefficient, and $g_d(w)$ gates the correction $c_a(s, w)$. These scores are logits, converted into action probabilities by masking and softmax.

At each construction step, the shared encoder represents candidate operation–machine pairs at three fixed anchors. Independent actors produce scores and hidden features, both mixed using the same preference-dependent coefficients. PC uses the mixed features and requested preference to refine the composed scores. Selecting and executing a feasible action updates the state. The encoder, actors, and correction are trained jointly through the final policy.

4.1 Independent Action Scoring

We use three fixed anchors $v_1, v_2, v_3 \in \Delta^{d-1}$ and a shared conditional encoder evaluated at each anchor. For candidate action a,

$$x_{a,k} = [E_\theta(s, v_k)]_a, \qquad z_{a,k} = A_{\phi_k}(x_{a,k}), \qquad k = 1, 2, 3. \tag{3}$$

The vector $x_{a,k}$ concatenates the candidate operation embedding, machine embedding, global operation and machine embeddings, and pair features. Encoder parameters θ are shared, while each actor has its own parameters ϕ_k. Let $h_{a,k} \in \mathbb{R}^H$ denote its final hidden representation before scalar scoring, where $H = 64$ is the actor hidden width.

All branches score the same candidate actions in the same state, so scores with the same action index can be combined directly.

4.2 SCORE COMPOSITION

The requested preference w determines the composition coefficients $b_k(w)$, while the encoder receives the fixed anchors. We instantiate these coefficients using the standard interval and triangular Bernstein bases (Farin, 2002, Chapters 4 and 18), applying them to actor scores and hidden features as specified below.

4.2.1 ANCHORS AND BASES

Two objectives: quadratic basis. For $w = (w_1, w_2)$, we use

$$v_1 = (1, 0), \quad v_2 = (\tfrac{1}{2}, \tfrac{1}{2}), \quad v_3 = (0, 1), \qquad b(w) = (w_1^2,\, 2w_1w_2,\, w_2^2). \tag{4}$$

These complete degree-two Bernstein coefficients define quadratic Bernstein composition on an interval. At an endpoint, only its corresponding branch contributes. At the midpoint, the coefficients are $(1/4, 1/2, 1/4)$, combining the midpoint branch with both endpoint branches.

Three objectives: linear basis. For three objectives, we use three branches with simplex vertices $v_k = e_k$, where e_k is the kth standard basis vector, and coefficients $b_k(w) = w_k$. These form the complete degree-one Bernstein basis on the triangular simplex. A complete degree-two basis would require six encoder–actor branches, with coefficients $w_1^2, w_2^2, w_3^2, 2w_1w_2, 2w_1w_3, 2w_2w_3$. We use the linear basis to keep the same three-branch computation as in the two-objective model.

4.2.2 SCORE AND FEATURE MIXING

Both bases satisfy $b_k(w) \geq 0$ and $\sum_k b_k(w) = 1$. We use the same coefficients to combine action scores and hidden representations:

$$z_a^{\text{base}}(s, w) = \sum_{k=1}^{3} b_k(w) z_{a,k}, \tag{5}$$

$$\bar{h}_a(s, w) = \sum_{k=1}^{3} b_k(w) h_{a,k}. \tag{6}$$

Gradients from the composed policy jointly train the branch features and the correction network that consumes their mixture.

Each base score lies between the smallest and largest branch scores for that action. At a fixed state, base scores vary quadratically with preference for two objectives and linearly for three. The mixed features retain candidate-action information before scalar scoring for the correction in Section 4.3.

4.3 SCORE REFINEMENT

We concatenate the mixed features $\bar{h}_a$ with the requested preference w and compute

$$c_a(s, w) = u^\top \tanh\big(W[\bar{h}_a(s, w); w] + b\big) + \beta, \tag{7}$$

$$g_d(w) = \frac{d}{d-1}\left(1 - \sum_{i=1}^{d} w_i^2\right), \qquad z_a(s, w) = z_a^{\text{base}}(s, w) + g_d(w) c_a(s, w). \tag{8}$$

Here $W \in \mathbb{R}^{64\times(H+d)}$, $b, u \in \mathbb{R}^{64}$, and $\beta \in \mathbb{R}$ are trainable parameters of the correction network. The mixed features make the correction action- and state-dependent: under the same preference, different candidate actions can receive different adjustments. Supplying w directly also distinguishes preferences that produce similar mixed features. The output layer is initialized to zero, so training begins with the base composition and learns an additive score correction. Network dimensions and score centering are specified in Appendix A.

For $d \geq 2$, the gate lies in $[0, 1]$, is zero at each simplex vertex, and equals one at the uniform preference. The final scores therefore equal the base scores at each vertex. For two objectives, $g_2(w) = 4w_1w_2$; at the midpoint, all three branches contribute and the gate equals one. For three objectives, $g_3(w) = \frac{3}{2}(1 - \sum_i w_i^2)$ and is generally nonzero along an edge; for example, $g_3(1/2, 1/2, 0) = 3/4$.

4.4 FEASIBLE ACTION SELECTION

After composition and correction, invalid-action masking (Huang & Ontañón, 2022) restricts the policy to the feasible set:

$$\pi(a \mid s, w) = \begin{cases} \dfrac{\exp z_a(s, w)}{\sum_{a' \in \mathcal{A}(s)} \exp z_{a'}(s, w)}, & a \in \mathcal{A}(s), \\ 0, & a \notin \mathcal{A}(s). \end{cases} \tag{9}$$

The softmax normalizes the final corrected scores over feasible actions. Greedy decoding selects the highest feasible score; sampling draws an action from π. The environment then updates the partial schedule. All branches are evaluated at the updated state, and construction continues until all operations are scheduled.

4.5 TRAINING AND INFERENCE

We train AC-PC with DCAN's multi-objective formulation of PPO (Schulman et al., 2017; Smit et al., 2026). For objective i, the reward is $r_{t,i} = \alpha_i[\widehat{f}_i^{\,\mathrm{norm}}(s_t) - \widehat{f}_i^{\,\mathrm{norm}}(s_{t+1})]$, using the partial-schedule estimates in Appendix A. The critic predicts an objective-value vector from mixed global embeddings, and actor advantages use preference-weighted rewards and values. The composed policy supplies the PPO likelihood ratio, jointly updating the encoder, actors, and correction. Appendix A specifies the training losses.

Training preferences follow a flat Dirichlet distribution; testing uses the fixed grids in Appendix A. Greedy evaluation constructs one schedule per preference, while sampling constructs ten. Returned schedules are pooled and Pareto filtered. Each new preference requires a rollout with three anchor-conditioned encoder evaluations per state, without retraining.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

We train a separate model for each task at 10×5, 20×5, 15×10, and 20×10 (jobs $\times$ machines), and test on 100 instances per task and size. Models use 1,500 PPO updates and validation-based checkpoint selection, with optimization details in Appendix A.

Baseline results for NSGA-II (Deb et al., 2002), MOEA/D (Zhang & Li, 2007), Hyper, WI-DAN, DCAN, and time-limited CP-SAT (Perron et al., 2023) are taken from Smit et al. (2026), together with the benchmarks and evaluation protocols. We rerun NSGA-II and MOEA/D under reduced search budgets (Appendix I). AC-PC uses an NVIDIA GeForce RTX 5090 (Appendix A); evolutionary reruns use their respective evaluation platforms, and other runtimes retain the settings reported by Smit et al. (2026).

Neural methods use matched preference counts, with one greedy schedule or ten sampled schedules per preference. We report normalized hypervolume (HV; higher is better) (Guerreiro et al., 2022), its relative gap to time-limited CP-SAT, distinct nondominated objective vectors, and runtime. CP-SAT is excluded from method rankings. HV uses the benchmark bounds and reference points; Appendix A gives the calculation details.

The following sections examine same-size performance, the four-objective extension, transfer, and the roles of independent scoring and correction. Appendices F–L give detailed results on larger instances, three-objective SD2 instances, public benchmarks (mk (Brandimarte, 1993) and Hurink's sets (Hurink et al., 1994)), search and sampling budgets, and other scheduling problems. Appendix C varies the inference preference count.

5.2 PERFORMANCE ON THE TRAINING SIZES

AC-PC exceeds DCAN HV in all 16 two-objective comparisons (Table 1), and sampling improves over greedy decoding at every task and size. For example, at 20×10 makespan–costs, greedy/sampling HV is 0.8138/0.8251 versus DCAN's 0.8083/0.8200.

Table 1: Same-size two-objective synthetic tests, with 100 instances per task and size. HV, Pareto-set size (Nr. Sol.), and seconds per instance are means. Gap is in percent relative to CP-SAT. Best HV is bold and second best is underlined at the displayed precision, excluding CP-SAT.

