RETHINKING NEURAL COMBINATORIAL OPTIMIZATION FOR JOB SCHEDULING PROBLEMS VIA TRANSITION-AWARE DUAL-SCALE SOLVER

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural scheduling policies face a representation–computation trade-off: compact candidate features can leave important action-induced changes implicit, whereas encoding complete successor states repeatedly processes largely unchanged information. Meanwhile, dependencies accumulate across decisions, while competition must be resolved among a changing set of feasible candidates, requiring both persistent context and detailed local comparison. To address these challenges, we propose the Transition-Aware Dual-Scale Solver (TADS). TADS combines a transition-complete action representation with a dual-scale encoder. For each feasible action, a task-specific compiler records the induced state changes as sparse typed edits. These edits reconstruct the canonical successor from the current state, while an order-invariant encoder maps each edit set to a transition embedding. The dual-scale encoder compares candidates through global–local reasoning. Global recurrent memory integrates the residual-state summary with the previously executed transition embedding to maintain context across decisions. Conditioned on this context, relation-aware local attention compares candidate transition embeddings and their scheduling relationships to guide action selection. The selected transition embedding feeds the next memory update, allowing subsequent decisions to account for the committed change. Experiments with independently trained policies across six scheduling formulations demonstrate competitive solution quality on synthetic test sets and public benchmarks.

1 INTRODUCTION

Job scheduling assigns resources and processing orders under precedence, eligibility, and timing constraints. It supports manufacturing, logistics, and computing, with different constraints defining formulations and objectives such as makespan and total weighted tardiness (Pinedo, 2016). Exact optimization methods can certify optimality but may incur substantial computational cost on large instances. Dispatching rules and constructive heuristics offer inexpensive decisions, with effectiveness depending on scheduling structure (Haupt, 1989; Nawaz et al., 1983). Learned constructive policies exploit recurring task structure to balance solution quality against inference cost by conditioning each assignment on the partial schedule (Bengio et al., 2021).

Neural constructive policies learn sequential decisions (Vinyals et al., 2015; Bello et al., 2017; Kool et al., 2019), with graph and attention encoders modeling job, operation, and resource interactions (Zhang et al., 2020; Kwon et al., 2021; Song et al., 2023; Wang et al., 2024). Recent work further simplifies residual states and improves policy optimization across formulations (Xiao et al., 2026; Choi et al., 2025). Yet two assignments evaluated from the same residual state may occupy different machines and release different operations, changing the choices available to later decisions. State and action features can support inference of these effects. Explicit transition features expose these consequences directly for comparing feasible scheduling alternatives.

Two design challenges follow. **1. Compact and complete transition descriptions.** A scheduling decision can jointly change machine availability, operation progress, and subsequent feasible choices. Selected consequence features may leave coupled changes implicit, whereas encoding each full successor repeatedly processes unchanged information. The challenge is to expose these

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.

changes for candidate comparison without duplicating the shared state, while retaining enough symbolic information to reconstruct the canonical successor. Reconstruction verifies completeness relative to the current state. **2. Temporal context and candidate interaction.** Executed assignments progressively reshape resource contention, while each decision requires comparison within a changing feasible frontier. For example, a locally attractive assignment can occupy a machine needed by later operations with fewer eligible alternatives, creating downstream delays. A recurrent summary can maintain compressed context across steps, but alone does not explicitly compare current candidates. Repeated full-residual attention can revisit unchanged content. The architectural challenge is to couple context maintenance with detailed candidate comparison through committed changes to resource availability and the remaining feasible choices.

We propose the **T**ransition-**A**ware **D**ual-Scale **S**olver, abbreviated as TADS. Its central design is to use the same transition embedding for comparing a feasible action and, if selected, updating the context for subsequent decisions. For each candidate, a task-specific compiler directly emits sparse typed edits describing resource updates, job progress, and feasibility changes. Together with the current canonical state, these edits reconstruct the exact canonical successor under the prescribed schedule-generation scheme. An order-invariant encoder maps each edit set to a transition embedding for scoring. Global recurrent memory integrates residual-state summaries with the last executed transition embedding, while relation-aware local attention compares current candidates under this context. Relations capture resource contention and mutually exclusive choices. The selected embedding then feeds the next memory update, linking evaluated consequences to committed scheduling changes. Reusing features computed during candidate evaluation also avoids separately encoding the committed edit set for the next memory update. Task-specific compilers and independently trained policies instantiate this common interface across scheduling formulations, preserving their action spaces and feasibility constraints. Our contributions are threefold:

- **Transition-complete action representation.** We introduce task-specific compilers that directly emit sparse typed edits for feasible actions. These edits reconstruct the exact canonical successor from the current state, providing a verifiable transition description without duplicating unchanged scheduling fields across candidates.
- **Transition-aware dual-scale architecture.** We couple candidate comparison and memory updates by reusing the selected transition embedding. Relation-aware local attention compares feasible actions under global recurrent context; the selected embedding then feeds the next memory update, carrying action effects across decisions.
- **Cross-formulation instantiation and evaluation.** We evaluate TADS across six distinct scheduling formulations, training a separate policy for each. Results show competitive solution quality on synthetic test sets and public benchmarks, with generalization from one training size per formulation to larger unseen test sizes. Controlled ablations assess the representation and encoder components under matched settings.

2 Related Work

A line of neural scheduling research models job–resource structure through graph policies (Zhang et al., 2020; Park et al., 2021; Cho et al., 2024), matrix encoders (Kwon et al., 2021), and heterogeneous graph or dual-attention architectures (Song et al., 2023; Wang et al., 2024). ReSched simplifies residual states (Xiao et al., 2026), while BQ-NCO reduces construction state spaces through bisimulation quotienting (Drakulic et al., 2023). We focus on candidate-specific state changes: sparse typed edits reconstruct each canonical successor from the current state under a fixed schedule-generation scheme. At the architectural level, Mamba-based scheduling encodes operation and machine sequences and decodes their interactions through cross-attention (Cao et al., 2026); CaDA combines graph-wide and sparse node attention for routing (Li et al., 2025). Our encoder couples memory across decisions with relation-aware attention over current candidate transitions, feeding the selected transition embedding into the next memory update. A separate line of work studies policy optimization through imitation, self-labeling, multiple starts, and adaptive self-imitation (Li et al., 2024; Corsini et al., 2024; Kwon et al., 2020; Choi et al., 2025). GOAL jointly trains a shared backbone with problem-specific adapters across routing, scheduling, and graph problems (Drakulic et al., 2025). We apply a common transition interface and dual-scale architecture across six scheduling formulations, with task-specific compilers capturing their action effects.

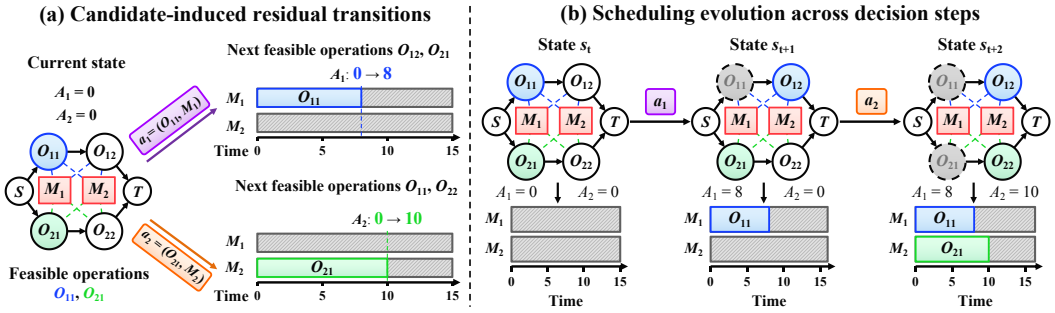


Figure 1: Effects of alternative assignments (a) and evolution of a partial schedule through successive commitments (b). A_m denotes machine availability time; gray operation nodes are committed. O_{ij} denotes operation j of job i; S/T are source/sink nodes. Bars show processing intervals.

3 Preliminaries

Job Scheduling Problems. For formulation $k \in \mathcal{K}$, let $I \sim \mathcal{D}_k$ be a scheduling instance and τ a complete sequence of feasible construction decisions. A fixed schedule-generation scheme with deterministic tie-breaking maps τ to a schedule satisfying resource, precedence, and timing constraints. Across the six evaluated formulations, $C_k(\tau)$ measures total weighted tardiness for SMSP/UPMSP and makespan otherwise. The policy π_{θ_k} is trained to minimize

$$\min_{\theta_k} \mathbb{E}_{I\sim\mathcal{D}_k,\, \tau\sim\pi_{\theta_k}(\cdot|I)} \left[C_k(\tau)\right], \tag{1}$$

where $\mathcal{D}_k$ is the instance distribution; Appendix A defines the six formulations.

MDP Formulation and Residual States. Construction is a finite-horizon deterministic MDP: feasible $a_t \in \mathcal{A}_k(s_t^k)$ induces $s_{t+1}^k = T_k(s_t^k, a_t)$ until all assignments are committed. Here, t indexes decisions, not physical time; processing intervals may overlap. A task adapter provides the canonical residual state

$$\bar{s}_t^k = \psi_k(s_t^k) = (\mathcal{V}_t, \mathcal{E}_t, \mathbf{X}_t, \mathcal{F}_t, c_t), \tag{2}$$

where $\mathcal{V}_t$ contains residual entities and resources, $\mathcal{E}_t$ their relations, $\mathbf{X}_t$ instance and scheduling attributes in original units, and c_t objective-relevant quantities. The frontier $\mathcal{F}_t = \mathcal{A}_k(s_t^k)$ contains full actions, such as operation–machine pairs in Figure 1. Attributes are normalized only for neural encoding. Under the fixed scheme, this state preserves feasible continuations and costs, as formalized in Appendix C.1. Panel (a) shows that choosing an operation–machine pair advances a job and occupies a resource, changing subsequent feasible operations and their earliest start times while leaving most state fields unchanged. Panel (b) shows how committed assignments shape later choices. These dynamics motivate distinguishing each candidate's joint effects on jobs and resources and assessing them against the constraints established by the current partial schedule.

4 Methods

Figure 2 presents TADS for sequential schedule construction. Job and operation assignments alter resource availability and other jobs' feasible choices. TADS exposes these consequences as typed edits, compares candidates under shared context, and feeds the selected transition into recurrent memory. Appendix B gives implementation details and algorithms.

4.1 Transition-Complete Action Representation

For each feasible scheduling decision $a \in \mathcal{A}_k(s_t^k)$, define its canonical successor and corresponding edits:

$$\bar{s}_{t+1}^{k,a} = \psi_k\big(T_k(s_t^k, a)\big), \qquad \Delta_t^k(a) = \mathrm{Diff}_k\Big(\bar{s}_t^k, \bar{s}_{t+1}^{k,a}\Big). \tag{3}$$

Write $\bar{s} = (P, \mathcal{R}_k(P))$: P stores instance data and scheduling fields, with resource availability and job progress; $\mathcal{R}_k$ derives masks and relations. Each edit $e = (\ell, i, j, x^-, x^+)$ specifies a type, affected entity, optional related entity, and old/new values; (ℓ, i, j) identifies the edited field.

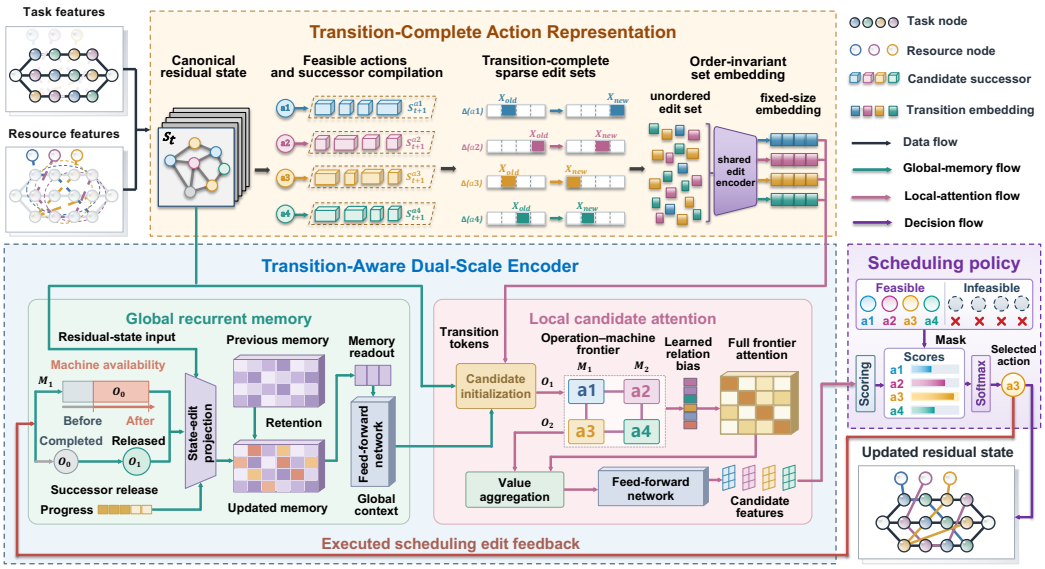


Figure 2: Overview of TADS. Typed action edits form transition embeddings. Global memory integrates residual-state context and executed-transition feedback, while relation-aware local attention compares feasible candidates. The selected transition feeds the next memory update.

Transition completeness. A symbolic edit set is transition-complete when, for every reachable state s and feasible action a under the fixed schedule-generation scheme,

$$\mathrm{Apply}_k\big(\psi_k(s), \Delta^k(s,a)\big) = \psi_k(T_k(s,a))\,. \tag{4}$$

Primary edits use typed keys q, successor map P^a, and $\perp$ for absent keys:

$$\begin{aligned} \Delta_P(s,a) &= \{(q, P(q), P^a(q)) : q \in \mathrm{dom}(P) \cup \mathrm{dom}(P^a),\ P(q) \neq P^a(q)\}, \\ \mathrm{Apply}_k(\bar{s}, \Delta^k(s,a)) &= (P \oplus \Delta_P, \mathcal{R}_k(P \oplus \Delta_P)). \end{aligned} \tag{5}$$

The compiler emits changed fields directly; $\oplus$ applies insertions, deletions, and updates while preserving other fields. Since $P \oplus \Delta_P = P^a$, deterministic reconstruction gives Equation 4, as derived in Appendix C.1. Completeness is relative to the current state; sparsity depends on changed primary fields, since unchanged resource and job information is preserved.

Completeness is defined by the effects of scheduling decisions. At fixed s, Equation 4 implies $\Delta^k(s,a) = \Delta^k(s,b) \Rightarrow \psi_k(T_k(s,a)) = \psi_k(T_k(s,b))$: equal-cost decisions require distinct edits when their resource or frontier changes differ. Full successors repeat unchanged fields; selected timing features may omit coupled resource updates and changes to the feasible frontier.

Encoding sparse edits. Encode each $e \in \mathcal{E}_a$ by edit type, affected job or resource features, and normalized changes, using invariant pooling (Zaheer et al., 2017):

$$h_e = \phi_e^k\left[E_\ell(\ell_e), h_{i_e}, h_{j_e}, \nu(x_e^-), \nu(x_e^+), \nu(x_e^+ - x_e^-)\right], \tag{6}$$

$$d_t^k(a) = \rho_\Delta^k\left(\sum_{e\in\mathcal{E}_a} h_e, \max_{e\in\mathcal{E}_a} h_e, \log(1+|\mathcal{E}_a|)\right). \tag{7}$$

Missing fields use masked nulls; nonnumerical edits have zero difference channels. Pooling aggregates affected job, operation, and resource features independently of edit order. These embeddings support candidate scoring; exact reconstruction uses the current state and symbolic edits.

4.2 Transition-Aware Dual-Scale Encoder

Global Recurrent Memory. Pool job and resource features into an invariant summary,

$$u_t^k = \rho_s^k\left(\left\{\phi_v^k(x_i) : i \in \mathcal{V}_t\right\}, \left\{\phi_r^k(x_{ij}) : (i,j) \in \mathcal{E}_t\right\}, c_t\right). \tag{8}$$

Initialize feedback with a zero-initialized trainable vector, $\hat{d}^k_{-1} = d^k_{\text{init}}$. Combine the residual summary with the last committed transition and construction progress:

$$z^k_t = W^k_z[u^k_t, \hat{d}^k_{t-1}, \eta(t/H_k)]. \tag{9}$$

Here H_k is the horizon and η embeds progress. Each Gated DeltaNet head (Yang et al., 2025) projects z_t to a normalized key k_t, query q_t, value v_t, and gates $\alpha_t, \beta_t \in (0, 1)$, then updates

$$S_t = S_{t-1}\big[\alpha_t(I - \beta_t k_t k_t^\top)\big] + \beta_t v_t k_t^\top, \qquad o_t = S_t q_t, \tag{10}$$

Residual and feed-forward blocks yield g^k_t; fixed-size S_t encodes prior scheduling commitments.

Local Candidate Attention. For the feasible frontier $\mathcal{F}^\Delta_t = \{\Delta^k_t(a) : a \in \mathcal{A}_k(s^k_t)\}$, initialize each candidate token by combining transition and context features:

$$x^{(0)}_a = \mathrm{LN}\big(W^k_d d^k_t(a) + W^k_g g^k_t + W^k_u u^k_t\big). \tag{11}$$

The multi-hot vector $r^k_t(a, b)$ records shared jobs, operations, resources, stages, frontier identities, and mutually exclusive choices, distinguishing resource contention from other scheduling links. For head h in layer ℓ, relation-aware attention (Shaw et al., 2018) uses:

$$B^h_{ab} = (w^h_B)^\top r^k_t(a, b), \tag{12}$$

$$A^h_{ab} = \mathrm{softmax}_b\left(\frac{(W^h_Q x^{(\ell)}_a)^\top (W^h_K x^{(\ell)}_b)}{\sqrt{d_h}} + B^h_{ab}\right). \tag{13}$$

Transformer sublayers (Vaswani et al., 2017) aggregate heads into $x^{(L)}_a$. Shared g^k_t conditions resource and job-progress comparisons without full-residual encoding per layer.

4.3 Scheduling Policy and Optimization

Let $\mathcal{H}^k_t = (s^k_0, a_0, \ldots, a_{t-1}, s^k_t)$ denote committed scheduling history. Under this context, the masked actor scores feasible decisions and the critic predicts remaining scheduling return:

$$\ell_t(a) = w^\top_\pi \,\mathrm{MLP}_\pi[x^{(L)}_a, g^k_t], \quad \pi_{\theta_k}(a \mid \mathcal{H}^k_t) = \mathrm{softmax}_{a \in \mathcal{A}_k(s^k_t)}\, \ell_t(a), \tag{14}$$

$$V_{\omega_k}(\mathcal{H}^k_t) = \mathrm{MLP}_V\Big[g^k_t, \mathrm{Pool}_{a \in \mathcal{A}_k(s^k_t)}\, x^{(L)}_a\Big]. \tag{15}$$

The committed transition embedding $\hat{d}^k_t = d^k_t(a_t)$ enters the next memory update through Equation 9, linking candidate evaluation to the next construction step.

Recurrent Policy Optimization. For a cost-consistent potential Φ_k with $\Phi_k(s_H) = C_k(\tau)$, use the undiscounted construction reward from the potential decrease:

$$r_t = \Phi_k(s_t) - \Phi_k(s_{t+1}), \qquad \gamma = 1. \tag{16}$$

The potential combines committed makespan or incurred weighted tardiness with a task-specific remaining-work or remaining-job lower bound. Terminal correction gives $\sum_{t=0}^{H-1} r_t = \Phi_k(s_0) - C_k(\tau)$, so maximizing return minimizes terminal cost. With GAE advantages $\widehat{A}_t$ and ratio $\varrho_t = \tilde{\pi}_{\theta_k,b}(a_t)/\pi_{\theta^{\text{old}}_k}(a_t \mid \mathcal{H}^k_t)$, recurrent PPO (Schulman et al., 2016; 2017) maximizes

$$J_{\text{clip}}(\theta_k) = \mathbb{E}_t\Big[\min\Big(\varrho_t \widehat{A}_t, \mathrm{clip}(\varrho_t, 1-\epsilon, 1+\epsilon)\widehat{A}_t\Big)\Big]. \tag{17}$$

Here ϵ controls clipping; value and entropy terms augment the surrogate. Appendix C.4 details replay from cached, detached context at segment start b, which approximates full-history replay.

Computational Cost. For hidden width d, let n_t, A_t, and $D_t = \sum_a |\Delta_t(a)|$ count residual entities, candidates, and edits. Encoding costs $O((D_t + A_t)d^2)$; per layer, recurrence costs $O(d^2)$ and local attention $O(A_t d^2 + A_t^2 d)$, versus $O(n_t d^2 + n_t^2 d)$ for full-residual attention. Appendix C.5 accounts for adaptation, compilation, encoding, and execution; savings depend on frontier size and compiler cost, and full-successor materialization can erase them.

Table 1: Results on six synthetic scheduling tasks. Obj.: objective; Gap (%): reference-relative gap; ↓: lower is better. Gray and bold: TADS results. Time: total test-set minutes (m) for SMSP/UPMSP/PFSP; per-instance seconds (s) for FFSP/JSSP/FJSP. OR-Tools reports its time limit. Greedy selects the highest-probability action; Sampling retains the best sampled schedule. Negative gaps indicate improvement over the task-specific reference objective.

SMSP									
Size	50			100			500		
Metric	Obj. ↓	Gap (%) ↓	Time (m) ↓	Obj. ↓	Gap (%) ↓	Time (m) ↓	Obj. ↓	Gap (%) ↓	Time (m) ↓
EDD (Heuristic)	0.3268	49.84	0.00	0.3950	66.24	0.00	0.7287	97.70	0.02
DeepACO (Metaheuristic)	0.2296	5.27	1.10	0.2551	7.36	2.60	0.5944	61.30	29.0
MP-ASIL (Sampling)	0.2181	0.00	0.08	0.2376	0.00	0.28	0.3691	0.16	14.5
TADS (Sampling)	**0.2168**	**−0.60**	**0.07**	**0.2357**	**−0.80**	**0.25**	**0.3639**	**−1.25**	**13.60**
UPMSP									
Size	50×3			50×6			100×6		
Metric	Obj. ↓	Gap (%) ↓	Time (m) ↓	Obj. ↓	Gap (%) ↓	Time (m) ↓	Obj. ↓	Gap (%) ↓	Time (m) ↓
ATCSR_{R_m} (Heuristic)	877.0	22.91	6.6	294.5	14.40	6.9	735.6	75.94	10.0
MP-ASIL (Sampling)	751.9	5.37	1.8	275.7	7.09	2.4	458.1	9.57	10.7
Cho et al. (Sampling)	784.4	9.94	1.8	294.2	14.29	2.4	502.4	20.16	10.6
TADS (Sampling)	**743.2**	**4.15**	**1.72**	**271.0**	**5.26**	**2.28**	**444.0**	**6.20**	**9.80**
PFSP									
Size	20×5			50×5			100×5		
Metric	Obj. ↓	Gap (%) ↓	Time (m) ↓	Obj. ↓	Gap (%) ↓	Time (m) ↓	Obj. ↓	Gap (%) ↓	Time (m) ↓
ILS (Metaheuristic)	29.96	2.74	0.15	65.33	4.38	0.30	121.49	3.63	0.60
IGA (Metaheuristic)	29.05	−0.37	1	63.19	0.96	5	118.08	0.72	25
NEH (Heuristic)	29.16	0.00	0.07	62.59	0.00	0.93	117.24	0.00	7
IL (Greedy)	31.93	9.50	0	68.05	8.72	0.02	125.34	6.91	0.03
TADS (Greedy)	**29.04**	**−0.41**	**0.011**	**62.50**	**−0.14**	**0.016**	**117.05**	**−0.16**	**0.041**
MP-ASIL (Sampling)	28.43	−2.50	0.02	62.48	−0.18	0.03	117.37	0.11	0.13
TADS (Sampling)	**28.25**	**−3.12**	**0.018**	**62.20**	**−0.62**	**0.027**	**116.78**	**−0.39**	**0.118**
FFSP									
Size	20×12			50×12			100×12		
Metric	Obj. ↓	Gap (%) ↓	Time (s) ↓	Obj. ↓	Gap (%) ↓	Time (s) ↓	Obj. ↓	Gap (%) ↓	Time (s) ↓
PSO (Metaheuristic)	29.10	21.45	46.17	55.10	15.42	82.46	97.30	11.56	154
MatNet (Greedy)	27.26	13.77	1.22	51.52	7.92	2.17	91.58	5.00	4.97
MACSIM (Greedy)	25.75	7.47	0.28	49.36	3.39	0.43	89.88	1.86	0.96
TADS (Greedy)	**25.35**	**5.80**	**0.27**	**48.95**	**2.53**	**0.41**	**89.20**	**2.27**	**0.90**
MatNet (Sampling)	25.43	6.14	3.88	49.68	4.06	8.91	89.72	2.87	18.00
MACSIM (Sampling)	23.96	0.00	0.49	47.74	0.00	0.91	87.22	0.00	3.60
TADS (Sampling)	**23.80**	**−0.67**	**0.47**	**47.40**	**−0.71**	**0.87**	**86.85**	**−0.42**	**3.40**
JSSP									
Size	30×20			50×20			100×20		
Metric	Obj. ↓	Gap (%) ↓	Time (s) ↓	Obj. ↓	Gap (%) ↓	Time (s) ↓	Obj. ↓	Gap (%) ↓	Time (s) ↓
SPT (Heuristic)	3208.69	65.3	0.721	4469.8	54.9	2.504	7516.12	35.1	16.661
MWKR (Heuristic)	3080.11	58.7	0.731	4273.08	48.1	2.523	7069.72	27.0	16.694
FDD/MWKR (Heuristic)	2883.88	48.6	0.731	3993.45	38.4	2.524	6658.17	19.6	16.723
MOPNR (Heuristic)	2809.62	44.7	0.720	3859.14	33.7	2.504	6385.32	14.7	16.625
L2D (Greedy)	2508.27	29.2	1.804	3522.5	22.1	4.872	6088.68	9.4	28.616
TADS (Greedy)	**2378.4**	**22.51**	**1.72**	**3286.8**	**13.93**	**4.50**	**5688.9**	**2.22**	**24.50**
FJSP									
Size	10×5			20×10			40×10		
Metric	Obj. ↓	Gap (%) ↓	Time (s) ↓	Obj. ↓	Gap (%) ↓	Time (s) ↓	Obj. ↓	Gap (%) ↓	Time (s) ↓
OR-Tools	96.32	0.00	1800	195.98	0.00	1800	365.96	0.00	1800
SPT (Heuristic)	129.82	34.76	0.16	255.17	30.25	0.71	445.17	21.66	1.50
MWKR (Heuristic)	113.23	17.58	0.16	216.11	10.30	0.71	414.82	13.37	1.50
HGNN (Greedy)	111.67	16.03	0.45	215.78	10.15	1.91	416.18	13.75	3.81
DANIEL (Greedy)	106.71	10.87	0.45	198.50	1.31	1.85	371.45	1.52	3.77
TADS (Greedy)	**105.80**	**9.84**	**0.43**	**196.20**	**0.11**	**1.78**	**367.40**	**0.39**	**3.60**
HGNN (Sampling)	105.59	9.66	1.11	214.81	9.64	6.23	415.25	13.49	24.40
DANIEL (Sampling)	101.67	5.57	0.74	193.91	−1.03	6.35	370.48	1.14	21.38
TADS (Sampling)	**100.40**	**4.24**	**0.70**	**192.10**	**−1.98**	**6.05**	**366.60**	**0.17**	**20.10**

Table 2: Public-benchmark gaps (%; lower is better). Baselines retain the budgets reported by Choi et al. (2025), Guo et al. (2024), and Xiao et al. (2026); TADS uses sampled decoding.

PFSP — Taillard									
Size	ILS	IGA	NEH	IL	Q-Learning	Q-NEH	L-M	MP-ASIL	**TADS**
20×5	6.81%	3.36%	2.40%	18.20%	9.90%	0.38%	1.41%	0.37%	**0.31%**
50×5	4.14%	1.97%	0.66%	12.30%	6.24%	0.18%	0.78%	0.22%	**0.17%**
100×5	3.44%	1.03%	0.41%	10.13%	4.87%	0.15%	0.37%	0.16%	**0.14%**

JSSP — Taillard										
Size	Gurobi	OR-Tools	L2D	SN	RASCL	RS	SI-GD	SLIM	MP-ASIL	**TADS**
30×20	13.2%	2.8%	33.6%	23.7%	21.5%	17.7%	13.8%	12.7%	12.8%	**11.60%**
50×20	13.6%	2.8%	26.5%	13.5%	13.2%	10.0%	6.7%	7.6%	7.0%	**6.10%**
100×20	11.0%	3.9%	13.6%	6.7%	5.9%	2.6%	1.7%	2.1%	1.0%	**0.95%**

FJSP — Brandimarte and Hurink								
Benchmark	MWKR	HGNN	MCGA	RS	DANIEL	MP-ASIL	ReSched	**TADS**
Brandimarte	28.91%	18.56%	18.67%	15.40%	9.53%	7.32%	6.61%	**6.15%**
Hurink (edata)	18.60%	8.71%	8.38%	7.90%	9.08%	7.24%	8.13%	**6.85%**
Hurink (rdata)	13.86%	5.57%	5.71%	4.72%	4.95%	4.66%	5.04%	**4.42%**
Hurink (vdata)	4.25%	1.32%	1.40%	0.70%	0.69%	0.50%	0.82%	**0.47%**

5 EXPERIMENTS

We evaluate six scheduling formulations on synthetic and public benchmarks. Section 5.2 reports main results; Section 5.3 analyzes representations, encoder components, attention, and runtime.

5.1 EXPERIMENTAL SETTINGS

Models and data. We train separate models on synthetic SMSP 50, UPMSP 50×3, PFSP 20×5, FFSP 20×12, JSSP 30×20, and FJSP 10×5 instances. Each task/seed uses one checkpoint selected at the training size for all three test sizes in Table 1, without retraining or fine-tuning. Table 2 tests frozen PFSP/JSSP/FJSP checkpoints on public benchmarks. Validation/test instances are held out; public benchmarks are excluded from selection. Appendix D details generators and splits.