			Metaheuristics		Greedy				Sampling				
	Size	Metric	NSGA-II	MOEA/D	Hyper	WI-DAN	DCAN	AC-PC	Hyper	WI-DAN	DCAN	AC-PC	CP-SAT
Makespan–costs	10×5	HV	0.7208	0.6616	0.5987	0.6999	0.7104	0.7273	0.6204	0.7581	0.7647	**0.7739**	0.8313
		Gap (%)	13.30	20.42	27.98	15.81	14.55	12.51	25.37	8.81	8.01	6.90	0.00
		Nr. Sol.	8.14	8.81	6.72	2.91	3.46	6.59	7.78	7.19	7.77	12.00	13.20
		Time (s)	247.85	254.85	0.57	0.57	0.87	0.40	2.26	2.37	3.00	1.82	—
	20×5	HV	0.5332	0.4807	0.5083	0.5536	0.5599	0.5910	0.5268	0.5696	0.5724	**0.6025**	0.6095
		Gap (%)	12.53	21.14	16.61	9.17	8.14	3.04	13.57	6.56	6.09	1.15	0.00
		Nr. Sol.	11.96	14.67	6.48	3.30	4.04	12.46	9.24	5.72	6.82	19.56	11.82
		Time (s)	652.04	614.81	1.41	1.47	1.99	0.92	8.07	8.51	10.43	6.07	—
	15×10	HV	0.6809	0.5570	0.7039	0.7650	0.7723	0.7831	0.7388	0.7920	0.8002	**0.8084**	0.9243
		Gap (%)	26.33	39.73	23.85	17.23	16.44	15.27	20.07	14.31	13.42	12.54	0.00
		Nr. Sol.	14.12	16.70	6.95	8.28	9.14	12.04	9.17	14.16	15.42	19.41	18.16
		Time (s)	1694.42	1062.99	2.86	3.07	4.02	2.09	21.19	22.72	26.98	15.14	—
	20×10	HV	0.6289	0.4879	0.7751	0.7966	0.8083	0.8138	0.7236	0.8073	0.8200	**0.8251**	0.8548
		Gap (%)	26.43	42.93	9.32	6.82	5.44	4.79	15.36	5.57	4.07	3.48	0.00
		Nr. Sol.	19.21	19.45	16.43	9.44	13.85	16.19	7.43	14.14	22.66	26.46	14.81
		Time (s)	2798.44	1774.73	4.67	5.05	6.57	3.38	38.02	41.66	48.32	26.66	—
Tardiness–costs	10×5	HV	0.7741	0.7006	0.6335	0.7109	0.7460	0.7790	0.7103	0.8102	0.8272	**0.8370**	0.9144
		Gap (%)	15.35	23.38	30.72	22.25	18.42	14.81	22.32	11.40	9.54	8.46	0.00
		Nr. Sol.	15.41	9.58	6.52	2.66	4.36	8.48	11.44	10.28	12.89	18.45	25.81
		Time (s)	296.60	256.28	0.59	0.59	0.89	0.41	2.61	2.66	3.29	1.92	—
	20×5	HV	0.5331	0.4234	0.5479	0.6372	0.6396	0.6561	0.5282	0.6645	0.6700	**0.6809**	0.7010
		Gap (%)	23.96	39.60	21.84	9.09	8.76	6.41	24.64	5.20	4.42	2.86	0.00
		Nr. Sol.	20.53	17.59	10.23	10.00	10.88	14.24	10.87	17.27	19.51	24.33	11.26
		Time (s)	516.84	619.33	1.55	1.59	2.24	0.96	9.47	9.93	12.02	6.64	—
	15×10	HV	0.6745	0.5131	0.7134	0.7982	0.8094	0.8107	0.7270	0.8258	0.8338	**0.8392**	0.9505
		Gap (%)	29.04	46.02	24.95	16.03	14.85	14.71	23.51	13.12	12.28	11.71	0.00
		Nr. Sol.	22.10	18.96	13.50	15.45	17.04	19.44	12.10	26.95	32.39	36.33	20.71
		Time (s)	1642.20	1048.72	3.12	3.28	4.45	2.20	23.90	25.69	30.03	16.43	—
	20×10	HV	0.5878	0.4229	0.7566	0.8100	0.8112	0.8273	0.7472	0.8274	0.8306	**0.8386**	0.8498
		Gap (%)	30.83	50.24	10.97	4.68	4.53	2.65	12.08	2.63	2.25	1.32	0.00
		Nr. Sol.	24.54	23.23	18.23	17.04	20.03	24.17	12.02	29.56	34.01	37.38	14.54
		Time (s)	2869.48	1745.29	5.51	5.43	7.17	3.67	44.66	47.02	54.38	29.31	—

For three objectives, AC-PC exceeds DCAN in seven of eight comparisons (Table 2). The exception is 20×10 sampling, with HV 0.6264 versus 0.6314. NSGA-II remains stronger at 10×5.

5.3 Extension to four objectives

To evaluate an additional objective, we use makespan, average flowtime, total earliness, and costs with four vertex actors and four-dimensional preference correction. Appendix D specifies the extension and protocol. AC-PC exceeds DCAN HV in all eight comparisons (Table 3), with relative gains of 10.26%–28.01%. Sampling improves AC-PC at every size and achieves the best method HV at the three larger sizes; NSGA-II leads at 10×5.

5.4 Transfer without retraining

Transferring the 20×10 models to 30×10 and 40×10, AC-PC exceeds DCAN HV in 10 of 12 comparisons; WI-DAN remains stronger on tardiness–costs at both sizes. The 15×10 three-objective model also exceeds DCAN in all eight comparisons on mk, rdata, edata, and vdata. On mk, sampling HV is 0.3648 versus 0.3047. NSGA-II remains stronger on rdata and edata under the default sampling budget. Their costs, and those of vdata, are constant across schedules, so normalized HV on these three sets measures makespan and flowtime. Appendices F and H give the protocols and complete results.

Table 2: Same-size three-objective synthetic tests, with 100 instances per size. AC-PC uses three independent plain actors with vertex composition and preference correction. Metrics and ranking follow Table 1.

			Metaheuristics		Greedy				Sampling				
	Size	Metric	NSGA-II	MOEA/D	Hyper	WI-DAN	DCAN	AC-PC	Hyper	WI-DAN	DCAN	AC-PC	CP-SAT
Makespan–flowtime–costs	10×5	HV	**0.6146**	0.4969	0.4123	0.4081	0.4647	0.5163	0.4673	0.4902	0.5130	0.5567	0.6831
		Gap (%)	10.02	27.25	39.64	40.26	31.97	24.42	31.59	28.24	24.90	18.50	0.00
		Nr. Sol.	167.40	67.01	23.57	7.33	22.64	33.45	48.43	39.50	52.84	83.00	64.98
		Time (s)	229.21	260.88	0.84	0.82	1.14	0.47	4.99	4.97	5.66	2.55	—
	20×5	HV	0.3661	0.2595	0.3800	0.4221	0.4318	0.4444	0.4021	0.4512	0.4529	**0.4635**	0.5210
		Gap (%)	29.73	50.20	27.06	18.97	17.11	14.70	22.83	13.39	13.06	11.04	0.00
		Nr. Sol.	230.89	90.90	31.23	28.15	32.96	35.78	71.75	80.70	81.77	87.14	59.95
		Time (s)	543.45	627.90	2.49	2.52	3.18	1.30	18.34	19.39	21.03	9.43	—
	15×10	HV	0.4654	0.3205	0.4844	0.5583	0.5793	0.5900	0.4850	0.5782	0.6025	**0.6127**	0.7336
		Gap (%)	36.57	56.32	33.97	23.90	21.03	19.57	33.89	21.18	17.88	16.48	0.00
		Nr. Sol.	86.70	37.91	34.10	29.06	43.96	50.69	75.54	82.73	121.23	138.76	68.32
		Time (s)	1649.08	1020.61	4.36	4.52	5.51	2.73	35.56	37.33	41.00	20.03	—
	20×10	HV	0.3834	0.2547	0.5056	0.5822	0.6142	0.6169	0.5025	0.6038	**0.6314**	0.6264	0.6612
		Gap (%)	42.01	61.49	23.54	11.96	7.12	6.71	24.00	8.69	4.51	5.27	0.00
		Nr. Sol.	106.06	41.38	37.29	43.08	57.83	60.70	62.65	114.14	179.74	185.02	60.83
		Time (s)	2797.74	1674.67	7.35	7.72	9.13	4.36	64.14	67.44	74.71	35.46	—

Table 3: Four-objective SD1 scheduling: makespan, average flowtime, total earliness, and costs. Results are means over 100 instances per size, with 120 preferences and separate training at each size. Metrics and ranking follow Table 1.

		Metaheuristics		Greedy				Sampling				
Size	Metric	NSGA-II	MOEA/D	Hyper	WI-DAN	DCAN	AC-PC	Hyper	WI-DAN	DCAN	AC-PC	CP-SAT
10×5	HV	**0.5628**	0.4785	0.3871	0.3618	0.4333	0.4913	0.4306	0.4508	0.4773	0.5278	0.6531
	Gap (%)	13.82	26.74	40.73	44.60	33.65	24.78	34.07	30.97	26.92	19.18	0.00
	Nr. Sol.	1038.92	147.41	50.71	7.23	47.49	42.96	178.60	155.77	183.66	126.46	73.54
	Time (s)	255.13	267.81	0.92	0.95	1.24	0.49	5.89	5.87	6.12	3.13	—
20×5	HV	0.3381	0.2561	0.3545	0.3849	0.3895	0.4336	0.3785	0.4177	0.4147	**0.4572**	0.5068
	Gap (%)	33.28	49.47	30.05	24.05	23.15	14.44	25.32	17.58	18.17	9.78	0.00
	Nr. Sol.	695.60	111.38	61.37	51.52	51.95	56.42	209.96	186.91	165.83	177.43	64.14
	Time (s)	537.58	629.61	2.87	2.83	3.51	1.59	23.07	22.90	23.92	11.81	—
15×10	HV	0.4599	0.3283	0.4447	0.4582	0.4611	0.5903	0.4653	0.4768	0.4826	**0.6063**	0.7106
	Gap (%)	35.29	53.81	37.42	35.52	35.12	16.93	34.53	32.90	32.09	14.67	0.00
	Nr. Sol.	399.58	55.32	63.34	53.10	59.52	74.37	250.78	212.46	213.13	297.03	90.31
	Time (s)	1652.62	1034.72	5.15	5.01	6.05	2.98	43.59	43.11	45.12	25.51	—
20×10	HV	0.3828	0.2572	0.5325	0.5239	0.5500	0.6126	0.5436	0.5341	0.5660	**0.6282**	0.6485
	Gap (%)	40.97	60.34	17.89	19.22	15.19	5.54	16.18	17.65	12.73	3.13	0.00
	Nr. Sol.	185.30	45.69	66.12	52.29	55.78	62.64	256.21	168.87	181.97	212.53	79.90
	Time (s)	2758.37	1697.82	9.36	9.18	10.33	5.12	81.66	77.99	82.74	45.71	—

5.5 Effects of independent scoring and score refinement

Table 4 compares three separately trained variants with fixed composition coefficients and ten samples per preference: AC (shared) ties actor parameters across anchors; AC uses independent actors; AC-PC adds correction. The first two variants omit correction.

Independent scoring improves HV in eight of eight two-objective and three of four three-objective cases; correction improves seven of eight and three of four, respectively. Exceptions occur for independent scoring at 15×10 with three objectives, and for correction at 10×5 makespan–costs and 20×10 with three objectives.

At 20×10, independent actors use 20,547 actor parameters versus 20,709 for widened sharing, both without PC (Table 5; Appendix B). Independence improves both two-objective tasks in both modes, with sampling HV gains of 0.0025/0.0142 for makespan–costs/tardiness–costs. Three-objective differences are $+0.0007$ for greedy and -0.0033 for sampling.

Table 4: Independent scoring and score refinement with matched composition and decoding budgets. Entries are mean sampling HV over 100 instances; bold marks the best result per task and size.