Training configuration. Each task uses seeds 0–4 and 400 recurrent PPO updates with four epochs per update. AdamW (Loshchilov & Hutter, 2019) uses a learning rate decaying linearly from 10^{-4} to 10^{-5}. Greedy validation every 25 updates selects the checkpoint with lowest mean objective. Appendix D.2 gives the remaining optimization settings. All task-specific models are trained on a single NVIDIA RTX 4090 GPU.

Baseline methods. Literature baselines follow MP-ASIL for SMSP/UPMSP/PFSP (Choi et al., 2025), including DeepACO (Ye et al., 2023); MACSIM for FFSP (Luttmann & Xie, 2026); L2D for JSSP (Zhang et al., 2020); and DANIEL/OR-Tools for FJSP (Wang et al., 2024; Perron & Furnon, 2025). Public comparisons also draw on Q-NEH and ReSched (Guo et al., 2024; Xiao et al., 2026). Published values retain source reference objectives, budgets, and hardware; Section 5.3 provides controlled comparisons on shared instances.

Metrics and inference. We average TADS objectives and instance-level $\mathrm{Gap}(\%) = 100(C - C^{\mathrm{ref}})/C^{\mathrm{ref}}$ over test instances and five training seeds. Zero references use absolute regret and exact-hit rate. Greedy constructs one schedule; Sampling retains the best of 100 complete schedules per instance in Table 1, or 128/512/100 for PFSP/JSSP/FJSP in Table 2. Both modes share checkpoints and instances; Appendix D.5 specifies aggregation and time units.

5.2 MAIN EXPERIMENTAL RESULTS

Results on synthetic test sets. Table 1 shows lower reported sampled objectives than MP-ASIL on SMSP, UPMSP, and PFSP at every size. Greedy TADS improves over L2D at all JSSP sizes, reach-

ing 2.22% versus 9.40% Gap at 100×20. FFSP/FJSP objectives improve over MACSIM/DANIEL across sizes under both decoding modes, although MACSIM retains a lower greedy Gap on FFSP 100×12, at 1.86% versus 2.27%. These results support generalization from a single training size per task to larger unseen sizes, with competitive quality across the evaluated scales. Section 5.3 compares timing under shared hardware and evaluator settings.

Results on public benchmarks. Table 2 covers Taillard PFSP/JSSP (Taillard, 1993) and Brandimarte/Hurink FJSP (Brandimarte, 1993; Hurink et al., 1994). TADS achieves the lowest reported gaps among the listed methods in all PFSP and FJSP groups and at JSSP 100×20. OR-Tools remains better on JSSP 30×20 and 50×20, with 2.8% in both groups versus 11.60% and 6.10%. These results assess transfer to public instances without benchmark-specific fine-tuning. Appendix E.6 reports objectives, gaps, and runtimes for both decoding modes.

5.3 Analysis

Table 3: Component ablation: gap (%); parentheses give differences from Full TADS in percentage points. TC: Transition-Complete representation; LCA: Local Candidate Attention; GRM: Global Recurrent Memory; DS: Dual-Scale encoding (LCA+GRM). Gray and bold identify Full TADS.

Variant	SMSP 100	UPMSP 100×6	PFSP 100×5	FFSP 100×12	JSSP 100×20	FJSP 40×10
w/o TC	0.91% (+0.73)	8.81% (+1.33)	0.49% (+0.65)	2.51% (+0.24)	3.34% (+1.12)	1.11% (+0.72)
w/o LCA	0.53% (+0.35)	9.04% (+1.56)	0.14% (+0.30)	2.63% (+0.36)	2.88% (+0.66)	1.20% (+0.81)
w/o GRM	1.00% (+0.82)	8.39% (+0.91)	0.35% (+0.51)	2.16% (−0.11)	3.51% (+1.29)	0.90% (+0.51)
w/o DS	1.44% (+1.26)	9.61% (+2.13)	0.79% (+0.95)	3.05% (+0.78)	4.11% (+1.89)	1.60% (+1.21)
Full TADS	**0.18%**	**7.48%**	**−0.16%**	**2.27%**	**2.22%**	**0.39%**

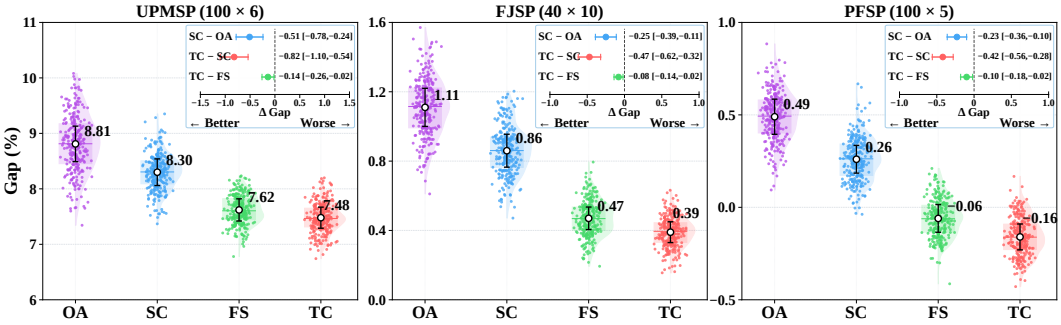


Figure 3: Candidate representations: Gap in percent and pairwise differences with intervals in pp. Negative differences favor the first representation; OA/SC/FS/TC are defined in the text.

Component ablation. Under shared training and greedy decoding, removing DS causes the largest Gap increase on all six tasks in Table 3. This supports combining persistent context with explicit comparisons among current candidates. The changing rankings of TC, LCA, and GRM are consistent with complementary roles for exposing action effects, resolving local competition, and retaining scheduling history. Memory benefits remain task-dependent: removing GRM improves FFSP from 2.27% to 2.16%. Appendix E.1 examines whether these component contributions persist across three sizes per task under the same training and evaluation controls.

Effect of candidate representations. Figure 3 compares OA, original action embeddings; SC, timing/load features; FS, full successors; and TC, transition-complete edits. TC achieves lower Gap across all three tasks. Its advantage over FS suggests that organizing features around action-induced changes can aid candidate scoring despite access to full successor information. This motivates explicit change encoding beyond symbolically complete inputs. Appendix E.2 extends distributions in Figure S3 and quantifies transition sparsity across task sizes.

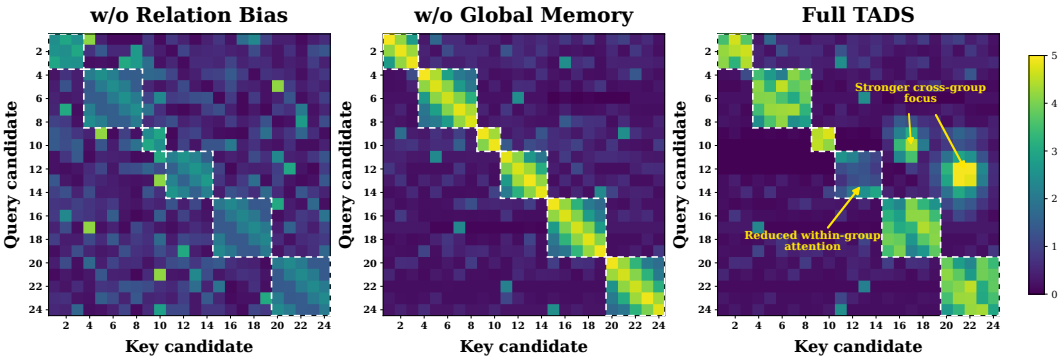


Figure 4: Candidate interactions on FJSP 40×10. Columns index keys and rows index queries; dashed boxes mark within-group blocks. Arrows highlight within-group and cross-group changes.

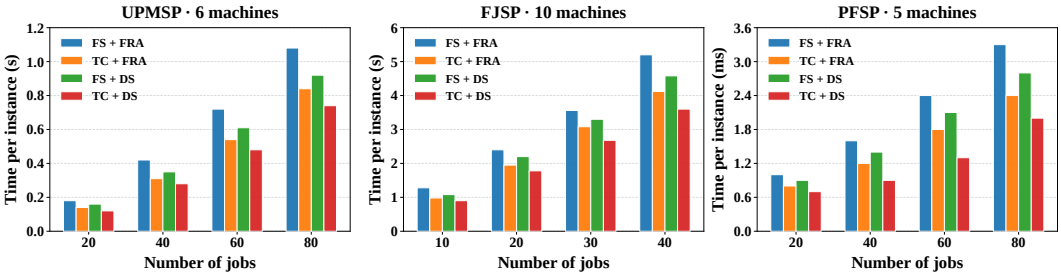


Figure 5: Schedule-construction time per instance: s for UPMSP/FJSP and ms for PFSP; lower is better. FS/TC denote full-successor/sparse-transition representations; FRA/DS denote full-residual attention/dual-scale encoding. TC+DS denotes the full TADS configuration.

Attention-pattern analysis. Figure 4 shows FJSP 40×10 interactions: rows index queries, columns index keys, and dashed boxes mark within-group blocks. Removing relation bias weakens group structure; removing memory makes within-group patterns more pronounced. Full TADS retains structure with stronger cross-group focus, consistent with scheduling relations structuring candidate comparisons and persistent context modulating their emphasis. Figure S2 in Appendix E.1.1 extends this qualitative comparison to five scheduling formulations.

Schedule-construction time. Figure 5 crosses FS/TC with FRA, full-residual attention, and DS, dual-scale encoding. TC+DS has the lowest complete-schedule latency across the plotted job counts. This is consistent with complementary savings: TC encodes sparse action effects, while DS confines detailed attention to candidates and maintains context recurrently. Savings depend on the feasible frontier size and candidate compilation cost. Appendix E.3 extends configurations in Figure S4 and separates compilation, network, total latency, and peak memory.

6 Conclusion

We presented TADS, which represents feasible scheduling actions as executable sparse edits and combines global recurrent memory with local candidate attention. Feeding the selected transition back into memory connects action comparison with subsequent context updates. Experiments cover six formulations and public benchmarks; component and representation analyses support the transition representation and both encoding scales, while the reported timing comparisons favor their combination over the other three configurations at the tested instance sizes. Component effects vary across tasks, supporting complementary roles for temporal context and candidate interactions. Separating exact symbolic reconstruction from learned compression provides a common interface for describing action consequences across task-specific scheduling policies.

AI USE STATEMENT

Generative AI tools were used selectively for language polishing, LaTeX editing, literature search assistance, and limited brainstorming. The research ideas, methodological development, experiments, analyses, and conclusions were developed and validated by the authors.

ETHICS STATEMENT

This study uses synthetic scheduling instances and established public benchmarks without human participants or personal data. Operational deployment of learned scheduling policies requires application-specific consideration of resource-allocation constraints, labor conditions, and distribution shift beyond the benchmark settings evaluated here.

REPRODUCIBILITY STATEMENT

Sections 3–4 specify the state interface, transition-completeness condition, encoder, and optimization procedure. Section 5 reports training hardware and hyperparameters, benchmark sources, decoding modes, and component controls. Table captions define objective, gap, and runtime conventions. Literature-reported baseline values retain their source-specific references, decoding budgets, and hardware settings.

REFERENCES

Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. In *International Conference on Learning Representations Workshop*, 2017. URL https://arxiv.org/abs/1611.09940.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421, 2021. doi: 10.1016/j.ejor.2020.07.063.

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.

Zhi Cao, Cong Zhang, Yaoxin Wu, Yaqing Hou, and Hongwei Ge. Mamba meets scheduling: Learning to solve flexible job shop scheduling with efficient sequence modeling. *arXiv preprint arXiv:2602.21546*, 2026. URL https://arxiv.org/abs/2602.21546.

Sang-Hyun Cho, Hyun-Jung Kim, and Lars Mönch. Reinforcement learning for unrelated parallel machine scheduling with release dates, setup times, and machine eligibility. In *Proceedings of the 2024 Winter Simulation Conference*, pp. 1773–1784, 2024. URL https://informs-sim.org/wsc24papers/con258.pdf.

Inguk Choi, Woo-Jin Shin, Sang-Hyun Cho, and Hyun-Jung Kim. Towards generalizable multi-policy optimization with self-evolution for job scheduling. In *Advances in Neural Information Processing Systems*, volume 38, pp. 24569–24616, 2025. doi: 10.52202/085713-0828. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/2330c4be5e12c504705ba1db8a507966-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.

Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. BQ-NCO: Bisimulation quotienting for efficient neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, pp. 77416–77429, 2023.

Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. GOAL: A generalist combinatorial optimization agent learner. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/826aea2253363fe04e8c4991b2a8869e-Abstract-Conference.html.

Daqiang Guo, Sichao Liu, Shiquan Ling, Mingxing Li, Yishuo Jiang, Ming Li, and George Q. Huang. The marriage of operations research and reinforcement learning: Integration of NEH into Q-learning algorithm for the permutation flowshop scheduling problem. *Expert Systems with Applications*, 255:124779, 2024. doi: 10.1016/j.eswa.2024.124779.

R. Haupt. A survey of priority rule-based scheduling. *OR Spektrum*, 11(1):3–16, 1989. doi: 10.1007/BF01721162.

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.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxBFsRqYm.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21188–21198, 2020.

Yeong-Dae Kwon, Jinho Choo, Iljoo Yoon, Minah Park, Duwon Park, and Youngjune Gwon. Matrix encoding networks for neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 34, pp. 5138–5149, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/29539ed932d32f1c56324cded92c07c2-Abstract.html.

Han Li, Fei Liu, Zhi Zheng, Yu Zhang, and Zhenkun Wang. CaDA: Cross-problem routing solver with constraint-aware dual-attention. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 35438–35456. PMLR, 2025. URL https://proceedings.mlr.press/v267/li25bi.html.

Longkang Li, Siyuan Liang, Zihao Zhu, Chris Ding, Hongyuan Zha, and Baoyuan Wu. Learning to optimize permutation flow shop scheduling via graph-based imitation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 20185–20193, 2024. URL https://ojs.aaai.org/index.php/AAAI/article/view/29998.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Laurin Luttmann and Lin Xie. Multi-action self-improvement for neural combinatorial optimization. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/1bf4cad47f5a54c98fbe7d10516ebf77-Abstract-Conference.html.

Muhammad Nawaz, E. E. Enscore, and Inyong Ham. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95, 1983. doi: 10.1016/0305-0483(83)90088-9.

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 and Vincent Furnon. OR-Tools. Google, 2025. URL https://developers.google.com/optimization/.

Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 5 edition, 2016. doi: 10.1007/978-3-319-26580-3.

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.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of NAACL-HLT, Volume 2 (Short Papers)*, pp. 464–468. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-2074.

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.

Eric Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285, 1993. doi: 10.1016/0377-2217(93)90182-M.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/29921001f2f04bd3baee84a12e98098f-Abstract.html.

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, 2024. doi: 10.1109/TNNLS.2023.3306421.

Xiangjie Xiao, Cong Zhang, Wen Song, and Zhiguang Cao. ReSched: Rethinking flexible job shop scheduling from a transformer-based architecture with simplified states. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=s5pWbwf2tk.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2412.06464.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Helan Liang, and Yong Li. DeepACO: Neural-enhanced ant systems for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-1895.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

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.

APPENDIX CONTENTS

A **Background and Problem Definitions** . . . 14
A.1 Notation and Scope . . . 14
A.2 Scheduling Formulations . . . 14
A.2.1 Single-Operation Tardiness: SMSP and UPMSP . . . 14
A.2.2 Stage-Ordered Flow Shops: PFSP and FFSP . . . 15
A.2.3 Operation-Routing Job Shops: JSSP and FJSP . . . 15
B **Method and Implementation Details** . . . 16
B.1 Scheduling Interfaces and Task Adapters . . . 16
B.2 Direct Sparse Compilation and Checked Application . . . 16
B.3 Representation and Encoder Implementation . . . 17
B.4 Policy Execution and Transition Feedback . . . 18
B.5 Recurrent Training with Contiguous Replay . . . 18
B.6 Greedy and Sampling-Based Decoding . . . 19
C **Mathematical Foundations and Detailed Derivations** . . . 20
C.1 Canonical States and Exact Transition Reconstruction . . . 20
C.2 Scheduling Transitions and Terminal-Consistent Bounds . . . 21
C.3 Encoder Symmetry and Recurrent Memory Derivations . . . 22
C.4 Scheduling Returns, Policy Optimization, and Decoding . . . 23
C.5 Computational Accounting and Evaluation Estimators . . . 24
D **Experimental Settings** . . . 25
D.1 Synthetic Instances and Split Boundaries . . . 25
D.2 Training Configuration . . . 25
D.3 Public Benchmarks and Literature Comparisons . . . 26
D.4 Controlled Variants and Comparison Factors . . . 26
D.5 Aggregation, Uncertainty, and Time Units . . . 26
D.6 Runtime Measurement Protocol . . . 26
E **Supplementary Experiments and Analysis** . . . 27
E.1 Component Contributions and Attention Patterns . . . 27
E.2 Candidate Representations and Transition Sparsity . . . 29
E.3 Computational Efficiency and Scaling . . . 29
E.4 Decoding Budget and Solution Quality . . . 29
E.5 Stability across Training Seeds . . . 32
E.6 Detailed Public-Benchmark Results . . . 33
E.7 Optimization and Architecture Sensitivity . . . 34

A Background and Problem Definitions

A.1 Notation and Scope

The task adapter defines feasibility and exact transitions; the learned policy scores feasible actions. All objectives are minimized. Negative gaps indicate improvement over the stated reference, not over a proven optimum. Construction steps count commitments rather than elapsed processing time. Unless noted, scheduling is deterministic, nonpreemptive, and append-only.

Table S1: Common notation. Task-specific indices are suppressed when unambiguous.

Symbol	Meaning
$I, \mathcal{D}_k$	An instance and the training distribution of formulation k.
n, m, q	Numbers of jobs, machines, and stages, respectively.
$O_{j\ell}, \mathcal{M}_{j\ell}$	Operation ℓ of job j and its eligible machines.
$p_{j\ell m}, r_j, d_j, w_j$	Processing duration, release time, due date, and job weight.
$S_{j\ell}, C_{j\ell}$	Physical start and completion times of an operation.
A_m, R_j	Current machine availability and readiness of the next operation of job j.
$s_t, \bar{s}_t = \psi_k(s_t)$	Environment state and its canonical residual representation.
$P, \mathcal{R}_k(P)$	Primary fields and deterministically reconstructed masks/relations.
$\mathcal{A}_k(s_t), A_t$	Feasible candidate set and its cardinality.
$a_t, T_k(s_t, a_t)$	Committed action and the deterministic state transition.
$\Delta_t(a), \mathcal{E}_a$	Symbolic transition edits and the edits supplied to the encoder.
$d_t(a), \hat{d}_t$	Candidate transition embedding and the selected embedding.
u_t, g_t, S_t	Residual summary, global context, and recurrent fast-weight memory.
$r_t(a, b), A^h_{ab}$	Candidate relation features and head-h attention weight.
$H_k, C_k(\tau), \Phi_k(s)$	Construction horizon, terminal cost, and shaping potential.
$\widehat{A}_t, \varrho_t$	Estimated policy advantage and PPO probability ratio.

A.2 Scheduling Formulations

Figure S1 summarizes the six evaluation formulations and their routing/eligibility structures. Standard terminology follows Pinedo (2016). Appendix C.2 collects transition equations, feasible frontiers, horizons, and lower-bound proofs. SMSP and UPMSP schedule one operation per job; UPMSP adds machine choice. PFSP commits a job across a shared machine permutation. FFSP, JSSP, and FJSP commit successive operations under stage or route constraints. These choices determine the feasible constructive decisions and affected resources. The diagram's processing times are illustrative; objective weights, due dates, and UPMSP releases/setups are specified separately.

Table S2: Task interfaces. $\mathcal{U}$ is the set of uncommitted jobs, L_j the operation count of job j, and K the number of FFSP stages. Horizon counts are construction decisions.

Task	Feasible constructive decision	Horizon	Terminal cost
SMSP	$j \in \mathcal{U}$	n	$\sum_j w_j(C_j - d_j)_+$
UPMSP	$(j, m), j \in \mathcal{U}, m \in \mathcal{M}_j$	n	$\sum_j w_j(C_j - d_j)_+$
PFSP	append $j \in \mathcal{U}$ to the common permutation	n	$C_{\max}$
FFSP	$(j, q_j, m), m \in \mathcal{M}_{q_j}$	nK	$C_{\max}$
JSSP	O_{j,q_j} on its fixed machine	$\sum_j L_j$	$C_{\max}$
FJSP	$(O_{j,q_j}, m), m \in \mathcal{M}_{j,q_j}$	$\sum_j L_j$	$C_{\max}$

A.2.1 Single-Operation Tardiness: SMSP and UPMSP

SMSP and UPMSP assign one operation to each job and minimize total weighted tardiness. In SMSP, jobs with durations $p_j > 0$, due dates d_j, and weights $w_j > 0$ are initially available on a single machine, so the decision is their processing order. Completion relative to each due date

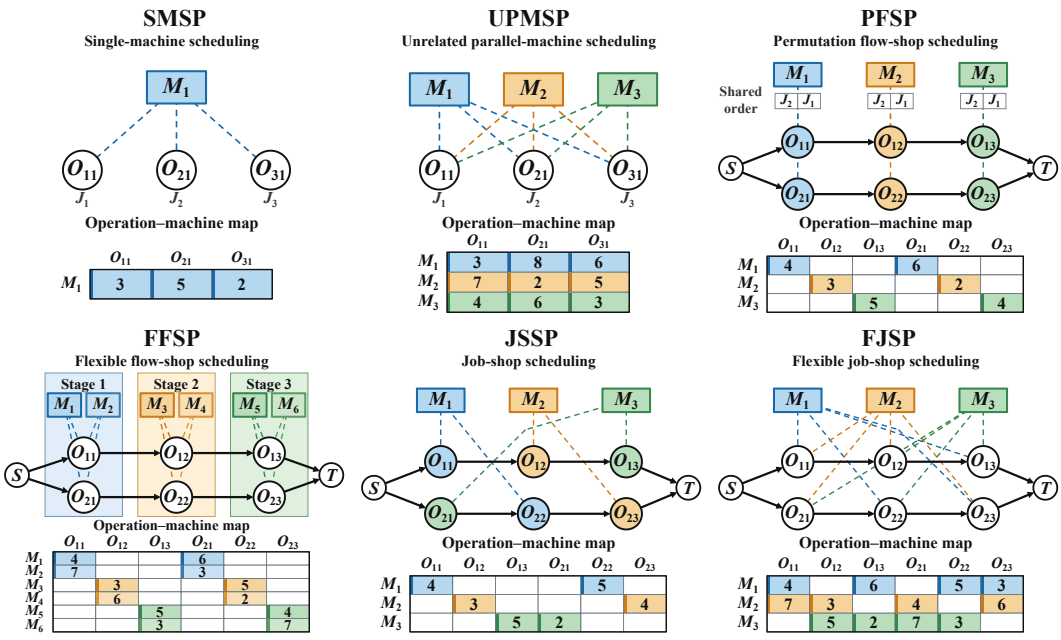


Figure S1: Illustrative structures of the six scheduling formulations. Circles denote operations and rectangles machines; solid arrows show job precedence, dashed links machine eligibility, and S/T dummy source/sink nodes. Operation–machine maps list processing times, with blank cells denoting ineligible pairs. PFSP shares one job order across machines; FFSP restricts alternatives by stage; JSSP fixes routes, whereas FJSP permits operation-specific machine alternatives.

determines the cost; makespan alone would not distinguish these orders. UPMSP introduces a joint job–machine decision: job j has an eligible set $\mathcal{M}_j$, processing time p_{jm}, release time r_j, and nonanticipatory setup $s_{ijm} \geq 0$ determined by the preceding job i. Consequently, the same job order can yield different completion times and costs under different assignments. Figure S1 shows the resource choices, while Table S2 specifies the common job-level horizon and objective.

A.2.2 Stage-Ordered Flow Shops: PFSP and FFSP

PFSP and FFSP minimize makespan while processing jobs through a common sequence of machines or stages. PFSP requires every machine to follow the same job permutation: appending one job extends its complete machine sequence, and the last-machine completion determines the terminal objective. FFSP instead provides parallel machines within each stage. A candidate (j, q_j, m) assigns the next operation of job j to an eligible machine in stage q_j, preserving stage precedence while allowing different jobs to proceed in parallel. Thus PFSP commits an entire job to a shared order, whereas FFSP commits one stage at a time. Their different decision horizons in Table S2 reflect this choice of construction granularity, rather than elapsed processing time.

A.2.3 Operation-Routing Job Shops: JSSP and FJSP

JSSP and FJSP permit job-specific operation routes and minimize makespan. JSSP prescribes a machine for every operation, so candidates are the next operations of unfinished jobs. FJSP additionally chooses a machine from an operation-specific nonempty eligible set, with processing duration depending on that choice. Singleton eligible sets recover fixed-route JSSP. Both formulations require $\sum_j L_j$ operation commitments, independent of the number of machine alternatives. Machine choice changes resource availability even when the alternatives leave the same number of remaining commitments. Public evaluation uses Taillard for JSSP and Brandimarte/Hurink for FJSP (Taillard, 1993; Brandimarte, 1993; Hurink et al., 1994).

B METHOD AND IMPLEMENTATION DETAILS

B.1 SCHEDULING INTERFACES AND TASK ADAPTERS

Primary fields store instance data, resource tails, job progress, and objective accumulators; Equation C.1 reconstructs masks and relations. Candidate evaluation preserves the environment and memory; only the selected action is committed. The auxiliary schedule ledger is excluded from the primary-field statistics in Table S11.

Initially, $(A_m, R_j, q_j) = (0, 0, 1)$. The nonpreemptive SGS appends after job readiness and machine availability, without gap insertion; schedules need not be active. Only UPMSP includes releases and setups. Blocking, no-wait constraints, transportation, and breakdowns are excluded.

Task-specific primary updates. SMSP removes a job from $\mathcal{U}$, advances tail e, and accumulates weighted tardiness y; UPMSP also updates the selected machine's last-job identity for setups. PFSP updates remaining jobs and completion frontier $\mathbf{f}$ sequentially (Equation C.7). FFSP/JSSP/FJSP update job progress, readiness, and the selected machine tail (Equation C.9), remove the committed operation's alternatives, and expose successor choices: next-stage assignments in FFSP, the prescribed successor in JSSP, and eligible successor assignments in FJSP.

B.2 DIRECT SPARSE COMPILATION AND CHECKED APPLICATION

Algorithm S1 directly compiles task-local edits and frontier events. Application checks before-values; derived events never overwrite primary fields. Only optional auditing constructs full successors for reconstruction checks.

Algorithm S1 Direct sparse COMPILEDELTA and checked application

function COMPILEDELTA($k, s, \bar{s}, a,$ audit)
  require $a \in \mathcal{A}_k(s)$ and $\bar{s} = \psi_k(s)$
  Extract P from $\bar{s}$; $j \leftarrow$ AFFECTEDJOB(a)
  $(U_a, \mathcal{F}_j^a) \leftarrow$ TASKLOCALCHANGES(k, P, a)
  $\Delta_P \leftarrow \varnothing$ ▷ U_a: candidate field assignments
  for all $(q, x^+) \in U_a$ **do**
    $x^- \leftarrow P(q)$ ▷ Absent keys evaluate to $\perp$
    if $x^- \neq x^+$ **then**
      $\Delta_P \leftarrow \Delta_P \cup \{(q, x^-, x^+)\}$
    end if
  end for
  $\mathcal{F}_j \leftarrow \{b \in \mathcal{A}_k(s) : \text{AFFECTEDJOB}(b) = j\}$
  $\Delta_D \leftarrow$ FRONTIEREVENTS$(\mathcal{F}_j, \mathcal{F}_j^a) \cup$ TASKEVENTS(k, P, a)
  $\Delta \leftarrow (\Delta_P, \Delta_D)$
  if audit **then**
    $\bar{s}^a \leftarrow \psi_k($READONLYSUCCESSOR$(k, s, a))$
    assert APPLYCHECK$(k, \bar{s}, \Delta) = \bar{s}^a$
  end if
  return Δ
end function

function APPLYCHECK$(k, \bar{s}, \Delta)$
  Extract P and (Δ_P, Δ_D); require a unique typed key per primary edit
  for all $(q, x^-, x^+) \in \Delta_P$ **do**
    assert valid key/type, $x^- \neq x^+$, and $P(q) = x^-$
  end for
  $P' \leftarrow$ COPYPRIMARY(P)
  for all $(q, x^-, x^+) \in \Delta_P$ **do**
    Set $P'(q) \leftarrow x^+$ if $x^+ \neq \perp$; otherwise delete q from P'
  end for
  $\bar{s}' \leftarrow (P', \mathcal{R}_k(P'))$
  return $\bar{s}'$
end function

Reconstruction and execution checks. Tests on small instances covered all feasible candidates along 24 randomized construction orders per task, totaling at least 100 transitions per task.

Sparse/full/policy-view successors agreed; evaluation preserved the live state, and selected patches executed without recompilation. Serialization, stale-edit, objective-replay, and telescoping-reward checks passed; UPMSP avoided full successor-frontier enumeration. These checks cover the tested inputs; reconstruction assumptions appear in Appendix C.

B.3 Representation and Encoder Implementation

B.3.1 Residual Features and Transition Encoding

Symbolic keys locate edits; twelve-channel entity attributes use zero for inapplicable fields (Table S3). Eight global scalars encode objective, potential, progress, remaining/frontier fractions, mean/maximum availability, and objective type. Residual relations cover membership, precedence, eligibility, visits, stages, and competition, with activity/duration/readiness/availability channels and relative same-job/machine/stage competition counts. Only neural features are normalized (Equation C.6).