Task	Variant	10×5	20×5	15×10	20×10
Makespan–costs	AC (shared)	0.7605	0.5726	0.7964	0.8191
	AC	**0.7762**	0.6019	0.8067	0.8248
	AC-PC	0.7739	**0.6025**	**0.8084**	**0.8251**
Tardiness–costs	AC (shared)	0.8207	0.6630	0.8330	0.8335
	AC	0.8321	0.6772	0.8383	0.8359
	AC-PC	**0.8370**	**0.6809**	**0.8392**	**0.8386**
Three objectives	AC (shared)	0.5211	0.4528	0.6060	0.6192
	AC	0.5474	0.4599	0.6054	**0.6315**
	AC-PC	**0.5567**	**0.4635**	**0.6127**	0.6264

Table 5: Actor-parameter-matched comparison on the 20×10 tasks. Entries are mean normalized HV over 100 instances; bold marks the higher value for each task and decoding mode. Both variants omit preference correction; ΔHV is independent minus shared.

Task	Decoding	AC (independent, 3×64)	AC (shared, 124)	ΔHV
Makespan–costs	Greedy	**0.8148**	0.8114	$+0.0034$
	Sampling	**0.8248**	0.8223	$+0.0025$
Tardiness–costs	Greedy	**0.8222**	0.8068	$+0.0154$
	Sampling	**0.8359**	0.8218	$+0.0142$
Three objectives	Greedy	**0.6165**	0.6158	$+0.0007$
	Sampling	0.6315	**0.6348**	-0.0033

5.6 Analysis of score refinement

We compare PC-on/off greedy rollouts from the same checkpoint using $J(x, w) = \sum_i w_i \alpha_i f_i(x)$, with raw objectives and the fixed scales in Appendix A.

Table 6: PC effects at 20×10 over 100 instances: gain is $J(x_{\text{off}}, w) - J(x_{\text{on}}, w)$; ΔHV is PC-on minus PC-off. Intervals use instance-paired bootstrap resampling.

Task	Full-trajectory gain [95% CI]	ΔHV
Makespan–costs	$+0.048$ $[-0.083, 0.178]$	$+0.00039$
Tardiness–costs	$+0.278$ $[0.077, 0.479]$	$+0.00322$
Three objectives	$+1.662$ $[1.540, 1.784]$	$+0.00383$

Table 6 shows positive mean objective and HV gains on all tasks; objective-gain intervals exclude zero for tardiness–costs and three objectives. Matched-state single-action gains exclude zero only for makespan–costs (Appendix E), indicating that the effect of individual corrections varies across tasks.

6 Conclusion

AC-PC composes independent actor scores and refines them for preference-conditioned scheduling. Experiments show gains over DCAN in most tested settings, including four objectives and transfer without retraining. Ablations and inference analysis support independent scoring and correction, with benefits varying by task and decoding mode. Weighted-sum training does not guarantee coverage of unsupported Pareto regions, and each decision requires multiple anchor-conditioned encoder evaluations.

AI USE STATEMENT

Writing assistance. Generative AI tools were used to polish the wording of the manuscript and assist with LaTeX formatting, including parts of the appendix. The main manuscript text was written by the authors, who reviewed and revised the AI-assisted writing.

Literature search. Generative AI tools were used to identify related work on multi-objective flexible job shop scheduling and neural combinatorial optimization. The authors are responsible for checking the selected references against their original sources and take responsibility for the final content of the paper.

ETHICS STATEMENT

This work studies multi-objective flexible job shop scheduling using synthetic instances and public scheduling benchmarks. The experiments do not involve human participants or personal data.

REPRODUCIBILITY STATEMENT

Section 4 describes the policy architecture, score composition, and preference correction. Section 5 specifies the datasets, comparison methods, and evaluation protocol. Appendix A details the network dimensions, objective scaling, reward estimates, PPO updates, and optimization settings, together with the preference grids, hypervolume normalization, hardware, and runtime measurement procedure. The core AC-PC code is provided in the Supplementary Material.

REFERENCES

Axel Abels, Diederik Roijers, Tom Lenaerts, Ann Nowé, and Denis Steckelmacher. Dynamic Weights in Multi-Objective Deep Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 11–20. PMLR, 2019. URL https://proceedings.mlr.press/v97/abels19a.html.

Paolo Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3):157–183, 1993. doi: 10.1007/bf02023073.

Rich Caruana. Multitask Learning. *Machine Learning*, 28(1):41–75, 1997. doi: 10.1023/a:1007379606734.

Jinbiao Chen, Jiahai Wang, Zizhen Zhang, Zhiguang Cao, Te Ye, and Siyuan Chen. Efficient Meta Neural Heuristic for Multi-Objective Combinatorial Optimization. In *Advances in Neural Information Processing Systems*, volume 36, pp. 56825–56837, 2023a. doi: 10.52202/075280-2482. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Jinbiao Chen, Zizhen Zhang, Zhiguang Cao, Yaoxin Wu, Yining Ma, Te Ye, and Jiahai Wang. Neural Multi-Objective Combinatorial Optimization with Diversity Enhancement. In *Advances in Neural Information Processing Systems*, volume 36, pp. 39176–39188, 2023b. doi: 10.52202/075280-1702. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/7b5ae891000049b91b3b62de596b1560-Abstract-Conference.html.

Jinbiao Chen, Zhiguang Cao, Jiahai Wang, Yaoxin Wu, Hanzhang Qin, Zizhen Zhang, and Yue-Jiao Gong. Rethinking Neural Multi-Objective Combinatorial Optimization via Neat Weight Embedding. In *International Conference on Learning Representations*, pp. 48510–48529, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/78efbc5386c5a7c241e7fcc482d3c3dc-Abstract-Conference.html.

Andrea Corsini, Angelo Porrello, Simone Calderara, and Mauro Dell'Amico. Self-Labeling the Job Shop Scheduling Problem. In *Advances in Neural Information Processing Systems*, volume 37, pp. 105528–105551, 2024. doi: 10.52202/079017-3349. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/be8987a75afeeadfa65007cf7ee25de0-Abstract-Conference.html.

Stéphane Dauzère-Pérès, Junwen Ding, Liji Shen, and Karim Tamssaouet. The flexible job shop scheduling problem: A review. *European Journal of Operational Research*, 314(2):409–432, 2024. doi: 10.1016/j.ejor.2023.05.017.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002. doi: 10.1109/4235.996017.

Matthias Ehrgott. *Multicriteria Optimization*. Springer, 2nd edition, 2005. doi: 10.1007/3-540-27659-9.

Mingfeng Fan, Yaoxin Wu, Zhiguang Cao, Wen Song, Guillaume Sartoretti, Huan Liu, and Guohua Wu. Conditional Neural Heuristic for Multiobjective Vehicle Routing Problems. *IEEE Transactions on Neural Networks and Learning Systems*, 36(3):4677–4689, 2025a. doi: 10.1109/tnnls.2024.3371706.

Mingfeng Fan, Jianan Zhou, Yifeng Zhang, Yaoxin Wu, Jinbiao Chen, and Guillaume Sartoretti. Preference-Driven Multi-Objective Combinatorial Optimization with Conditional Computation. In *Advances in Neural Information Processing Systems*, volume 38, pp. 11594–11627, 2025b. doi: 10.52202/085713-0389. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/10fc83943b4540a9524af6fc67a23fef-Abstract-Conference.html.

Gerald Farin. *Curves and Surfaces for CAGD: A Practical Guide*. Morgan Kaufmann, 5th edition, 2002. ISBN 9781558607378. URL https://www.farinhansford.com/books/cagd/toc.html.

Xiaoyu Fu, Shenshen Gu, and Chee-Meng Chew. Optimizing the multi-objective traveling salesman problem with a deep reinforcement learning algorithm using cross fusion attention networks. *Neural Networks*, 192:107904, 2025. doi: 10.1016/j.neunet.2025.107904.

Andreia P. Guerreiro, Carlos M. Fonseca, and Luís Paquete. The Hypervolume Indicator: Computational Problems and Algorithms. *ACM Computing Surveys*, 54(6):119:1–119:42, 2022. doi: 10.1145/3453474.

Conor F. Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M. Zintgraf, Richard Dazeley, Fredrik Heintz, Enda Howley, Athirai A. Irissappane, Patrick Mannion, Ann Nowé, Gabriel Ramos, Marcello Restelli, Peter Vamplew, and Diederik M. Roijers. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, 2022. doi: 10.1007/s10458-022-09552-y.

Shengyi Huang and Santiago Ontañón. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. *Proceedings of the International FLAIRS Conference*, 35, 2022. doi: 10.32473/flairs.v35i.130584.

Johann Hurink, Bernd Jurisch, and Monika Thole. Tabu search for the job-shop scheduling problem with multi-purpose machines. *OR Spektrum*, 15(4):205–215, 1994. doi: 10.1007/bf01719451.

Kun Lei, Peng Guo, Wenchao Zhao, Yi Wang, Linmao Qian, Xiangyin Meng, and Liansheng Tang. A multi-action deep reinforcement learning framework for flexible Job-shop scheduling problem. *Expert Systems with Applications*, 205:117796, 2022. doi: 10.1016/j.eswa.2022.117796.

Kaiwen Li, Tao Zhang, and Rui Wang. Deep Reinforcement Learning for Multiobjective Optimization. *IEEE Transactions on Cybernetics*, 51(6):3103–3114, 2021. doi: 10.1109/tcyb.2020.2977661.

Xi Lin, Zhiyuan Yang, and Qingfu Zhang. Pareto set learning for neural multi-objective combinatorial optimization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=QuObT9BTWo.

Wei Liu, Yaoxin Wu, Yingqian Zhang, Thomas Bäck, and Yingjie Fan. Adversarial instance generation and robust training for neural combinatorial optimization with multiple objectives. *arXiv preprint arXiv:2601.01665*, 2026. URL https://arxiv.org/abs/2601.01665.

Yongfan Lu, Zixiang Di, Bingdong Li, Shengcai Liu, Hong Qian, Peng Yang, Ke Tang, and Aimin Zhou. Context-aware diversity enhancement for neural multi-objective combinatorial optimization. *arXiv preprint arXiv:2405.08604v3*, 2025. URL https://arxiv.org/abs/2405.08604v3.

Shu Luo, Linxuan Zhang, and Yushun Fan. Dynamic multi-objective scheduling for flexible job shop by deep reinforcement learning. *Computers & Industrial Engineering*, 159:107489, 2021. doi: 10.1016/j.cie.2021.107489.

R. Timothy Marler and Jasbir S. Arora. The weighted sum method for multi-objective optimization: new insights. *Structural and Multidisciplinary Optimization*, 41(6):853–862, 2010. doi: 10.1007/s00158-009-0460-7.

Ghasem Moslehi and Mehdi Mahnam. A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, 129(1):14–22, 2011. doi: 10.1016/j.ijpe.2010.08.004.

Junyoung Park, Jaehyeong Chun, Sang Hun Kim, Youngkook Kim, and Jinkyoo Park. Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning. *International Journal of Production Research*, 59(11):3360–3377, 2021. doi: 10.1080/00207543.2020.1870013.

Laurent Perron, Frédéric Didier, and Steven Gay. The CP-SAT-LP Solver (Invited Talk). In *29th International Conference on Principles and Practice of Constraint Programming*, volume 280 of *Leibniz International Proceedings in Informatics*, pp. 3:1–3:2, 2023. doi: 10.4230/LIPIcs.CP.2023.3.

D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley. A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013. doi: 10.1613/jair.3987.

Sebastian Ruder. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv preprint arXiv:1706.05098*, 2017. URL https://arxiv.org/abs/1706.05098.

Rubén Ruiz and José Antonio Vázquez-Rodríguez. The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205(1):1–18, 2010. doi: 10.1016/j.ejor.2009.09.024.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://arxiv.org/abs/1707.06347.

Igor G. Smit, Jianan Zhou, Robbert Reijnen, Yaoxin Wu, Jian Chen, Cong Zhang, Zaharah Bukhsh, Yingqian Zhang, and Wim Nuijten. Graph neural networks for job shop scheduling problems: A survey. *Computers & Operations Research*, 176:106914, 2025. doi: 10.1016/j.cor.2024.106914.

Igor G. Smit, Yaoxin Wu, Pavel Troubil, Yingqian Zhang, and Wim P. M. Nuijten. Neural multi-objective combinatorial optimization for flexible job shop scheduling problems. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=YAgOaYedLQ.

Wen Song, Xinyang Chen, Qiqiang Li, and Zhiguang Cao. Flexible Job-Shop Scheduling via Graph Neural Network and Deep Reinforcement Learning. *IEEE Transactions on Industrial Informatics*, 19(2):1600–1610, 2023. doi: 10.1109/tii.2022.3189725.

Chupeng Su, Cong Zhang, Chuang Wang, Weihong Cen, Gang Chen, and Longhan Xie. Fast Pareto set approximation for multi-objective flexible job shop scheduling via parallel preference-conditioned graph reinforcement learning. *Swarm and Evolutionary Computation*, 88:101605, 2024. doi: 10.1016/j.swevo.2024.101605.

Karim Tamssaouet, Stéphane Dauzère-Pérès, Sebastian Knopp, Abdoul Bitar, and Claude Yugma. Multiobjective optimization for complex flexible job-shop scheduling problems. *European Journal of Operational Research*, 296(1):87–100, 2022. doi: 10.1016/j.ejor.2021.03.069.

Liting Wang, Chao Song, Yu Sun, Cuihua Lu, and Qinghua Chen. A Neural Multi-Objective Capacitated Vehicle Routing Optimization Algorithm Based on Preference Adjustment. *Electronics*, 12 (19):4167, 2023. doi: 10.3390/electronics12194167.

Runqing Wang, Gang Wang, Jian Sun, Fang Deng, and Jie Chen. Flexible Job Shop Scheduling via Dual Attention Network-Based Reinforcement Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 35(3):3091–3102, 2024a. doi: 10.1109/tnnls.2023.3306421.

Zhenkun Wang, Shunyu Yao, Genghui Li, and Qingfu Zhang. Multiobjective Combinatorial Optimization Using a Single Deep Reinforcement Learning Model. *IEEE Transactions on Cybernetics*, 54(3):1984–1996, 2024b. doi: 10.1109/tcyb.2023.3312476.

Xuan Wu, Jinbiao Chen, Yang Li, Lijie Wen, Chunguo Wu, Yuanshu Li, Yubin Xiao, Chunyan Miao, You Zhou, and Di Wang. WeCon: An efficient weight-conditioned neural solver for multi-objective combinatorial optimization problems. *arXiv preprint arXiv:2605.22876*, 2026. URL https://arxiv.org/abs/2605.22876.

Biao Xiao, Zhengcai Zhao, Yingchen Wu, Xialin Zhu, Shixin Peng, and Honghua Su. An improved MOEA/D for multi-objective flexible job shop scheduling by considering efficiency and cost. *Computers & Operations Research*, 167:106674, 2024. doi: 10.1016/j.cor.2024.106674.

Runzhe Yang, Xingyuan Sun, and Karthik Narasimhan. A Generalized Algorithm for Multi-Objective Reinforcement Learning and Policy Adaptation. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/4a46fbfca3f1465a27b210f4bdfe6ab3-Abstract.html.

Te Ye, Zizhen Zhang, Jinbiao Chen, and Jiahai Wang. Weight-Specific-Decoder Attention Model to Solve Multiobjective Combinatorial Optimization Problems. In *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2839–2844, 2022. doi: 10.1109/smc53654.2022.9945568.

Te Ye, Zizhen Zhang, Qingfu Zhang, Jinbiao Chen, and Jiahai Wang. Solving Multiobjective Combinatorial Optimization via Learning to Improve Method. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 9(3):2122–2136, 2025. doi: 10.1109/tetci.2025.3540424.

Cong Zhang, Wen Song, Zhiguang Cao, Jie Zhang, Puay Siew Tan, and Chi Xu. Learning to Dispatch for Job Shop Scheduling via Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1621–1632, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/11958dfee29b6709f48a9ba0387a2431-Abstract.html.

Haotian Zhang, Jialong Shi, Jianyong Sun, Qingfu Zhang, and Zongben Xu. Neuro-PLS: A Generalizable Local Search Framework for Multiobjective Combinatorial Optimization. *IEEE Transactions on Evolutionary Computation*, 30(4):1564–1578, 2026. doi: 10.1109/tevc.2025.3589640.

Qingfu Zhang and Hui Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007. doi: 10.1109/tevc.2007.892759.

Yongxin Zhang, Jiahai Wang, Zizhen Zhang, and Yalan Zhou. MODRL/D-EL: Multiobjective Deep Reinforcement Learning with Evolutionary Learning for Multiobjective Optimization. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021. doi: 10.1109/ijcnn52387.2021.9534083.

Zhi Zheng, Shunyu Yao, Genghui Li, Linxi Han, and Zhenkun Wang. Pareto Improver: Learning Improvement Heuristics for Multi-Objective Route Planning. *IEEE Transactions on Intelligent Transportation Systems*, 25(1):1033–1043, 2024. doi: 10.1109/tits.2023.3313688.

A IMPLEMENTATION AND EVALUATION DETAILS

Actor features and score refinement. The encoder has two operation/machine attention layers, with four heads per block and output dimensions 32 and 8. The actor input concatenates four eight-dimensional embeddings and eight pair features, giving dimension 40. Each of the three actors has its own parameters, two hidden layers of width 64, and a scalar action-score output. All AC variants and extensions use actors without residual connections. Equation 6 combines their final hidden representations using the same composition coefficients as the base logits.

For two objectives, preference correction takes 64 mixed hidden features and two preference components. Its $66 \to 64 \to 1$ network contains 4,353 parameters, including biases, and uses the gate $g_2(w) = 4w_1w_2$. For three objectives, the corresponding $67 \to 64 \to 1$ network contains 4,417 parameters and uses $g_3(w) = \frac{3}{2}(1 - \sum_i w_i^2)$. Both networks use a tanh hidden activation. Only the final linear layer is initialized to zero.

Before mixing, the implementation subtracts each branch's mean logit over feasible actions. The gated correction is also centered over feasible actions before addition. Each subtraction changes all feasible logits by the same scalar, leaving the masked softmax unchanged. Equations 5–9 omit these constants.

Objective scaling. The environment first normalizes processing times and operation costs. Let p^-, p^+ be the minimum and maximum entries of the processing-time array for the parallel environment batch, and let c^-, c^+ be the corresponding extrema of the cost array. With raw operation cost $c_{ij}^k = p_{\max} - p_{ij}^k$, the normalized quantities are

$$\widehat{p}_{ij}^k = \frac{p_{ij}^k - p^-}{p^+ - p^- + \varepsilon}, \qquad \widehat{c}_{ij}^k = \frac{c_{ij}^k - c^-}{c^+ - c^- + \varepsilon}, \qquad \varepsilon = 10^{-8}. \tag{10}$$

These extrema are taken over the dense arrays before masking incompatible machine entries, whose raw processing times are zero. The normalized scheduling timeline and due dates use $\widehat{p}$, while cost estimates use $\widehat{c}$. The resulting objective values are f_i^{norm}; fixed multipliers then give $\widetilde{f}_i = \alpha_i f_i^{\mathrm{norm}}$ and scale the corresponding reward differences.

For the synthetic training instances, the fixed multiplier tuples $(\alpha_{\mathrm{mk}}, \alpha_{\mathrm{td}}, \alpha_{\mathrm{fl}}, \alpha_{\mathrm{co}})$ are

$$\begin{aligned} 10 \times 5 &: (1,\ 3.2/11,\ 3.2/3,\ 3.2/2.5), \\ 20 \times 5 &: (1,\ 8.5/80,\ 8.5/4.5,\ 8.5/4.8), \\ 15 \times 10 &: (1,\ 4/11,\ 4/4.5,\ 4/10), \\ 20 \times 10 &: (1,\ 7/45,\ 7/6,\ 7/13.5). \end{aligned} \tag{11}$$

Each task selects the components for its objectives. Scale transfer to 30×10 and 40×10 retains the 20×10 multipliers; public-benchmark transfer uses the 15×10 multipliers. These training scales are distinct from the hypervolume normalization used to evaluate returned schedules in the original objective units.