Table S3: Entity feature groups. Values are task-normalized.

Channels	Scheduling interpretation
Active; status	Residual-problem membership and scheduling status.
Completion fraction	Progress through a job's operations or assignments.
Processing min/mean/max	Duration statistics over eligible processing choices.
Ready or release	Release time or predecessor completion.
Due date; weight	Weighted-tardiness parameters.
Resource availability	Completion of the resource's committed sequence.
Remaining work	Work in uncommitted operations or stages.
Eligibility fraction	Fraction of admissible resources.

Each edit concatenates edit/subject/object type embeddings, subject/object attributes, and value channels: $3 \times 16 + 2 \times 12 + 8 = 80$ dimensions. Values include before/after values, signed/absolute differences, presence flags, and Boolean/semantic indicators; presence separates deletion from zero. Types cover completion, release, readiness, availability, load, cost, relations, and candidate enabling/disabling. Two-layer MLPs map edits and sum/max/$\log(1 + |\mathcal{E}_a|)$ pooling to width 128; the count excludes padding, and empty edit sets use zero log-count, sum, and maximum.

B.3.2 Candidate Relations, Padding, and Attention

Six channels encode shared job, operation, machine, stage, frontier identity, and mutual exclusion. Frontier identity means the same job–operation/stage pair; readiness changes are encoded by edits. Padded keys and actor logits are masked, padded query outputs are zeroed after residual blocks, and terminal states bypass softmax. Relation axes and masks follow candidate permutations (Equation C.12). Tests confirmed edit-pooling invariance; consistently reordering candidates, features, and both relation axes permuted actor logits while preserving critic values.

Table S4: Batched tensor shapes. B is batch size, N residual entities, R residual relations, A padded candidates, and E padded edits per candidate.

Tensor	Shape	Validity mask
Entity features	$B \times N \times 12$	$B \times N$
Residual-relation features	$B \times R \times 4$	$B \times R$
State scalars	$B \times 8$	Per valid environment
Edit subject/object features	$B \times A \times E \times 12$	$B \times A \times E$
Edit values	$B \times A \times E \times 8$	$B \times A \times E$
Candidate relations	$B \times A \times A \times 6$	Candidate pair mask
Transition embeddings	$B \times A \times 128$	$B \times A$
Action logits	$B \times A$	$B \times A$

B.3.3 Network State and Execution Feedback

Global/local branches each have three layers: four global heads with key/value widths 64/128, and eight local heads at width 128. Feed-forward width is 512; dropout is zero. Fast-weight storage is 384 KiB per trajectory, excluding activations and parameters (Equation C.22). Each trajectory has separate memory, cleared at reset, and trainable zero-initialized feedback $\hat{d}_{-1} = d_{\text{init}}$; subsequent feedback uses the selected embedding.

B.4 Policy Execution and Transition Feedback

Algorithm S2 updates recurrent memory M_{t-1} once per decision using executed-transition feedback $\hat{d}_{t-1}$. Actor and critic share encoders; masks exclude padding from attention, logits, and critic pooling.

Algorithm S2 One transition-aware policy step, without environment mutation

function PolicyStep($k, s, t, H_k, M_{t-1}, \hat{d}_{t-1}, \theta, \omega, \text{audit}$)
  $\bar{s}_t \leftarrow \psi_k(s)$; $(\mathcal{C}_t, \mu_t) \leftarrow$ FeasibilityMask$(k, \bar{s}_t)$
  $\mathcal{A}_t \leftarrow \{a \in \mathcal{C}_t : \mu_t(a) = 1\}$
  require $\mathcal{A}_t \neq \varnothing$ ▷ Terminal states are handled by the caller
  $u_t \leftarrow$ StateSummary$(\bar{s}_t)$
  $(M_t, g_t) \leftarrow$ GlobalUpdate$(M_{t-1}, u_t, \hat{d}_{t-1}, \eta(t/H_k))$
  for all $a \in \mathcal{A}_t$ **do**
    $\Delta_t(a) \leftarrow$ CompileDelta$(k, s, \bar{s}_t, a, \text{audit})$
    $d_t(a) \leftarrow \text{Enc}_\theta(\Delta_t(a); \bar{s}_t)$
  end for
  $r_t^{\text{rel}} \leftarrow$ CandidateRelations$(k, \bar{s}_t, \{\Delta_t(a)\}_{a \in \mathcal{A}_t})$
  $X_t \leftarrow$ LocalAttention$(\{d_t(a)\}, g_t, u_t, r_t^{\text{rel}}, \mu_t)$
  Compute actor logits $\ell_t(a)$ and critic value V_t using Equations 14–15
  Set $\ell_t(a) \leftarrow -\infty$ for masked or padded candidates
  $\pi_t \leftarrow \text{softmax}(\ell_t)$
  return $(\pi_t, V_t, M_t, \{d_t(a)\}, \{\Delta_t(a)\}, \mu_t)$
end function

Algorithm S3 Collect a complete trajectory with executed-transition feedback

function Rollout($k, I, \theta, \omega, \text{mode}, \text{audit}$)
  $s_0 \leftarrow$ ResetEnvironment(k, I); $M_{-1} \leftarrow M^{\text{reset}}$; $\hat{d}_{-1} \leftarrow d_{\text{init}}$
  $t \leftarrow 0$; trajectory record $\mathcal{D}_I \leftarrow \varnothing$
  while s_t is not terminal **do**
    $F_t \leftarrow$ PolicyStep$(k, s_t, t, H_k, M_{t-1}, \hat{d}_{t-1}, \theta, \omega, \text{audit})$
    Unpack F_t as $(\pi_t, V_t, M_t, \{d_t(a)\}, \{\Delta_t(a)\}, \mu_t)$
    if mode $=$ greedy **then**
      $a_t \leftarrow \arg\max_{a:\mu_t(a)=1} \pi_t(a)$ using the fixed semantic tie rule
    else
      $a_t \sim \pi_t$ over feasible actions
    end if
    $s_{t+1} \leftarrow$ ExecuteOnce(k, s_t, a_t)
    if audit **then**
      assert ApplyCheck$(k, \psi_k(s_t), \Delta_t(a_t)) = \psi_k(s_{t+1})$
    end if
    $r_t \leftarrow \Phi_k(s_t) - \Phi_k(s_{t+1})$ ▷ $\Phi_k(s_{t+1}) = C_k(\tau)$ if terminal
    $\hat{d}_t \leftarrow d_t(a_t)$ ▷ Only the executed candidate supplies feedback
    Append $(a_t, r_t, \log \pi_t(a_t), V_t, \mu_t, \text{terminal}_{t+1})$ to $\mathcal{D}_I$
    Store the policy input and detached memory/feedback at segment boundaries
    $t \leftarrow t + 1$
  end while
  Set terminal value $V_t \leftarrow 0$; compute terminal objective and return $\mathcal{D}_I$ and schedule
end function

B.5 Recurrent Training with Contiguous Replay

Algorithm S4 computes GAE on complete rollouts, then replays 64-step segments with differentiable feedback. Write $\tilde{\pi}_{\theta,b}(a_t)$ for the conditional replay probability. Padding is excluded from the

loss and advantage normalization. For segments beginning after the episode start, the incoming recurrent memory and previous-transition feedback are cached under the behavior policy and detached during optimization. This implements truncated backpropagation with fixed boundary context and is an approximation to full-history recurrent replay.

Algorithm S4 Task-specific recurrent PPO with stored boundary context

Require: Task k; parameters (θ, ω); updates U; rollout count B; PPO epochs K
for $u = 1, \dots, U$ **do**
  Freeze $(\theta^{\text{old}}, \omega^{\text{old}}) \leftarrow (\theta, \omega)$; clear buffer $\mathcal{D}$
  for $i = 1, \dots, B$ **do**
    Sample $I_i \sim \mathcal{D}_k$
    $(\mathcal{D}_i, \tau_i) \leftarrow \textsc{Rollout}(k, I_i, \theta^{\text{old}}, \omega^{\text{old}}, \text{sample}, \text{false})$
    Add episode record $\mathcal{D}_i$ to $\mathcal{D}$
  end for
  for all episodes in $\mathcal{D}$ **do**
    Set $V_H^{\text{old}} \leftarrow 0$ and $\widehat{A}_H \leftarrow 0$
    for $t = H - 1, \dots, 0$ **do**
      $b_t \leftarrow 1 - \text{terminal}_{t+1}$; $\delta_t^V \leftarrow r_t + b_t V_{t+1}^{\text{old}} - V_t^{\text{old}}$
      $\widehat{A}_t \leftarrow \delta_t^V + \lambda b_t \widehat{A}_{t+1}$; $\widehat{R}_t \leftarrow \widehat{A}_t + V_t^{\text{old}}$
    end for
  end for
  for $e = 1, \dots, K$ **do**
    for all minibatches of complete episodes or contiguous segments $[b, e')$ **do**
      Initialize $(M_{b-1}, \hat{d}_{b-1})$: current reset if $b = 0$;
      otherwise load detached behavior-policy boundary context
      for $t = b, \dots, e' - 1$ in each segment **do**
        Re-encode stored inputs; advance global memory and local attention
        Evaluate current log probability at the recorded a_t and current value V_t
        Select $\hat{d}_t \leftarrow d_t(a_t)$ from the current embeddings; carry M_t forward
        $\varrho_t \leftarrow \exp(\log \tilde{\pi}_{\theta,b}(a_t) - \log \pi_{\theta^{\text{old}}}(a_t \mid \mathcal{H}_t))$
      end for
      Form Equation C.19 on valid, nonpadding decisions only
      Backpropagate through each segment, clip gradient norm, and update (θ, ω)
    end for
  end for
end for

Replay validation. With unchanged behavior-policy parameters, replay reproduced stored log probabilities, pre-decision memory, and selected-transition feedback. Segment starts matched cached boundary states, and recurrent memory retained its fixed shape.

B.6 Greedy and Sampling-Based Decoding

Greedy and Sampling use the same model and feasibility rules. Sampling-N returns the lowest actual terminal objective among N complete trajectories.

Algorithm S5 Greedy decoding or Sampling-N for one instance

Require: Instance I; trained parameters (θ, ω); mode; sample budget $N \geq 1$
Disable dropout; set $R \leftarrow 1$ for greedy and $R \leftarrow N$ for sampling
$C^{\text{best}} \leftarrow +\infty$; $\tau^{\text{best}} \leftarrow \varnothing$
for $i = 1, \dots, R$ **do**
  Reset environment and recurrent memory; assign an independent random stream if sampling
  $(\mathcal{D}_i, \tau_i) \leftarrow \textsc{Rollout}(k, I, \theta, \omega, \text{mode}, \text{false})$
  Check schedule feasibility and evaluate $C_i \leftarrow C_k(\tau_i)$
  If $C_i < C^{\text{best}}$, set $(C^{\text{best}}, \tau^{\text{best}}) \leftarrow (C_i, \tau_i)$
end for
return $(\tau^{\text{best}}, C^{\text{best}})$

C Mathematical Foundations and Detailed Derivations

C.1 Canonical States and Exact Transition Reconstruction

Fix an instance, task k, and deterministic schedule-generation scheme (SGS). Primary fields P retain all generator inputs; $\mathcal{R}_k$ rebuilds masks and relations. Canonical actions identify jobs, operations, and eligible resources.

$$\begin{aligned} \bar{s} = \psi_k(s) = (P(s), \mathcal{R}_k(P(s))), \qquad \bar{T}_k(\bar{s}, a) = \psi_k(T_k(s, a)), \\ \psi_k(s) = \psi_k(\tilde{s}) \implies \begin{cases} \mathcal{A}_k(s) = \mathcal{A}_k(\tilde{s}), \quad \mathbf{1}_{\text{term}}(s) = \mathbf{1}_{\text{term}}(\tilde{s}), \\ \psi_k(T_k(s, a)) = \psi_k(T_k(\tilde{s}, a)), \quad a \in \mathcal{A}_k(s), \end{cases} \\ C_k(\tau) = c_k^{\text{term}}(\psi_k(s_H)). \end{aligned} \tag{C.1}$$

Induction over common feasible prefixes gives identical successors and costs; backward induction on the finite construction tree yields

$$\begin{aligned} \psi_k(s_t) = \psi_k(\tilde{s}_t) \implies \psi_k(s_{t+h}) = \psi_k(\tilde{s}_{t+h}) \implies C_k(\tau) = C_k(\tilde{\tau}), \\ J_k^*(\bar{s}) = \begin{cases} c_k^{\text{term}}(\bar{s}), & \bar{s} \text{ terminal}, \\ \min\limits_{a \in \mathcal{A}_k(\bar{s})} J_k^*(\bar{T}_k(\bar{s}, a)), & \text{otherwise}. \end{cases} \end{aligned} \tag{C.2}$$

Sequence-dependent setups require last-job identities; insertion into idle gaps requires occupied intervals. Machine tails alone apply to the append-only SGS below.

Reconstruction and changed-key support. Let $P^a = P(T_k(s, a))$. Extend both maps to $K = \operatorname{dom} P \cup \operatorname{dom} P^a$ with absence symbol $\perp$. A typed key q identifies one authoritative field. With one assignment per changed key and all before-values checked against the same base,

$$\begin{aligned} K_\Delta = \{q \in K : P(q) \neq P^a(q)\}, \qquad \Delta_P = \{(q, P(q), P^a(q)) : q \in K_\Delta\}, \\ (P \oplus \Delta_P)(q) = \begin{cases} P^a(q), & q \in K_\Delta, \\ P(q) = P^a(q), & q \notin K_\Delta, \end{cases} \\ \text{Apply}_k(\bar{s}, \Delta) = (P \oplus \Delta_P, \mathcal{R}_k(P \oplus \Delta_P)) = \psi_k(T_k(s, a)), \\ P \oplus E = P^a \implies K_\Delta \subseteq \text{keys}(E) \implies |E| \geq |K_\Delta| = |\Delta_P|. \end{aligned} \tag{C.3}$$

The cases cover insertion/deletion through $\perp$; minimality is relative to this direct-assignment schema. Distinct successors and sequential replay satisfy

$$\begin{aligned} \Delta(s, a) = \Delta(s, b) \implies \text{Apply}_k(\bar{s}, \Delta(s, a)) = \text{Apply}_k(\bar{s}, \Delta(s, b)) \\ \implies \psi_k(T_k(s, a)) = \psi_k(T_k(s, b)), \\ P_{t+h} = ((P_t \oplus \Delta_{P,t}) \oplus \cdots) \oplus \Delta_{P,t+h-1}. \end{aligned} \tag{C.4}$$

Sparse commitments, coupled candidate effects. For the operation-level SGS in Appendix C.2, set $a = (j, \ell, m)$, $b = (i, q_i, u)$, $i \neq j$, and $\mathcal{G}_j(\ell) = \{(j, \ell, u) : u \in \mathcal{M}_{j\ell}\}$, empty after job completion. Three primary writes recover frontier and competing start-time changes:

$$\begin{aligned} (q_j', R_j', A_m') = (\ell + 1, C_{j\ell}, C_{j\ell}), \qquad \mathcal{A}_k(s') = [\mathcal{A}_k(s) \setminus \mathcal{G}_j(\ell)] \cup \mathcal{G}_j(\ell + 1), \\ S_b = \max(R_i, A_u), \qquad S_b' - S_b = \mathbf{1}_{\{u=m\}}[\max(R_i, C_{j\ell}) - \max(R_i, A_m)] \geq 0. \end{aligned} \tag{C.5}$$

Unchanged primary fields are shared; derived candidate effects follow from the writes.

Physical units and neural features. For stored processing alternatives $\mathcal{P}_I$, maximum setup $s_{\max}$, and absent release/due-date maxima set to zero, normalization is

$$\begin{aligned} T_I = \max\left\{ \sum_{p \in \mathcal{P}_I} p + n s_{\max} \mathbf{1}_{\{k=\text{UPMSP}\}}, d_{\max}, r_{\max}, 1 \right\}, \\ \widetilde{x} = x / T_I, \qquad \widetilde{w}_j = w_j / \max\{\max_i w_i, 1\}, \\ \widetilde{C}_{\max} = C_{\max} / T_I, \qquad \widetilde{C}_{\text{tard}} = C_{\text{tard}} / \max\{T_I \textstyle\sum_j w_j, 1\}. \end{aligned} \tag{C.6}$$

Reconstruction uses original units; feature normalization and reward scaling are separate.

C.2 Scheduling Transitions and Terminal-Consistent Bounds

Assume nonpreemption, positive durations, nonnegative weights/setups, nonempty eligibility, and append-only placement. Let $\mathcal{U}, A_m, q_j, R_j, y$ denote uncommitted jobs, machine tails, next operations, readiness, and incurred tardiness. There are M machines; unlisted fields stay fixed.

Job-level transitions and bounds. Actions remove j from $\mathcal{U}$. SMSP has tail e and no releases; UPMSP retains last job b_m for setup; PFSP propagates machine tails f_r:

$$\begin{gathered}\text{SMSP}: \quad C_j = e + p_j, \quad e' = C_j; \\ \text{UPMSP}: \quad C_j = \max(A_m, r_j) + s_{b_m j m} + p_{jm}, \\ \text{UPMSP}: \quad (A'_m, b'_m) = (C_j, j); \qquad \text{SMSP/UPMSP}: \quad y' = y + w_j (C_j - d_j)_+; \\ \text{PFSP}: \quad g_0 = 0, \quad g_r = \max(f_r, g_{r-1}) + p_{jr}, \quad f'_r = g_r \ (r = 1, \dots, M).\end{gathered} \tag{C.7}$$

For any SGS continuation τ, earliest-completion and machine-load bounds give

$$\begin{gathered}C_i^\tau \geq L_i, \qquad L_i^{\mathrm{S}} = e + p_i, \quad L_i^{\mathrm{U}} = \min_{m \in \mathcal{M}_i} \{\max(A_m, r_i) + p_{im}\}, \\ C(\tau) \geq y + \textstyle\sum_{i \in \mathcal{U}} w_i (L_i - d_i)_+ = \Phi_{\mathrm{S}} \text{ or } \Phi_{\mathrm{U}}, \\ \text{PFSP}: \quad C_{\max}^\tau \geq F_r^\tau \geq f_r + \textstyle\sum_{i \in \mathcal{U}} p_{ir} \\ \Longrightarrow \quad \Phi_{\mathrm{P}} := \max_r (f_r + \textstyle\sum_{i \in \mathcal{U}} p_{ir}) \leq C_{\max}^\tau.\end{gathered} \tag{C.8}$$

UPMSP omits remaining setups because later predecessors may reduce them.

Operation-level commitments. FFSP uses stage eligibility $\mathcal{M}_{j\ell}$; JSSP fixes $\mathcal{M}_{j\ell} = \{\mu_{j\ell}\}$; FJSP permits operation-specific alternatives. All three satisfy

$$\begin{gathered}\mathcal{A}_k(s) = \{(j, q_j, m) : q_j \leq L_j, \ m \in \mathcal{M}_{j,q_j}\}, \\ a = (j, \ell, m): \quad S_{j\ell} = \max(R_j, A_m), \quad C_{j\ell} = S_{j\ell} + p_{j\ell m}, \\ (q'_j, R'_j, A'_m) = (\ell + 1, C_{j\ell}, C_{j\ell}), \qquad \underline{p}_{j\ell} = \min_{m \in \mathcal{M}_{j\ell}} p_{j\ell m}, \\ C_{jL_j}^\tau \geq R_j + \textstyle\sum_{\ell=q_j}^{L_j} \underline{p}_{j\ell}, \qquad F_m^\tau \geq A_m + W_m^\tau, \\ M C_{\max}^\tau \geq \textstyle\sum_m F_m^\tau \geq \sum_m A_m + \sum_j \sum_{\ell=q_j}^{L_j} \underline{p}_{j\ell}.\end{gathered} \tag{C.9}$$

Here W_m^τ is remaining load and F_m^τ the final tail. Hence

$$\begin{gathered}L_j^{\mathrm{rem}} = \textstyle\sum_{\ell=q_j}^{L_j} \underline{p}_{j\ell}, \qquad \Phi_{\mathrm{chain}} = \max\{\max_m A_m, \ \max_j (R_j + L_j^{\mathrm{rem}})\}, \\ \Phi_{\mathrm{FFSP}} = \Phi_{\mathrm{chain}}, \qquad \Phi_{\mathrm{JSSP}} = \max\big\{\Phi_{\mathrm{chain}}, \ \max_m (A_m + \textstyle\sum_{j,\, q_j \leq \ell \leq L_j : \mu_{j\ell} = m} p_{j\ell})\big\}, \\ \Phi_{\mathrm{FJSP}} = \max\big\{\Phi_{\mathrm{chain}}, \ M^{-1}(\textstyle\sum_m A_m + \sum_j L_j^{\mathrm{rem}})\big\}, \\ \Phi_k(s) \leq \min_{\tau \in \mathcal{T}_k(s)} C_k(\tau), \qquad \Phi_k(s_H) = C_k(\tau), \qquad N_{\mathrm{rem}}(T_k(s,a)) = N_{\mathrm{rem}}(s) - 1.\end{gathered} \tag{C.10}$$

Here $N_{\mathrm{rem}} = |\mathcal{U}|$ or $\sum_j (L_j - q_j + 1)_+$; terminal residual sums vanish.

Monotonicity under a scheduling commitment. Set $Q_r = f_r + \sum_{i \in \mathcal{U}} p_{ir}$, $\chi_j = R_j + L_j^{\mathrm{rem}}$, $\Lambda = M^{-1}(\sum_m A_m + \sum_j L_j^{\mathrm{rem}})$; Ω_m is the JSSP load bound:

$$\begin{gathered}Q'_r - Q_r = g_r - f_r - p_{jr} = (g_{r-1} - f_r)_+ \geq 0, \\ \chi'_j - \chi_j = (A_m - R_j)_+ + p_{j\ell m} - \underline{p}_{j\ell} \geq 0, \\ M(\Lambda' - \Lambda) = (R_j - A_m)_+ + p_{j\ell m} - \underline{p}_{j\ell} \geq 0, \quad \Omega'_m - \Omega_m = (R_j - A_m)_+ \geq 0, \\ \Phi_k(s') \geq \Phi_k(s) \quad \Longrightarrow \quad r_t = \Phi_k(s_t) - \Phi_k(s_{t+1}) \leq 0.\end{gathered} \tag{C.11}$$

For SMSP/UPMSP, remaining completion bounds cannot decrease and the committed completion dominates its old bound, giving the same inequality for nonnegative weights.

C.3 Encoder Symmetry and Recurrent Memory Derivations

From unordered edits to equivariant candidate scores. Let σ permute edits, Π permute candidates, and B contain relation biases and feasible-key masks. Shared projections and rowwise softmax yield

$$\begin{aligned} d(\sigma\mathcal{E}_a) &= \rho\left(\sum_{e\in\mathcal{E}_a} h_e, \max_{e\in\mathcal{E}_a} h_e, \log(1+|\mathcal{E}_a|)\right) = d(\mathcal{E}_a), \\ X' = \Pi X, \quad B' &= \Pi B \Pi^\top, \quad (Q', K', V') = (\Pi Q, \Pi K, \Pi V), \\ \mathrm{softmax}_{\mathrm{row}}\left(Q'K'^\top/\sqrt{d_h} + B'\right) &= \Pi\,\mathrm{softmax}_{\mathrm{row}}\left(QK^\top/\sqrt{d_h} + B\right)\Pi^\top, \\ \mathrm{Attn}(X', B') &= \Pi\,\mathrm{Attn}(X, B), \quad \ell' = \Pi\ell, \quad \pi' = \Pi\pi, \quad V'_\omega = V_\omega. \end{aligned} \tag{C.12}$$

Masks and relations permute jointly, padded queries are zeroed, and dropout is zero or coupled. No position features are used; greedy ties need invariant resolution.

Relation-conditioned competition. Fix token features and valid keys. Let $e_{ab} = q_a^\top k_b/\sqrt{d_h}$; A^0 uses the same content scores with zero relation bias. For valid b, c,

$$\begin{aligned} A_{ab} &= \frac{\exp(e_{ab} + w_B^\top r_{ab})}{\sum_c \exp(e_{ac} + w_B^\top r_{ac})}, \qquad o_a = \sum_b A_{ab} V_b, \\ \log\frac{A_{ab}}{A_{ac}} - \log\frac{A^0_{ab}}{A^0_{ac}} &= w_B^\top (r_{ab} - r_{ac}), \\ \nabla_{w_B} \log A_{ab} = r_{ab} - \sum_c A_{ac} r_{ac}, &\qquad \frac{\partial o_a}{\partial w_B} = \sum_b A_{ab}(V_b - o_a) r_{ab}^\top. \end{aligned} \tag{C.13}$$

Gated correction and chronological memory. Fix a head $S \in \mathbb{R}^{d_v \times d_k}$, unit key k_t, value v_t, and gates; $\widetilde{S}_{t-1} = \alpha_t S_{t-1}$. Equation 10 is a quadratic correction:

$$\begin{aligned} \mathcal{E}_t(S) = \tfrac{1}{2}\|Sk_t - v_t\|_2^2, &\qquad \nabla_S \mathcal{E}_t(S) = (Sk_t - v_t)k_t^\top, \\ S_t = \widetilde{S}_{t-1} - \beta_t \nabla_S \mathcal{E}_t(\widetilde{S}_{t-1}) &= \alpha_t S_{t-1}(I - \beta_t k_t k_t^\top) + \beta_t v_t k_t^\top, \\ S_t k_t - v_t = (1-\beta_t)(\widetilde{S}_{t-1}k_t - v_t), &\qquad \mathcal{E}_t(S_t) = (1-\beta_t)^2 \mathcal{E}_t(\widetilde{S}_{t-1}), \\ k_t^\top u = 0 \quad \Longrightarrow \quad S_t u &= \alpha_t S_{t-1} u. \end{aligned} \tag{C.14}$$

Define $F_t = \alpha_t(I - \beta_t k_t k_t^\top)$, $U_t = \beta_t v_t k_t^\top$, and $F_{i:j} = F_i \cdots F_j$; empty products equal I. Unrolling the forward operator gives

$$\begin{aligned} S_t = S_{t-1}F_t + U_t &= S_{-1}F_{0:t} + \sum_{i=0}^{t} U_i F_{i+1:t}, \\ o_t = S_t q_t &= S_{-1}F_{0:t}q_t + \sum_{i=0}^{t} \beta_i v_i (k_i^\top F_{i+1:t} q_t), \\ z_t = W_z[u_t, \widehat{d}_{t-1}, \eta(t/H)], &\qquad \widehat{d}_t = d_t(a_t). \end{aligned} \tag{C.15}$$

Only the selected transition supplies the next context write.

Propagation of context-write perturbations. For common unit keys and gates $0 \le \alpha_t, \beta_t \le 1$, set $D_t = S_t - S'_t$ and $\delta v_t = v_t - v'_t$. Substitution and submultiplicativity give

$$\begin{aligned} \|F_t\|_2 \le \alpha_t, \qquad D_t = D_{t-1}F_t + \beta_t \delta v_t k_t^\top &= D_{-1}F_{0:t} + \sum_{i=0}^{t} \beta_i \delta v_i k_i^\top F_{i+1:t}, \\ \|D_t\|_F \le \left(\prod_{j=0}^{t} \alpha_j\right)\|D_{-1}\|_F + \sum_{i=0}^{t} \beta_i \|\delta v_i\|_2 \prod_{j=i+1}^{t} \alpha_j, \\ q_t = q'_t \quad \Longrightarrow \quad \|o_t - o'_t\|_2 \le \|D_t\|_F \|q_t\|_2. \end{aligned} \tag{C.16}$$

The bound separates initialization and write errors conditional on common keys/gates.

C.4 Scheduling Returns, Policy Optimization, and Decoding

For a policy-independent potential with $\Phi(s_H) = C(\tau)$ and $\gamma = 1$, Equation 16 telescopes to scheduling cost and exact advantage:

$$\begin{gathered} G_t = \sum_{j=t}^{H-1} [\Phi(s_j) - \Phi(s_{j+1})] = \Phi(s_t) - C(\tau), \\ \mathbb{E}[G_0] = \mathbb{E}_I[\Phi(s_0(I))] - \mathbb{E}[C(\tau)] \implies \arg\max_\theta \mathbb{E}[G_0] = \arg\min_\theta \mathbb{E}[C(\tau)], \\ V^\pi(\mathcal{H}_t) = \Phi(s_t) - \mathcal{C}_t^\pi, \qquad Q^\pi(\mathcal{H}_t, a) = \Phi(s_t) - \mathcal{C}_t^\pi(a), \\ A^\pi(\mathcal{H}_t, a) = Q^\pi(\mathcal{H}_t, a) - V^\pi(\mathcal{H}_t) = \mathcal{C}_t^\pi - \mathcal{C}_t^\pi(a). \end{gathered} \tag{C.17}$$

Here $\mathcal{C}_t^\pi = \mathbb{E}[C \mid \mathcal{H}_t]$ and $\mathcal{C}_t^\pi(a) = \mathbb{E}[C \mid \mathcal{H}_t, a_t = a]$. Instance-dependent reward scaling reweights the objective.

From trajectories to the cost gradient. For differentiable positive policies, fixed feasible sets, and action-independent b_t,

$$\begin{gathered} p_\theta(\tau \mid I) = \textstyle\prod_{t=0}^{H-1} \pi_\theta(a_t \mid \mathcal{H}_t), \qquad \nabla \log p_\theta(\tau \mid I) = \textstyle\sum_t \nabla \log \pi_\theta(a_t \mid \mathcal{H}_t), \\ \mathbb{E}[b_t \nabla \log \pi_\theta(a_t \mid \mathcal{H}_t) \mid \mathcal{H}_t] = b_t \textstyle\sum_{a \in \mathcal{A}(s_t)} \nabla \pi_\theta(a \mid \mathcal{H}_t) = 0, \\ \nabla J(\theta) = \mathbb{E}[\textstyle\sum_t \nabla \log \pi_\theta(a_t \mid \mathcal{H}_t) G_t] \\ = \mathbb{E}[\textstyle\sum_t \nabla \log \pi_\theta(a_t \mid \mathcal{H}_t)(\mathcal{C}_t^\pi - \mathcal{C}_t^\pi(a_t))], \quad J = \mathbb{E}[G_0]. \end{gathered} \tag{C.18}$$

Differentiation and expectation are assumed interchangeable.