Partial-schedule estimates. All quantities in this paragraph use normalized processing times and costs. Let q_i be the number of assigned operations of job i, C_{ij} their assigned completion times, and $\underline{p}_{ij} = \min_{k \in \mathcal{M}_{ij}} \widehat{p}_{ij}^k$. With $C_{i0} = 0$, the environment's operation-completion estimate is

$$L_{ij}(s) = \begin{cases} C_{ij}, & j \le q_i, \\ C_{iq_i} + \sum_{u=q_i+1}^{j} \underline{p}_{iu}, & j > q_i. \end{cases} \tag{12}$$

Assigning an operation replaces its estimate with its completion time and shifts the estimates of its successors by the same difference. Write $L_i = L_{in_i}$, let $\widehat{D}_i$ be the normalized due date, and let $\widehat{S}_i$ be the first operation's assigned start. After each assignment, the objective estimates are

$$\widehat{f}_{\mathrm{mk}}^{\mathrm{norm}}(s) = \max_{i,j} L_{ij}(s), \qquad \widehat{f}_{\mathrm{td}}^{\mathrm{norm}}(s) = \sum_i \max\{L_i - \widehat{D}_i, 0\}, \tag{13}$$

$$\widehat{f}_{\mathrm{ear}}^{\mathrm{norm}}(s) = \sum_i \max\{\widehat{D}_i - L_i, 0\}, \qquad \widehat{f}_{\mathrm{fl}}^{\mathrm{norm}}(s) = \frac{1}{|\mathcal{J}|} \sum_i F_i(s), \tag{14}$$

where $F_i(s) = \sum_{j=1}^{n_i} \underline{p}_{ij}$ for an unstarted job and $F_i(s) = \max_j L_{ij}(s) - \widehat{S}_i$ otherwise. Let k_{ij} denote the machine assigned to operation O_{ij}. The cost estimate adds assigned costs and minimum eligible costs for unassigned operations:

$$\widehat{f}_{\mathrm{co}}^{\,\mathrm{norm}}(s) = \sum_i \left(\sum_{j \le q_i} \widehat{c}_{ij}^{\,k_{ij}} + \sum_{j > q_i} \min_{k \in \mathcal{M}_{ij}} \widehat{c}_{ij}^{\,k} \right). \tag{15}$$

The initial estimates use the same expressions except for flowtime, which is initialized as the mean of $L_{ij}(s_0)$ over all operations, following the benchmark implementation. Rewards are the scaled decreases in these estimates, as defined in Section 4.5.

Critic. For each anchor, the encoder returns global operation and machine embeddings. We average these across anchors using $b_k(w)$, concatenate the two averages, and pass them through a three-layer critic with hidden width 64. Its output has dimension two or three, matching the task's objective count.

PPO objective. For a rollout of length T, let $r_t = (r_{t,1}, \ldots, r_{t,d})$, $V_t = V_{\mathrm{old}}(s_t, w) \in \mathbb{R}^d$, and $V_T = 0$. With discount factor $\gamma = 1$ and GAE parameter $\lambda = 0.98$, vector temporal-difference residuals and generalized advantage estimates (Schulman et al., 2016) are

$$\delta_t = r_t + \gamma V_{t+1} - V_t, \qquad B_t = \sum_{l=0}^{T-1-t} (\gamma\lambda)^l \delta_{t+l}. \tag{16}$$

The actor uses $A_t = w^\top B_t$, normalized by its mean and sample standard deviation within each trajectory: $\widehat{A}_t = (A_t - \overline{A})/(s_A + 10^{-8})$. The vector critic target is $G_t = V_t + B_t$, computed before advantage normalization and held fixed during the PPO updates. The clipped policy objective, with threshold $\epsilon = 0.2$, is

$$L_{\mathrm{clip}} = \mathbb{E}_t\Big[\min\Big(\rho_t \widehat{A}_t, \mathrm{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\widehat{A}_t\Big)\Big], \quad \rho_t = \frac{\pi(a_t \mid s_t, w)}{\pi_{\mathrm{old}}(a_t \mid s_t, w)}. \tag{17}$$

Here π_{old} generated the rollout, and π is the final masked policy after composition and correction. With $V_i(s_t, w)$ denoting the current critic's ith output, the critic loss and policy entropy are

$$L_V = \mathbb{E}_t\left[\frac{1}{d}\sum_{i=1}^{d}\big(V_i(s_t, w) - G_{t,i}\big)^2\right], \tag{18}$$

$$\mathcal{H} = \mathbb{E}_t\left[-\sum_{a \in \mathcal{A}(s_t)} \pi(a \mid s_t, w) \log \pi(a \mid s_t, w)\right]. \tag{19}$$

We minimize $-L_{\mathrm{clip}} + 0.5 L_V - 0.01\mathcal{H}$, averaging over the minibatch.

Computation per decision. Let Q be the number of candidate operation–machine pairs before masking and H the actor hidden width. With K anchors, the policy requires K encoder evaluations and K actor evaluations, giving work $K(C_E + C_A) + C_{\mathrm{PC}} + O(KQH)$ for encoder cost C_E, actor cost C_A, and correction cost C_{PC}. The implementation batches the anchors along the environment dimension and applies each actor to its branch features. Branch activations scale linearly with K at fixed instance size; encoder parameters are shared. Here $K = 3$ for two and three objectives and $K = 4$ for four objectives.

Optimization settings. The configuration uses a learning rate of 3×10^{-4}, 1,500 training updates, 20 parallel environments, instance refresh every 20 updates, and validation every 40 updates. PPO uses four optimization passes, a minibatch size of 1,024, clipping threshold 0.2, discount factor 1, advantage-estimation parameter 0.98, value-loss coefficient 0.5, and entropy coefficient 0.01. Dropout is zero. We use these settings for all training runs.

Preference grids and returned sets. Two-objective testing uses $\{(i/100, 1 - i/100) : i = 0, \ldots, 100\}$. Three-objective testing uses $\{(i, j, k)/13 : i, j, k \in \mathbb{Z}_{\geq 0},\ i + j + k = 13\}$, containing $\binom{15}{2} = 105$ preferences, except when varying the grid in Table 7. A greedy evaluation constructs one schedule per preference. A sampling evaluation constructs ten per preference, for 1,010 or 1,050 schedules on the default grids before deduplication and Pareto filtering. Greedy and sampling are separate evaluations: the sampled pool contains only sampled schedules. The reported set size counts distinct nondominated objective vectors.

Hypervolume normalization. For a returned nondominated set P, let ℓ be the instance's objective lower-bound vector and r its benchmark reference point. The implementation reports

$$\mathrm{HV}_{\mathrm{norm}}(P) = \frac{\mathrm{HV}(P - \ell;\ r - \ell + \varepsilon \mathbf{1})}{\prod_{i=1}^{d}(r_i - \ell_i + \varepsilon)}, \qquad \varepsilon = 10^{-8}. \tag{20}$$

Here $\mathbf{1} \in \mathbb{R}^d$ is the all-ones vector. We report arithmetic means of the per-instance normalized hypervolumes. If objective j is constant at its lower bound ℓ_j, its factor $r_j - \ell_j + \varepsilon$ cancels between the hypervolume and the denominator. The normalized HV then equals that obtained after removing dimension j, including when $r_j = \ell_j$.

Gap and reporting conventions. AC-PC's Gap is $100(\overline{\mathrm{HV}}_{\mathrm{CP}} - \overline{\mathrm{HV}})/\overline{\mathrm{HV}}_{\mathrm{CP}}$, where $\overline{\mathrm{HV}}_{\mathrm{CP}}$ is the mean HV of the time-limited CP-SAT reference; baseline gaps retain their reported values. A negative gap indicates higher HV than this reference. HV is reported to four decimal places; Gap, distinct nondominated-vector count (Nr. Sol.), and runtime use two.

Hardware and runtime measurement. All AC-PC experiments use a server with an Intel Core Ultra 9 285K CPU and one NVIDIA GeForce RTX 5090 GPU (32 GB). The software environment uses Python 3.11.14, PyTorch 2.11.0.dev20260108, and CUDA 12.8. The AC ablations use the same platform. Runtime measurements use the GPU exclusively, with no other GPU workloads running concurrently.

Inference processes one test instance at a time, batching its preferences and sampled trajectories. Reported time is the mean wall-clock duration of the scheduling loop per instance, including policy evaluation, action selection, and environment updates. It excludes model loading, instance initialization, Pareto filtering, hypervolume computation, and file output. Standard greedy and sampling evaluations batch all preferences, with one or ten trajectories per preference, respectively. Larger sampling budgets use batches of ten trajectories per preference and sum the scheduling times across batches (Appendix J). Runtimes taken from the literature or measured for evolutionary reruns use their respective platforms and do not provide a controlled comparison of computational efficiency with AC-PC.

B ACTOR-PARAMETER-MATCHED COMPARISON

We compare the three independent width-64 actors in AC with one shared actor widened to 124 units and tied across the same three anchors. Both variants omit preference correction and use the same training and evaluation protocol. With a 40-dimensional input, two hidden layers of width h, and a scalar output, an actor contains $P(h) = h^2 + 43h + 1$ parameters. The independent actors contain $3P(64) = 20{,}547$ actor parameters in total, while the widened shared actor contains $P(124) = 20{,}709$, or 0.79% more.

Table 5 in the main text reports the comparison results.

C INFERENCE PREFERENCE COUNT

For the 10×5 and 20×10 three-objective checkpoints, we retain the 100 test instances per size and HV references from Table 2, varying only the inference grid. The grids are $\{(i, j, k)/m : i, j, k \in \mathbb{Z}_{\geq 0},\ i + j + k = m\}$, with integer resolution $m \in \{3, 8, 13, 30, 44\}$ giving $N \in \{10, 45, 105, 496, 1035\}$ preferences. Both decoding modes use $N \leq 105$; larger grids use greedy decoding. Table 7 reuses four $N = 105$ results from Table 2 and adds 12 evaluations.

Table 7: AC-PC results for three objectives with varying inference preference counts, averaged over 100 instances per size. Greedy constructs one schedule per preference and Sample constructs ten. Gap to CP-SAT, nondominated-vector count (Nr. Sol.), runtime, and reporting precision follow Table 1.

		$N = 10$		$N = 45$		$N = 105$		$N = 496$	$N = 1035$
Size	Metric	Greedy	Sample	Greedy	Sample	Greedy	Sample	Greedy	Greedy
10×5	HV	0.4680	0.5181	0.5026	0.5448	0.5163	0.5567	0.5296	0.5328
	Gap (%)	31.48	24.15	26.42	20.25	24.42	18.50	22.47	22.01
	Nr. Sol.	8.43	33.23	22.65	62.93	33.45	83.00	50.88	57.79
	Time (s)	0.29	0.46	0.33	1.19	0.47	2.55	1.29	2.50
20×10	HV	0.5789	0.5959	0.6089	0.6197	0.6169	0.6264	0.6301	0.6344
	Gap (%)	12.44	9.87	7.92	6.28	6.71	5.27	4.70	4.05
	Nr. Sol.	9.31	48.72	33.67	125.78	60.70	185.02	146.23	198.28
	Time (s)	1.28	4.05	2.16	15.66	4.36	35.46	17.05	34.52

Greedy HV increases with N at both sizes, with diminishing gains: increasing N from 496 to 1,035 adds 0.0032/0.0043 HV while roughly doubling runtime. At similar rollout budgets, $N = 105$ sampling outperforms $N = 1{,}035$ greedy at 10×5, whereas greedy is stronger at 20×10.