Finite-horizon estimation and the clipped surrogate. Set $m_t = 1 - \mathbf{1}_{\text{terminal}}$, $V_H^{\text{old}} = 0$, and $T = H - t$. GAE (Schulman et al., 2016) uses complete rollouts; PPO (Schulman et al., 2017) replays $\tilde{\pi}_{\theta,b}$ from a fixed behavior boundary for $b > 0$ or the current reset for $b = 0$:

$$\begin{gathered} \delta_t^V = r_t + m_t V_{t+1}^{\text{old}} - V_t^{\text{old}}, \quad \widehat{A}_t = \delta_t^V + \lambda m_t \widehat{A}_{t+1}, \quad \widehat{R}_t = \widehat{A}_t + V_t^{\text{old}}, \\ \widehat{A}_t = \textstyle\sum_{j=0}^{T-1} \lambda^j (r_{t+j} + V_{t+j+1}^{\text{old}} - V_{t+j}^{\text{old}}) \\ = -V_t^{\text{old}} + \textstyle\sum_{j=0}^{T-1} \lambda^j r_{t+j} + (1-\lambda) \textstyle\sum_{j=1}^{T-1} \lambda^{j-1} V_{t+j}^{\text{old}}, \\ \varrho_t = \frac{\tilde{\pi}_{\theta,b}(a_t)}{\pi_{\text{old}}(a_t \mid \mathcal{H}_t)}, \qquad \ell_t = \min\big(\varrho_t \bar{A}_t, \text{clip}(\varrho_t, 1-\epsilon, 1+\epsilon)\bar{A}_t\big), \\ \mathcal{L} = -\mathbb{E}_t[\ell_t] + \tfrac{c_V}{2} \mathbb{E}_t[(\widetilde{V}_{\omega,b} - \widehat{R}_t)^2] - c_{\text{ent}} \mathbb{E}_t[\text{Ent}(\tilde{\pi}_{\theta,b})]. \end{gathered} \tag{C.19}$$

Here $\bar{A}_t = (\widehat{A}_t - \mu_A)/\max(\sigma_A, 10^{-8})$ uses valid rollout steps; targets and behavior probabilities are fixed. Values and entropy use the same replay; $\lambda = 1$ gives $\widehat{R}_t = G_t$.

The gradient removed by boundary detachment. For memory/feedback $c_t = f_\theta(c_{t-1}, x_t)$ along a common trajectory, let $J_t = \partial f_\theta / \partial c_{t-1}$, $U_t = \partial f_\theta / \partial \theta$, and segment start $b > 0$:

$$\begin{gathered} \Gamma_t := \tfrac{dc_t}{d\theta} = J_t \Gamma_{t-1} + U_t, \qquad \Gamma_t = (J_t \cdots J_b)\Gamma_{b-1} + \textstyle\sum_{i=b}^t (J_t \cdots J_{i+1}) U_i, \\ \Gamma_t^{\text{trunc}} = \textstyle\sum_{i=b}^t (J_t \cdots J_{i+1}) U_i, \qquad \Gamma_t - \Gamma_t^{\text{trunc}} = (J_t \cdots J_b)\Gamma_{b-1}, \\ \nabla_\theta \log \pi_t - \nabla_\theta^{\text{trunc}} \log \pi_t = \Gamma_{b-1}^\top (J_t \cdots J_b)^\top \nabla_{c_t} \log \pi_t. \end{gathered} \tag{C.20}$$

Empty products are identities. Algorithm S4 additionally uses cached behavior-policy boundary context. After parameter updates, this cached context may differ from a full prefix replay under the current parameters; Equation C.20 describes gradient truncation and does not bound this additional forward-context discrepancy.

Expected gains from policy sampling. For iid nonnegative integrable costs at a fixed instance/checkpoint, let $F_i(c) = \Pr(C_{i,b} \le c)$ and $C_{i,\text{best}}^{(B)} = \min_{b \le B} C_{i,b}$. Independence gives

$$\begin{gathered} \Pr(C_{i,\text{best}}^{(B)} > c) = \textstyle\prod_{b=1}^B \Pr(C_{i,b} > c) = [1 - F_i(c)]^B, \\ \mathbb{E}[C_{i,\text{best}}^{(B)}] - \mathbb{E}[C_{i,\text{best}}^{(B+1)}] = \textstyle\int_0^\infty F_i(c)[1 - F_i(c)]^B \, dc \ge 0. \end{gathered} \tag{C.21}$$

The decreasing integrand proves diminishing expected gains between iid sampling budgets.

C.5 Computational Accounting and Evaluation Estimators

End-to-end work and persistent storage. Let n_t, m_t, A_t, D_t count entities, relations, candidates, and edits; d is width, L_G, L_L are depths, B is active trajectories, h_G is global heads per layer, and b_{scalar} is bytes per scalar.

$$\begin{gathered} C_t^{\text{neural}} = O\big((n_t + m_t + D_t + A_t)d^2 + L_G d^2 + L_L(A_t d^2 + A_t^2 d)\big), \\ C_{\text{episode}} = \sum_{t=0}^{H-1} (C_t^{\text{adapt}} + C_t^{\text{compile}} + C_t^{\text{neural}} + C_t^{\text{execute}}), \\ N_{\text{state}} = BL_G h_G d_v d_k, \quad M_{\text{state}}^{\text{KiB}} = b_{\text{scalar}} N_{\text{state}}/2^{10}, \\ M_{\text{state}}^{\text{KiB}}\big|_{B=1} = 4(3 \cdot 4 \cdot 128 \cdot 64)/2^{10} = 384. \end{gathered} \tag{C.22}$$

For $K_t > 0$ fixed-schema primary fields and D_t^P edits, shared-base logical counts give

$$\begin{gathered} W_{\text{full}} = A_t K_t, \quad W_{\text{sparse}} = K_t + D_t^P, \quad \frac{W_{\text{sparse}}}{W_{\text{full}}} = \frac{1}{A_t} + \frac{\bar{e}_t^P}{K_t}, \quad \bar{e}_t^P = D_t^P / A_t, \\ W_{\text{sparse}} < W_{\text{full}} \iff \bar{e}_t^P < K_t(1 - A_t^{-1}) \qquad (A_t > 0). \end{gathered} \tag{C.23}$$

Logical counts exclude metadata and derived relations and do not imply wall-clock speedup; local attention remains quadratic. Memory above counts recurrent state only.

Latency and decoding budget. With resident parameters and synchronized boundaries, $T_{v,i,r}$ times repeat r on instance i for variant v; B_{dec} is the sampling budget.

$$\begin{gathered} T_{\text{wall}} = t_{\text{stop}} - t_{\text{start}}, \qquad T_{\text{sched}} = T_{\text{init}} + T_{\text{overhead}} + \textstyle\sum_t \sum_{x \in \mathcal{X}} T_{x,t}, \\ \mathcal{X} = \{\text{adapt}, \text{compile}, \text{encode}, \text{choose}, \text{execute}\}, \\ C_{i,\text{best}}^{(B_{\text{dec}})} = \min_{b \le B_{\text{dec}}} C_{i,b}, \qquad T_{i,\text{amort}}^{(B_{\text{dec}})} = T_{i,\text{deliver}}^{(B_{\text{dec}})} / B_{\text{dec}}, \\ \widehat{T}_{v,i} = R_i^{-1} \textstyle\sum_r T_{v,i,r}, \quad \widehat{T}_v = N^{-1} \textstyle\sum_i \widehat{T}_{v,i}, \quad \rho(u,v) = \frac{\sum_i \widehat{T}_{u,i}}{\sum_i \widehat{T}_{v,i}}. \end{gathered} \tag{C.24}$$

Stage times assume serial work; amortized time differs from delivery latency.

Gap aggregation and paired seed uncertainty. Use positive references C_i^{ref}, a fixed benchmark, and $S > 1$ iid training-seed pairs. Let $d_{s,i} = g_{v,s,i} - g_{\text{Full},s,i}$.

$$\begin{gathered} g_i = 100(C_i - C_i^{\text{ref}})/C_i^{\text{ref}}, \quad \bar{g} = N^{-1} \textstyle\sum_i g_i, \quad g_{\text{ratio}} = \frac{\sum_i C_i^{\text{ref}} g_i}{\sum_i C_i^{\text{ref}}}, \\ g_{\text{ratio}} - \bar{g} = \text{Cov}_N(C^{\text{ref}}, g)/\bar{C}^{\text{ref}}, \qquad \delta_s = N^{-1} \textstyle\sum_i d_{s,i}, \quad \widehat{\delta} = S^{-1} \textstyle\sum_s \delta_s, \\ \text{Var}(\widehat{\delta}) = \frac{\text{Var}(\delta_1)}{S} = \frac{1}{SN^2} \textstyle\sum_{i,j} \text{Cov}(d_{1,i}, d_{1,j}), \\ \widehat{\sigma}_{\text{seed}}^2 = \frac{\sum_s (\delta_s - \widehat{\delta})^2}{S-1}, \qquad \widehat{\text{SE}} = \frac{\widehat{\sigma}_{\text{seed}}}{\sqrt{S}}, \quad \mathcal{I}_{1-\alpha} = \widehat{\delta} \pm t_{S-1,1-\alpha/2} \widehat{\text{SE}}. \end{gathered} \tag{C.25}$$

Cov_N uses denominator N; differences are percentage points. The interval is exact for normal seed effects and otherwise approximate.

Matched-state attention diagnostics. Average row-normalized arrays over layers/heads $\mathcal{L}, \mathcal{H}$ for A valid candidates; g maps candidates to nonempty disjoint groups G_c.

$$\begin{gathered} \bar{W}_{ab} = \frac{\sum_{\ell \in \mathcal{L}, h \in \mathcal{H}} W_{ab}^{\ell,h}}{|\mathcal{L}||\mathcal{H}|}, \quad \textstyle\sum_b \bar{W}_{ab} = 1, \quad E_{ab} = A\bar{W}_{ab}, \\ M_a^{\text{within}} = \textstyle\sum_{b: g(b) = g(a)} \bar{W}_{ab}, \quad Q_a^{\text{within}} = AM_a^{\text{within}}/|G_{g(a)}|, \quad M_a^{\text{cross}} = 1 - M_a^{\text{within}}, \\ H_a = -\textstyle\sum_b \bar{W}_{ab} \log \bar{W}_{ab}, \qquad D_{\text{KL}}(\bar{W}_{a,:} \| \text{Unif}_A) = \log A - H_a \ge 0. \end{gathered} \tag{C.26}$$

Use $0 \log 0 = 0$ and include self-weight; uniform attention gives $E = Q = 1$.

D EXPERIMENTAL SETTINGS

D.1 SYNTHETIC INSTANCES AND SPLIT BOUNDARIES

Table S5 specifies fixed validation counts at the training size and test counts at each of three sizes. Training uses fresh instances; validation/test pools are disjoint from training, and public benchmarks are excluded from selection. Sizes $n \times m$ denote jobs and machines.

Table S5: Synthetic generators and configured validation/test counts; N_{test} is per size.

Task	N_{val}	N_{test}	Generation recipe
SMSP	100	100	$p_j, w_j \sim U[0,1)$ and $d_j \sim nU[0,1)$.
UPMSP	100	500	$p_{jm} \sim U_{\mathbb{Z}}[1,99]$, $w_j \sim U[0,1)$, and setups $\sim U_{\mathbb{Z}}[0,9]$; eligibility/release/due rules below.
PFSP	100	1,000	$p_{jm} \sim \text{Gamma}(\text{shape} = 1, \text{rate} = 2)$, stored in float32. No additional objective rescaling is applied.
FFSP	100	1,000	Three stages, four machines/stage; independent integer durations in $[2,9]$.
JSSP	100	100	Independent machine permutations per job; integer durations in $[1,99]$, inclusive.
FJSP	100	100	SD1-style: operations/job $\sim U_{\mathbb{Z}}[\lfloor 0.8m \rfloor, \lfloor 1.2m \rfloor]$, eligible machines/operation $\sim U_{\mathbb{Z}}[1,m]$, operation mean duration $\sim U_{\mathbb{Z}}[1,20]$, with eligible-machine durations sampled within $\pm 20\%$ of the operation mean.

UPMSP parameterization. Each machine uniformly selects $\max(1, \lfloor n/2 \rfloor)$ jobs without replacement; uncovered jobs receive one uniformly drawn machine. Tightness/range pairs $(T, R) \in \{0.2, 0.4, 0.6, 0.8, 1.0\}^2$ are balanced. For inclusive integer-uniform $U_{\mathbb{Z}}$,

$$\widehat{p} = \frac{1}{nm} \sum_{j=1}^{n} \sum_{u=1}^{m} p_{ju}, \qquad r_j \sim U_{\mathbb{Z}}[0, \lfloor \widehat{p}/2 \rfloor], \quad d_j - r_j \sim U_{\mathbb{Z}}[L_j, U_j],$$

$$L_j = \lfloor (\widehat{p} - r_j)(1 - T - R/2) \rfloor, \qquad U_j = \lfloor (\widehat{p} - r_j)(1 - T + R/2) \rfloor.$$

The mean includes ineligible entries; negative due offsets and dummy-predecessor setups are retained. Each (T, R) cell has four validation and 20 test instances per size; training cycles by index modulo 25. Draw order is processing times, eligibility/repair, weights, releases, due dates, and setups. This reproduces the distribution, not the source papers' fixed instances.

Additional evaluation configurations. Each Figure S4 setting uses 500 UPMSP, 1,000 PFSP/FFSP, or 100 JSSP/FJSP instances, as in Table S5. All four representation–encoder combinations share the instance pool within each setting.

D.2 TRAINING CONFIGURATION

Independent task models use the architecture in Appendix B.3 and one NVIDIA RTX 4090. PPO uses four epochs/update, $\epsilon = 0.2$, $\gamma = 1$, $\lambda = 0.98$, AdamW $10^{-4} \to 10^{-5}$, gradient clipping 0.5, and 64-step segments. All 36 CPU implementation tests passed under Python 3.10.9, PyTorch 2.2.2, and pytest 8.4.2.

Table S6: Optimization and training configuration.

Setting	Value	Setting	Value
Value / entropy coefficient	0.5 / 0.01	Advantage normalization	Enabled
Training seeds	0–4	PPO updates	400
Trajectories / update	50; JSSP: 5	Segments / minibatch	8
Validation	Every 25 updates	Validation decoding	Greedy
Training size	SMSP 50; UPMSP 50×3; PFSP 20×5; FFSP 20×12; JSSP 30×20; FJSP 10×5		

Checkpoint selection. Every 25 updates, greedy evaluation on the full validation pool selects the checkpoint with lowest mean raw objective among 16 evaluations; ties favor earlier updates. Both decoding modes reuse this checkpoint across all synthetic test sizes without retraining or fine-tuning.