D Additional details for the four-objective extension

We evaluate four-objective SD1 FJSP at 10×5, 20×5, 15×10, and 20×10. The objective vector contains makespan, average flowtime, total earliness, and costs. Total earliness is $f_{\text{ear}} = \sum_i \max(D_i - C_i, 0)$, with the same due dates D_i defined in Section 3. The benchmark reward uses an earliness upper estimate obtained from job-completion lower bounds, while evaluation uses the completed schedules.

We extend AC-PC to four objectives using four independent actors at the four unit-vector anchors. Their scores and hidden features are combined with weights $w_1, \ldots, w_4$, extending the sums in Equations 5 and 6 to four terms. The encoder is shared across anchors, and the correction uses the four-dimensional preference with $g_4(\mathbf{w}) = \frac{4}{3}(1 - \sum_{i=1}^{4} w_i^2)$. Its $68 \to 64 \to 1$ MLP adds 4,481 parameters and retains the zero-initialized output layer. The critic has four objective outputs.

Each size is trained from scratch with the optimization settings in Appendix A. Validation uses 100 instances and 56 Das–Dennis preferences ($m = 5$), with checkpoint selection by validation HV. The earliness multipliers are $1/4.5$, $1/4$, $1/8$, and $1/7$ in the size order above; the other three objective multipliers follow Appendix A.

Testing uses 100 instances per size and 120 Das–Dennis preferences ($m = 7$), with the greedy and sampling protocols in Appendix A. Table 3 uses the benchmark's four-dimensional normalization bounds and reference points.

E Additional score refinement analysis

The preference maps use 100 fixed instances for each 20×10 task, with 201 evenly spaced two-objective preferences or the complete $m = 20$ three-objective grid of 231 preferences. PC-on and PC-off enable or disable correction in the same AC-PC checkpoint and use greedy decoding. We evaluate each schedule with $J(x, w) = \sum_{i=1}^{d} w_i \alpha_i f_i(x)$, using objectives in their original units and the multipliers in Appendix A. Full-trajectory gain is $J(x_{\text{off}}, w) - J(x_{\text{on}}, w)$, where x_{off} and x_{on} are the resulting schedules.

The branch analysis uses 20 fixed instances, five preferences, and states from PC-on and PC-off trajectories at approximately 20%, 50%, and 80% of construction, giving 600 states per task. Two-objective first weights are $0.1, 0.3, 0.5, 0.7, 0.9$. Three-objective preferences are $(1/3, 1/3, 1/3)$, the three permutations of $(0.8, 0.1, 0.1)$, and $(0.45, 0.45, 0.1)$. At each state, one branch takes the base action and the other the corrected action; both then follow the full AC-PC policy with greedy

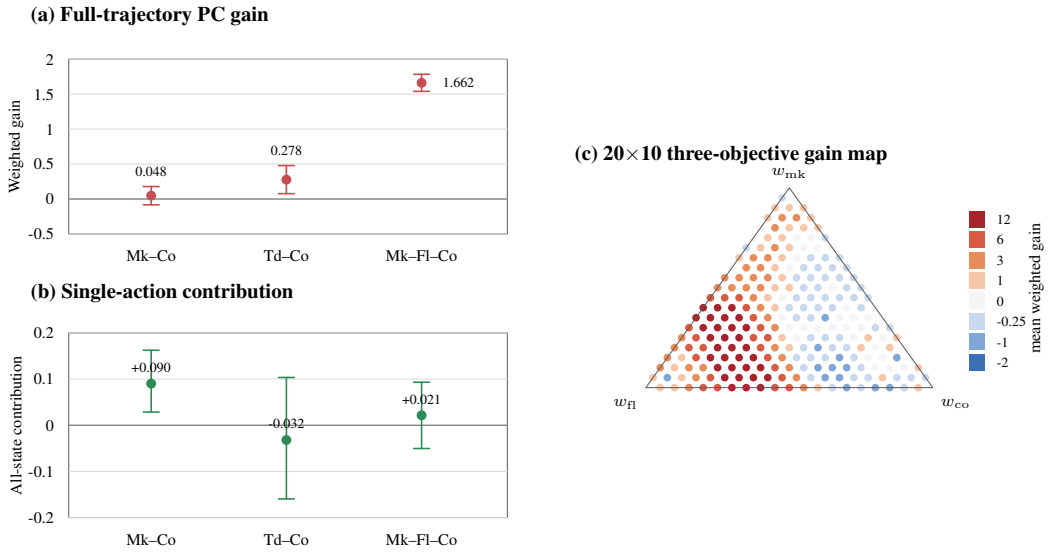


Figure 2: Score refinement on the three 20×10 tasks. (a) Mean full-trajectory gain $J(x_{\mathrm{off}}, w) - J(x_{\mathrm{on}}, w)$. (b) Mean contribution of one corrected action, with both branches subsequently following the same AC-PC policy. Positive values favor correction; error bars in (a) and (b) are instance-paired bootstrap 95% confidence intervals. (c) Mean full-trajectory gain over the three-objective preference simplex.

decoding. Branch gain is the first rollout's J minus the second's, with zero assigned when the actions match.

Table 8: Score refinement results on the three 20×10 tasks. Full-trajectory gain is the mean difference $J(x_{\mathrm{off}}, w) - J(x_{\mathrm{on}}, w)$ for the same checkpoint; positive values favor correction. ΔHV is PC-on minus PC-off. Single-action contribution averages the branch gain over sampled states, assigning zero when PC retains the base action. Intervals use instance-paired bootstrap resampling.

Size	Objectives	Full-trajectory gain [95% CI]	ΔHV	Single-action contribution [95% CI]
20×10	Mk–Co	$+0.048$ $[-0.083, 0.178]$	$+0.00039$	$+0.090$ $[0.029, 0.162]$
20×10	Td–Co	$+0.278$ $[0.077, 0.479]$	$+0.00322$	-0.032 $[-0.160, 0.103]$
20×10	Mk–Fl–Co	$+1.662$ $[1.540, 1.784]$	$+0.00383$	$+0.021$ $[-0.050, 0.093]$

Table 8 reports both analyses. Over full trajectories, enabling correction lowers the mean preference-weighted objective and increases HV on all three tasks. The objective-gain confidence intervals are above zero for tardiness–costs and three objectives, but cross zero for makespan–costs. Figure 2(c) also shows that gains vary with preference, including negative gains near the cost vertex.

For the single-action analysis (Figure 2(b)), only makespan–costs has a gain interval above zero. The mean contribution is negative for tardiness–costs and positive for three objectives, but both intervals cross zero.

F TRANSFER TO LARGER SYNTHETIC INSTANCES

Evaluation protocol. We transfer the 20×10 checkpoints to 30×10 and 40×10 synthetic instances without retraining, using 100 test instances per task and size and the evaluation protocol in Appendix A.

Table 9: Transfer from 20×10 to larger synthetic instances without retraining, with 100 instances per task and size. Metrics and ranking follow Table 1.

		Metaheuristics		Greedy				Sampling				
Size	Metric	NSGA-II	MOEA/D	Hyper	WI-DAN	DCAN	AC-PC	Hyper	WI-DAN	DCAN	AC-PC	CP-SAT
Makespan–costs												
30×10	HV	0.5477	0.3848	0.7221	0.7229	0.7437	0.7535	0.6584	0.7294	0.7493	**0.7560**	0.7425
	Gap (%)	26.24	48.18	2.74	2.64	−0.17	−1.48	11.33	1.77	−0.91	−1.81	0.00
	Nr. Sol.	29.73	27.80	24.08	11.67	19.46	23.11	7.49	18.14	32.62	39.66	11.61
	Time (s)	5674.45	3580.27	9.54	9.63	12.09	6.64	82.49	86.64	100.90	57.33	—
40×10	HV	0.4832	0.3128	0.6436	0.6387	0.6629	0.6746	0.5852	0.6469	0.6688	**0.6750**	0.6326
	Gap (%)	23.61	50.54	−1.74	−0.97	−4.79	−6.64	7.48	−2.28	−5.73	−6.71	0.00
	Nr. Sol.	34.17	31.28	28.80	12.57	22.31	26.90	7.61	22.91	39.86	49.32	11.23
	Time (s)	10013.26	6233.48	15.69	16.26	20.48	11.27	148.92	156.98	181.16	102.24	—
Tardiness–costs												
30×10	HV	0.4652	0.3082	0.6676	0.7627	0.7464	0.7587	0.6562	**0.7694**	0.7601	0.7591	0.6872
	Gap (%)	32.31	55.16	2.85	−10.98	−8.60	−10.40	4.51	−11.95	−10.61	−10.46	0.00
	Nr. Sol.	33.40	28.54	19.07	25.57	24.53	31.40	15.53	41.03	41.43	42.46	10.18
	Time (s)	5664.65	3454.61	10.90	11.58	14.03	7.24	98.26	105.23	119.65	63.85	—
40×10	HV	0.3751	0.2349	0.5708	0.7040	0.6649	0.6890	0.5659	**0.7084**	0.6815	0.6848	0.5390
	Gap (%)	30.41	56.41	−5.89	−30.61	−23.35	−27.83	−4.99	−31.42	−26.43	−27.05	0.00
	Nr. Sol.	40.00	28.81	15.41	29.35	25.25	34.88	14.97	49.78	41.45	46.20	7.71
	Time (s)	10020.31	5826.27	19.93	19.68	24.12	12.42	188.55	194.54	220.47	116.26	—
Makespan–flowtime–costs												
30×10	HV	0.2789	0.1709	0.4640	0.5401	0.5707	0.5776	0.4382	0.5563	**0.5811**	0.5800	0.4766
	Gap (%)	41.49	64.13	2.64	−13.33	−19.76	−21.19	8.06	−16.74	−21.93	−21.69	0.00
	Nr. Sol.	137.67	45.18	44.62	46.62	60.85	69.08	50.15	131.78	205.99	217.28	46.04
	Time (s)	5820.54	3306.71	18.42	16.08	18.58	8.97	180.95	150.65	165.46	90.96	—
40×10	HV	0.2022	0.1165	0.3993	0.4773	0.5087	0.5192	0.3724	0.4918	0.5194	**0.5215**	0.3043
	Gap (%)	33.56	61.73	−31.22	−56.84	−67.17	−70.62	−22.39	−61.63	−70.70	−71.38	0.00
	Nr. Sol.	66.59	44.26	49.97	51.09	63.53	71.02	65.19	151.06	224.75	237.32	35.81
	Time (s)	10028.12	5705.95	26.53	26.80	30.87	16.36	263.06	275.59	298.31	170.26	—

Results. AC-PC exceeds DCAN HV in 10 of 12 comparisons (Table 9): seven of eight for two objectives and three of four for three. Its advantages therefore largely persist at the larger sizes, although WI-DAN achieves higher HV in all four tardiness–costs comparisons.

Sampling does not always improve transfer performance. At 40×10 tardiness–costs, it lowers AC-PC HV from 0.6890 to 0.6848 despite returning more nondominated vectors. Greedy and sampling generate separate sets, so a larger sampled set can still have lower HV.

G THREE-OBJECTIVE SD2 SYNTHETIC INSTANCES

Dataset and protocol. SD2 samples eligible processing times independently from $U(1, 99)$, producing sharper differences among machine alternatives than SD1's correlated times. We evaluate makespan, average flowtime, and costs at the four sizes in Table 2.

A separate three-objective model is trained from scratch at each size and tested on 100 instances, following the architecture, optimization, and evaluation settings in Appendix A.

Table 10: Three-objective SD2 results for makespan, average flowtime, and costs. Results are means over 100 instances per size, with 105 preferences and separate training at each size. Sampling constructs ten schedules per preference. Metrics and ranking follow Table 1.

		Metaheuristics		Greedy				Sampling				
Size	Metric	NSGA-II	MOEA/D	Hyper	WI-DAN	DCAN	AC-PC	Hyper	WI-DAN	DCAN	AC-PC	CP-SAT
10×5	HV	0.4531	0.3789	0.3868	0.3594	0.3918	0.4141	0.4092	0.4123	0.4340	**0.4543**	0.5158
	Gap (%)	12.16	26.55	25.01	30.32	24.05	19.71	20.66	20.07	15.87	11.92	0.00
	Nr. Sol.	479.91	193.70	51.90	23.75	49.21	52.37	137.85	150.34	183.25	211.54	75.62
	Time (s)	248.22	260.93	0.83	0.82	1.11	0.46	5.05	5.03	5.65	2.52	—
20×5	HV	0.2759	0.2212	0.3416	0.3383	0.3608	0.3603	0.3428	0.3701	**0.3827**	0.3823	0.3884
	Gap (%)	28.96	43.06	12.04	12.91	7.10	7.24	11.74	4.70	1.46	1.56	0.00
	Nr. Sol.	692.81	228.96	62.85	45.76	64.60	64.80	182.79	220.66	242.45	240.11	87.55
	Time (s)	565.30	626.18	2.51	2.60	3.20	1.22	18.76	19.60	21.41	9.42	—
15×10	HV	0.2831	0.1992	0.4024	0.4171	0.4174	0.4233	0.4217	0.4369	0.4354	**0.4428**	0.4705
	Gap (%)	39.84	57.67	14.47	11.37	11.29	10.04	10.37	7.15	7.47	5.89	0.00
	Nr. Sol.	360.96	120.93	65.32	46.25	64.42	68.50	262.92	277.51	289.54	322.73	88.82
	Time (s)	1668.57	1009.72	4.40	4.57	5.53	2.58	34.83	37.60	41.35	19.85	—
20×10	HV	0.2333	0.1597	0.3725	0.3878	0.3917	0.4039	0.3874	0.4097	0.4133	**0.4245**	0.3841
	Gap (%)	39.25	58.43	3.01	−0.96	−1.99	−5.14	−0.86	−6.66	−7.60	−10.52	0.00
	Nr. Sol.	398.97	119.22	67.14	61.14	68.45	74.60	275.70	285.84	305.61	350.85	83.85
	Time (s)	2757.34	1634.10	7.48	7.70	9.07	4.02	62.10	66.94	72.69	34.65	—

Results. AC-PC exceeds DCAN in six of eight comparisons (Table 10), with small shortfalls in both modes at 20×5. Sampling improves AC-PC HV at all four sizes.

H TRANSFER TO PUBLIC BENCHMARKS

We transfer the 15×10 three-objective model to mk, rdata, edata, and vdata, with 10, 40, 40, and 40 instances, respectively. Evaluation follows Appendix A.

Table 11: Public-benchmark transfer without retraining: 10 mk instances and 40 per Hurink set. Metrics and ranking follow Table 1.

		Metaheuristics		Greedy				Sampling				
Dataset	Metric	NSGA-II	MOEA/D	Hyper	WI-DAN	DCAN	AC-PC	Hyper	WI-DAN	DCAN	AC-PC	CP-SAT
mk	HV	0.3416	0.2517	0.2977	0.2519	0.2743	0.3332	0.3172	0.2878	0.3047	**0.3648**	0.5085
	Gap (%)	32.82	50.49	41.45	50.46	46.06	34.48	37.62	43.39	40.07	28.27	0.00
	Nr. Sol.	275.70	94.30	34.30	18.10	27.50	43.70	88.40	51.10	84.50	134.20	68.90
	Time (s)	1787.04	1122.09	4.59	4.78	5.73	2.92	39.05	41.10	44.05	21.97	—
rdata	HV	**0.6586**	0.5652	0.6004	0.5900	0.6060	0.6342	0.6166	0.6033	0.6367	0.6499	0.7296
	Gap (%)	9.73	22.53	17.71	19.14	16.94	13.08	15.50	17.31	12.73	10.92	0.00
	Nr. Sol.	8.98	6.98	5.28	6.08	5.03	6.78	9.00	9.28	9.48	10.05	15.18
	Time (s)	2120.14	1164.22	5.34	5.68	6.10	3.20	47.79	44.82	47.79	26.30	—
edata	HV	**0.6439**	0.5773	0.5256	0.5212	0.5272	0.5537	0.5441	0.5427	0.5689	0.5756	0.7137
	Gap (%)	9.79	19.11	26.36	26.98	26.14	22.42	23.76	23.96	20.30	19.35	0.00
	Nr. Sol.	10.40	6.70	4.23	3.95	3.40	5.47	6.23	6.23	7.43	7.47	16.00
	Time (s)	2106.18	1140.89	5.32	5.66	6.20	3.20	47.25	45.20	47.78	26.24	—
vdata	HV	0.7180	0.6133	0.6957	0.6799	0.7143	0.7281	0.7056	0.6800	0.7378	**0.7442**	0.7907
	Gap (%)	9.20	22.44	12.02	14.01	9.66	7.92	10.77	14.00	6.69	5.89	0.00
	Nr. Sol.	11.70	6.50	6.70	7.28	6.90	7.50	9.18	10.43	10.90	11.85	12.03
	Time (s)	2236.94	1189.89	5.34	5.65	6.20	3.27	47.05	45.85	48.26	26.43	—

AC-PC exceeds DCAN HV in all eight comparisons (Table 11). NSGA-II remains stronger than AC-PC sampling on rdata and edata.

On rdata, edata, and vdata, each operation has identical processing times across alternative machines, making cost assignment-independent under Equation 2. This cost equals both its lower bound and reference value. By the normalization in Appendix A, the constant dimension cancels, so the reported HV equals the two-dimensional normalized HV for makespan and average flowtime.

Appendix I examines reduced evolutionary search budgets. Appendix J compares AC-PC and DCAN with 10, 100, and 400 samples per preference.

I PUBLIC BENCHMARKS UNDER REDUCED SEARCH BUDGETS

Evaluation protocol. We reduce evolutionary search budgets on the public instances and three-objective task in Appendix H. NSGA-II uses a population of 100 and runs for 50 or 100 generations. MOEA/D uses 105 decomposition directions with configured offspring-evaluation budgets of 4,000 or 8,000, completing each search sweep before termination. The four settings produce 520 instance evaluations. Neural results are reused from Table 11.

Results. Increasing the search budget improves mean HV for both evolutionary algorithms on every dataset. Doubling the budget gives relative HV gains of 4.24–9.87% for NSGA-II and 2.85–5.58% for MOEA/D, while their recorded runtimes approximately double. NSGA-II with 100 generations achieves higher HV than MOEA/D with the 8,000-evaluation budget on all four datasets.

Greedy AC-PC exceeds all four evolutionary settings on mk, rdata, and vdata; sampling improves HV further and returns the largest mean nondominated sets on these datasets. On edata, NSGA-II with 100 generations has the highest HV and largest set, while even 50 generations exceed AC-PC greedy HV.

Table 12: Public-benchmark results with reduced evolutionary search budgets. All neural-policy columns are reused from Table 11. Metrics and ranking follow Table 1; sampling uses ten schedules per preference.

		Metaheuristics				Greedy			Sampling		
Dataset	Metric	NSGA-II$_{50}$	NSGA-II$_{100}$	MOEA/D$_{4000}$	MOEA/D$_{8000}$	WI-DAN	DCAN	AC-PC	WI-DAN	DCAN	AC-PC
mk	HV	0.2563	0.2816	0.1979	0.2090	0.2519	0.2743	0.3332	0.2878	0.3047	**0.3648**
	Gap (%)	49.59	44.62	61.08	58.91	50.46	46.06	34.48	43.39	40.07	28.27
	Nr. Sol.	87.80	123.40	49.10	63.90	18.10	27.50	43.70	51.10	84.50	134.20
	Time (s)	13.97	28.13	13.14	25.44	4.78	5.73	2.92	41.10	44.05	21.97
rdata	HV	0.5616	0.5898	0.4983	0.5125	0.5900	0.6060	0.6342	0.6033	0.6367	**0.6499**
	Gap (%)	23.03	19.15	31.70	29.75	19.14	16.94	13.08	17.31	12.73	10.92
	Nr. Sol.	6.70	7.83	5.12	5.35	6.08	5.03	6.78	9.28	9.48	10.05
	Time (s)	15.11	29.65	13.65	26.66	5.68	6.10	3.20	44.82	47.79	26.30
edata	HV	0.5617	**0.5866**	0.5168	0.5342	0.5212	0.5272	0.5537	0.5427	0.5689	0.5756
	Gap (%)	21.30	17.80	27.59	25.15	26.98	26.14	22.42	23.96	20.30	19.35
	Nr. Sol.	7.58	8.30	5.30	5.85	3.95	3.40	5.47	6.23	7.43	7.47
	Time (s)	14.77	29.32	13.40	26.17	5.66	6.20	3.20	45.20	47.78	26.24
vdata	HV	0.6320	0.6587	0.5506	0.5675	0.6799	0.7143	0.7281	0.6800	0.7378	**0.7442**
	Gap (%)	20.08	16.69	30.37	28.23	14.01	9.66	7.92	14.00	6.69	5.89
	Nr. Sol.	7.45	8.93	5.35	5.03	7.28	6.90	7.50	10.43	10.90	11.85
	Time (s)	15.53	30.63	14.34	27.75	5.65	6.20	3.27	45.85	48.26	26.43

J PUBLIC BENCHMARKS WITH INCREASED INFERENCE SAMPLING

Evaluation protocol. Using the same trained 15×10 model and fixed grid of 105 preferences, we evaluate 10, 100, and 400 sampled schedules per preference on the four public datasets. The 10-sample results are reused from Table 11; larger budgets are generated in batches of ten. The returned set is the nondominated union over all batches, and runtime covers the complete sampling budget.

Table 13: Public-benchmark results with 10, 100, and 400 samples per preference on a fixed grid of 105.

		Metaheuristics		DCAN sampling			AC-PC sampling			
Dataset	Metric	NSGA-II	MOEA/D	10	100	400	10	100	400	CP-SAT
mk	HV	0.3416	0.2517	0.3047	0.3258	0.3351	0.3648	0.3850	**0.3930**	0.5085
	Gap (%)	32.82	50.49	40.07	35.93	34.09	28.27	24.30	22.70	0.00
	Nr. Sol.	275.70	94.30	84.50	162.30	200.40	134.20	259.70	353.70	68.90
	Time (s)	1787.04	1122.09	44.05	411.15	1647.12	21.97	224.04	845.56	—
rdata	HV	0.6586	0.5652	0.6367	0.6487	0.6561	0.6499	0.6639	**0.6695**	0.7296
	Gap (%)	9.73	22.53	12.73	11.10	10.07	10.92	9.00	8.24	0.00
	Nr. Sol.	8.98	6.98	9.48	11.72	14.28	10.05	13.78	14.47	15.18
	Time (s)	2120.14	1164.22	47.79	485.12	1938.43	26.30	255.39	1018.59	—
edata	HV	**0.6439**	0.5773	0.5689	0.5811	0.5883	0.5756	0.5893	0.5971	0.7137
	Gap (%)	9.79	19.11	20.30	18.58	17.58	19.35	17.42	16.33	0.00
	Nr. Sol.	10.40	6.70	7.43	9.40	10.55	7.47	10.03	10.72	16.00
	Time (s)	2106.18	1140.89	47.78	486.01	1944.45	26.24	251.58	1028.15	—
vdata	HV	0.7180	0.6133	0.7378	0.7474	0.7524	0.7442	0.7548	**0.7609**	0.7907
	Gap (%)	9.20	22.44	6.69	5.47	4.84	5.89	4.55	3.77	0.00
	Nr. Sol.	11.70	6.50	10.90	12.60	15.15	11.85	15.10	15.40	12.03
	Time (s)	2236.94	1189.89	48.26	487.44	1957.78	26.43	255.35	1023.65	—

Results. AC-PC exceeds DCAN HV in all 12 dataset–budget comparisons (Table 13). More sampling improves HV on every dataset, with diminishing gains. At 400 samples, AC-PC has the highest non-CP-SAT HV on mk, rdata, and vdata, while NSGA-II remains stronger on edata (0.6439 versus 0.5971).

K FLEXIBLE FLOW SHOP SCHEDULING

Problem and training protocol. In the flexible flow shop scheduling problem (FFSP), all jobs visit the same ordered stages, and each operation can be assigned only to a machine at its stage (Ruiz & Vázquez-Rodríguez, 2010). We use two synthetic benchmark configurations: 15 jobs and five stages with $(3, 2, 3, 2, 2)$ machines, and 20 jobs and four stages with $(3, 3, 3, 3)$ machines. Both configurations contain 12 machines. The objectives remain makespan, average flowtime, and costs, as defined in Equations 1–2.

We train a separate three-objective AC-PC model from scratch for each configuration, following Appendix A. Validation uses 100 instances and 45 preferences, with checkpoint selection by validation performance. The objective multipliers for makespan, flowtime, and costs are $(1, 4/4.5, 4/10)$ for both configurations.

Evaluation and results. Table 14 evaluates 100 test instances per configuration using the three-objective preference grid and decoding protocol in Appendix A, with the benchmark's instance-specific bounds and reference points.

Table 14: FFSP with makespan, average flowtime, and costs: 100 instances per configuration, with separate training at each size. Size denotes jobs $\times$ stages, with 12 machines in both cases. Metrics, Gap calculation, and ranking follow Table 1.

		Metaheuristics		Greedy				Sampling				
Size	Metric	NSGA-II	MOEA/D	Hyper	WI-DAN	DCAN	AC-PC	Hyper	WI-DAN	DCAN	AC-PC	CP-SAT
15×5	HV	**0.4492**	0.3202	0.3808	0.3898	0.4014	0.4005	0.4175	0.4336	<u>0.4385</u>	0.4379	0.5541
	Gap (%)	18.92	42.20	31.27	29.66	27.56	27.72	24.65	21.74	20.85	20.98	0.00
	Nr. Sol.	417.96	163.73	30.27	30.94	33.38	32.69	77.34	104.36	108.80	102.30	76.52
	Time (s)	639.38	359.20	1.75	1.73	2.47	0.88	13.26	13.14	15.05	7.07	—
20×4	HV	0.4039	0.2925	0.4057	0.3082	0.4200	0.4293	0.4374	0.3647	<u>0.4521</u>	**0.4598**	0.5424
	Gap (%)	25.54	46.08	25.20	43.17	22.56	20.85	19.35	32.76	16.65	15.22	0.00
	Nr. Sol.	477.05	207.32	46.69	13.02	50.03	49.62	151.83	111.49	177.35	193.42	75.71
	Time (s)	699.80	389.60	2.13	2.20	2.65	1.05	16.84	16.92	17.94	8.83	—

AC-PC is slightly below DCAN in both modes at 15 jobs and five stages, where NSGA-II achieves the highest non-CP-SAT HV. At 20 jobs and four stages, AC-PC exceeds DCAN in both modes, and sampling achieves the best non-CP-SAT HV. Sampling improves AC-PC at both configurations, while CP-SAT retains higher HV on both.

L JOB SHOP SCHEDULING

Problem and training protocol. We evaluate synthetic JSSP instances at 6×6, 10×10, 15×15, and 20×20. Each operation has a fixed machine, so the cost objective is constant across feasible schedules. We therefore optimize makespan and total tardiness, using the benchmark's due dates, normalization bounds, and reference points.

We train one two-objective AC-PC model from scratch per size, following Appendix A. The objective multipliers $(\alpha_{\mathrm{mk}}, \alpha_{\mathrm{td}})$ are $(1, 2/2.6)$, $(1, 4/9)$, $(1, 6.5/25)$, and $(1, 8.7/39)$ for 6×6, 10×10, 15×15, and 20×20, respectively. Validation uses 100 instances and 51 preferences, with checkpoint selection by validation HV.

Evaluation. We test 100 instances per size using the two-objective preference grid and decoding protocol in Appendix A.

Table 15: Synthetic JSSP with makespan and total tardiness objectives. Results are means over 100 instances per size, with 101 preferences and separate training at each size. Sampling uses ten schedules per preference. Metrics and ranking follow Table 1.

		Metaheuristics		Greedy				Sampling				
Size	Metric	NSGA-II	MOEA/D	Hyper	WI-DAN	DCAN	AC-PC	Hyper	WI-DAN	DCAN	AC-PC	CP-SAT
6×6	HV	<u>0.8703</u>	**0.8729**	0.8354	0.7722	0.8002	0.8185	0.8450	0.8251	0.8244	0.8439	0.8778
	Gap (%)	0.85	0.56	4.83	12.02	8.84	6.75	3.73	6.00	6.07	3.86	0.00
	Nr. Sol.	4.04	4.03	2.73	1.07	1.56	2.07	3.08	2.24	2.22	3.05	3.17
	Time (s)	998.01	194.50	0.40	0.39	0.61	0.25	1.57	1.46	1.82	0.97	—
10×10	HV	**0.8867**	0.8592	0.8767	0.8412	0.8597	0.8674	<u>0.8844</u>	0.8761	0.8817	0.8839	0.9203
	Gap (%)	3.66	6.64	4.74	8.60	6.58	5.75	3.91	4.81	4.20	3.95	0.00
	Nr. Sol.	4.18	4.52	3.48	1.32	2.25	2.76	4.23	3.20	4.36	4.78	5.35
	Time (s)	980.67	662.93	1.66	1.72	2.35	0.98	10.91	11.69	13.49	8.07	—
15×15	HV	0.8776	0.7955	0.9003	0.8797	0.8935	0.8997	<u>0.9133</u>	0.9074	0.9104	**0.9134**	0.9494
	Gap (%)	7.57	16.21	5.18	7.34	5.89	5.24	3.81	4.42	4.11	3.79	0.00
	Nr. Sol.	4.19	2.85	3.97	1.37	2.98	4.15	6.34	4.42	5.92	6.72	13.27
	Time (s)	4823.55	1694.13	7.57	7.62	9.71	4.74	68.58	69.46	82.01	49.45	—
20×20	HV	0.8394	0.7791	0.9164	0.9108	0.9145	0.9167	0.9239	0.9224	<u>0.9245</u>	**0.9255**	0.9576
	Gap (%)	12.34	18.65	4.31	4.89	4.50	4.27	3.52	3.68	3.46	3.35	0.00
	Nr. Sol.	3.02	2.50	4.33	3.21	4.18	4.69	6.60	5.68	6.85	6.87	10.50
	Time (s)	19926.21	4561.18	28.24	28.29	30.35	18.13	267.84	275.62	278.54	184.34	—

Results. AC-PC exceeds DCAN in HV and nondominated-vector count in all eight comparisons (Table 15), with relative HV gains of 0.11%–2.37%. Sampling improves AC-PC at every size. Hyper and NSGA-II remain stronger at the two smaller sizes; Hyper also leads in greedy mode at 15×15, where sampling is nearly tied. At 20×20, AC-PC leads the neural baselines in both modes. Both AC-PC modes exceed NSGA-II and MOEA/D at the two larger sizes, while CP-SAT retains the highest HV throughout.