D.3 Public Benchmarks and Literature Comparisons

Frozen synthetic-trained models use the public groups in Table S7 (Taillard, 1993; Brandimarte, 1993; Hurink et al., 1994).

Table S7: Public benchmark groups, instance counts, and TADS schedules sampled per instance. PFSP/JSSP use Taillard groups.

Benchmark group	Instances	Samples
PFSP 20×5	10	128
PFSP 50×5	10	128
PFSP 100×5	10	128
JSSP 30×20	10	512
JSSP 50×20	10	512
JSSP 100×20	10	512

FJSP benchmark	Instances	Samples
Brandimarte Mk1–Mk10	10	100
Hurink edata	40	100
Hurink rdata	40	100
Hurink vdata	40	100

FJSP follows the DANIEL evaluation subset: Brandimarte Mk1–Mk10 and 40 instances per Hurink group. Greedy and sampling use identical instances and reference objectives.

Literature baselines retain source hardware and budgets. In Table 1, L2D uses its 30×20 checkpoint at all JSSP sizes; the OR-Tools reference budgets are 3600 s for JSSP and 1800 s for FJSP.

D.4 Controlled Variants and Comparison Factors

Removing TC substitutes original-action features for scoring and feedback; removing LCA/GRM removes local comparisons/persistent memory, and removing DS removes both. The scheduling interface is shared. Cost comparisons cross FS/TC with FRA/DS and include preprocessing and candidate scoring.

Table S8: Representation–encoder combinations for computational-cost comparisons.

Configuration	Candidate representation	Contextual encoding
FS + FRA	Full successor	Full-residual attention
TC + FRA	Typed transition edits	Full-residual attention
FS + DS	Full successor	Global memory + local attention
TC + DS	Typed transition edits	Global memory + local attention

D.5 Aggregation, Uncertainty, and Time Units

Gap averages instance-level differences against task references. Table 1 Sampling uses 100 complete schedules per instance; results are averaged over seeds 0–4 unless reported seed-by-seed. Table S14 reports paired w/o TC minus Full TADS means for seeds 0–4, with Mean/SD across these five pairs; decoding repetitions are separate (Appendix C.5). Table 1 uses total test-set minutes for SMSP/UPMSP/PFSP and seconds/instance for FFSP/JSSP/FJSP. Figure 5 uses seconds/instance for UPMSP/FJSP and milliseconds/instance for PFSP.

For Figure S3, colored points show each test instance's Gap averaged over seeds 0–4. Black markers show the test-instance mean; vertical bars are 95% bootstrap confidence intervals from 10,000 instance-level resamples. Insets show paired mean differences with 95% paired-bootstrap confidence intervals using the same resampled instance indices.

D.6 Runtime Measurement Protocol

Table S9: Evaluator defaults with resident parameters; sampling includes best-schedule selection.

Setting	Value	Setting	Value
Greedy trajectories	1	Greedy / sampling batch	1 / 16
Timed repeats / instance	1	Dedicated warm-ups	0
Neural precision	Float32; no AMP	Gradients / dropout	Disabled / 0
Clock	perf_counter	CUDA synchronization	Both boundaries
Timed scope	Environment initialization, complete decoding, and best-schedule selection		
Excluded scope	Model loading and independent schedule verification		

E Supplementary Experiments and Analysis

E.1 Component Contributions and Attention Patterns

Table S10: Component contributions across instance sizes. Gap (%) and its difference Δ from Full TADS (percentage points) compare each removal with the full model within the same task and size.

SMSP	50		100		500	
Variant	Gap (%)	Δ	Gap (%)	Δ	Gap (%)	Δ
w/o TC	0.42%	+0.36	0.91%	+0.73	2.34%	+1.16
w/o LCA	0.26%	+0.20	0.53%	+0.35	1.70%	+0.52
w/o GRM	0.49%	+0.43	1.00%	+0.82	2.61%	+1.43
w/o DS	0.70%	+0.64	1.44%	+1.26	3.26%	+2.08
Full TADS	**0.06%**	**0.00**	**0.18%**	**0.00**	**1.18%**	**0.00**
UPMSP	50×3		50×6		100×6	
Variant	Gap (%)	Δ	Gap (%)	Δ	Gap (%)	Δ
w/o TC	6.02%	+0.92	7.55%	+1.07	8.81%	+1.33
w/o LCA	6.19%	+1.09	7.68%	+1.20	9.04%	+1.56
w/o GRM	5.74%	+0.64	7.23%	+0.75	8.39%	+0.91
w/o DS	6.64%	+1.54	8.14%	+1.66	9.61%	+2.13
Full TADS	**5.10%**	**0.00**	**6.48%**	**0.00**	**7.48%**	**0.00**
PFSP	20×5		50×5		100×5	
Variant	Gap (%)	Δ	Gap (%)	Δ	Gap (%)	Δ
w/o TC	-0.05%	+0.36	0.30%	+0.44	0.49%	+0.65
w/o LCA	-0.20%	+0.21	0.04%	+0.18	0.14%	+0.30
w/o GRM	-0.10%	+0.31	0.19%	+0.33	0.35%	+0.51
w/o DS	0.10%	+0.51	0.52%	+0.66	0.79%	+0.95
Full TADS	**-0.41%**	**0.00**	**-0.14%**	**0.00**	**-0.16%**	**0.00**
FFSP	20×12		50×12		100×12	
Variant	Gap (%)	Δ	Gap (%)	Δ	Gap (%)	Δ
w/o TC	6.17%	+0.37	2.81%	+0.28	2.51%	+0.24
w/o LCA	6.34%	+0.54	2.95%	+0.42	2.63%	+0.36
w/o GRM	5.90%	+0.10	2.58%	+0.05	2.16%	-0.11
w/o DS	6.75%	+0.95	3.34%	+0.81	3.05%	+0.78
Full TADS	**5.80%**	**0.00**	**2.53%**	**0.00**	**2.27%**	**0.00**
JSSP	30×20		50×20		100×20	
Variant	Gap (%)	Δ	Gap (%)	Δ	Gap (%)	Δ
w/o TC	23.27%	+0.76	14.88%	+0.95	3.34%	+1.12
w/o LCA	23.01%	+0.50	14.48%	+0.55	2.88%	+0.66
w/o GRM	23.43%	+0.92	15.07%	+1.14	3.51%	+1.29
w/o DS	23.89%	+1.38	15.63%	+1.70	4.11%	+1.89
Full TADS	**22.51%**	**0.00**	**13.93%**	**0.00**	**2.22%**	**0.00**
FJSP	10×5		20×10		40×10	
Variant	Gap (%)	Δ	Gap (%)	Δ	Gap (%)	Δ
w/o TC	10.31%	+0.47	0.61%	+0.50	1.11%	+0.72
w/o LCA	10.42%	+0.58	0.72%	+0.61	1.20%	+0.81
w/o GRM	10.19%	+0.35	0.48%	+0.37	0.90%	+0.51
w/o DS	10.75%	+0.91	1.02%	+0.91	1.60%	+1.21
Full TADS	**9.84%**	**0.00**	**0.11%**	**0.00**	**0.39%**	**0.00**

Protocol. Greedy decoding with shared data, five training seeds, and update budgets. Controls follow Appendix D.4. Full TADS appears last; negative Δ favors the ablated variant.

Components across instance sizes. Removing DS yields the largest Gap increase at every size (Table S10). LCA has the largest individual effect for UPMSP/FJSP, and GRM for SMSP/JSSP. FFSP 100×12 is an exception: removing GRM lowers Gap from 2.27% to 2.16%.

E.1.1 Attention Patterns across Formulations

Figure S2 compares three configurations at the same state and with the same ordered 24 candidates, averaging eight heads in local layer 3. Colors show $E_{ab} = A\bar{W}_{ab}$ from Equation C.26, with A valid candidates and no submatrix renormalization. White dashed boxes outline blocks in the displayed candidate ordering. The within-group and cross-group annotations refer to locations inside and outside these blocks; they do not establish that candidates share a job, stage, or machine.

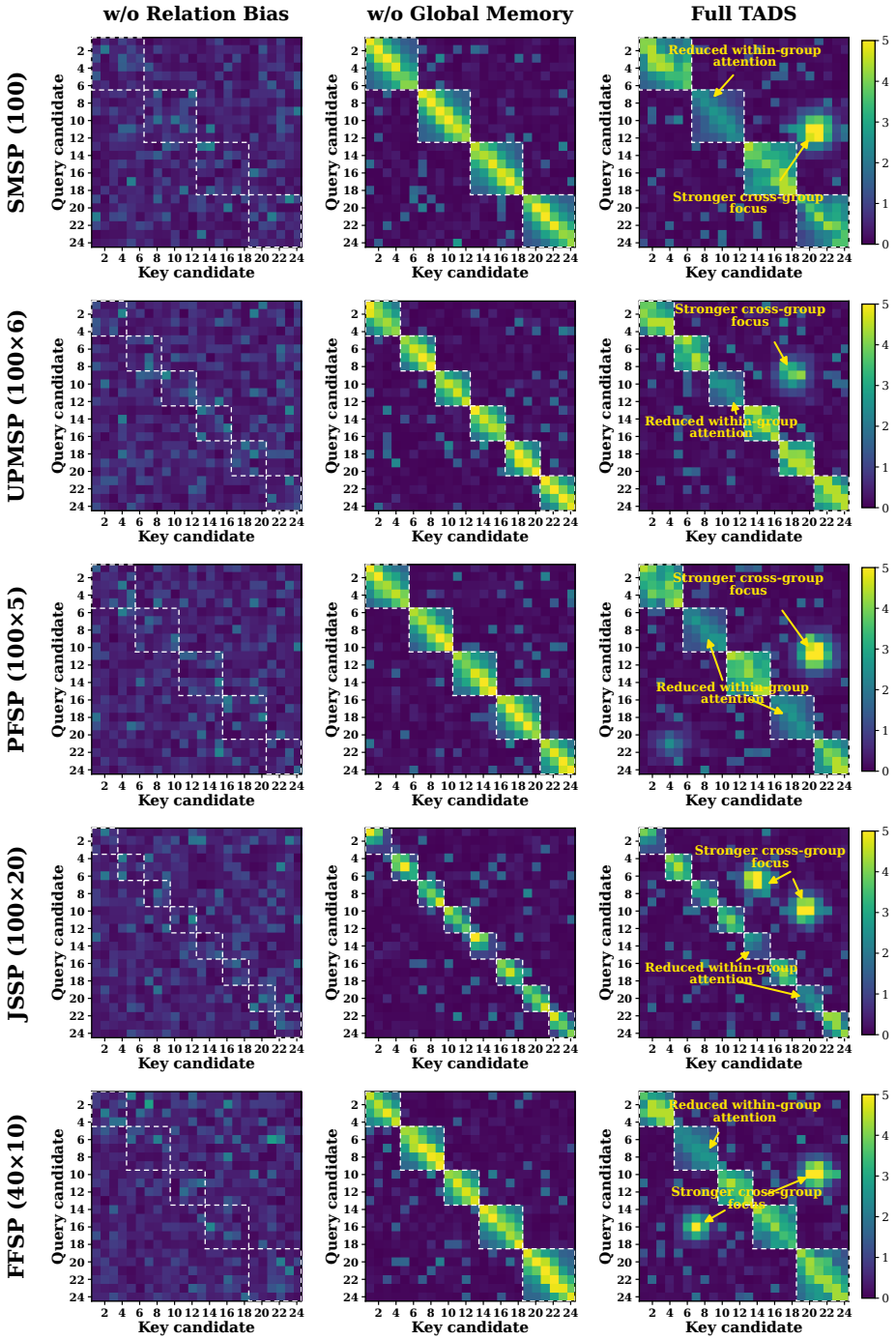


Figure S2: Attention patterns across five formulations. Columns compare w/o Relation Bias, w/o Global Memory, and Full TADS. Matrix rows index queries and columns index keys; colors show enrichment relative to uniform attention. White dashed boxes outline blocks in the displayed candidate ordering.

E.2 Candidate Representations and Transition Sparsity

Table S11: Transition sparsity and serialized payload across task sizes. The final column gives the full-to-sparse payload ratio computed from the reported KiB values.

Task	Size	Primary-field sparsity			Payload (KiB)		Payload ratio
		$\lvert K\rvert$	Mean $\lvert\Delta_P\rvert$	Edited (%)	Full	Sparse	Full/Sparse ($\times$)
SMSP	50	52	2.50	4.81%	1.234	0.094	13.13
	100	102	2.50	2.45%	2.406	0.094	25.60
	500	502	2.50	0.50%	11.781	0.094	125.33
UPMSP	50×3	57	3.50	6.14%	1.352	0.125	10.82
	50×6	63	3.50	5.56%	1.492	0.125	11.94
	100×6	113	3.50	3.10%	2.664	0.125	21.31
PFSP	20×5	25	6.00	24.00%	0.602	0.203	2.97
	50×5	55	6.00	10.91%	1.305	0.203	6.43
	100×5	105	6.00	5.71%	2.477	0.203	12.20
FFSP	20×12	52	3.00	5.77%	1.234	0.109	11.32
	50×12	112	3.00	2.68%	2.641	0.109	24.23
	100×12	212	3.00	1.42%	4.984	0.109	45.72
JSSP	30×20	80	3.00	3.75%	1.891	0.109	17.35
	50×20	120	3.00	2.50%	2.828	0.109	25.94
	100×20	220	3.00	1.36%	5.172	0.109	47.45
FJSP	10×5	25	3.00	12.00%	0.602	0.109	5.52
	20×10	50	3.00	6.00%	1.188	0.109	10.90
	40×10	90	3.00	3.33%	2.125	0.109	19.50

$K = \operatorname{dom} P \cup \operatorname{dom} P^a$ indexes mutable primary fields; immutable instance data are shared. Mean $\lvert\Delta_P\rvert$ counts changed primary fields, and Edited is their fraction of $\lvert K\rvert$. Full and Sparse denote successor and patch payloads in KiB (2^{10} bytes). Full/Sparse measures storage reduction and is computed from the rounded payloads shown; it is not a reconstruction success rate. The auxiliary schedule ledger is excluded from $\lvert K\rvert$, edit counts, and payload measurements.

Transition sparsity and storage. Table S11 quantifies changed fields and serialized payloads. Within each task, the mean edit count is constant across the three sizes, while primary-state size grows. For SMSP, the edited fraction falls from 4.81% to 0.50% and full payload grows from 1.234 to 11.781 KiB; sparse payload remains 0.094 KiB. The full-to-sparse payload ratio therefore rises from 13.13 to 125.33. These ratios quantify representation storage; exact reconstruction follows the conditions in Appendix C.1, with implementation checks reported in Appendix B.2.

E.3 Computational Efficiency and Scaling

Representation–architecture interaction. Table S12 crosses FS/TC with FRA/DS. TC reduces compilation time under both encoders, and DS reduces network time and peak memory under both representations. TC+DS has the lowest total latency and memory in all six evaluated configurations. Quality has a different ordering on FFSP, where TC+FRA achieves a slightly lower Gap. Separately profiled stage times need not sum to elapsed wall time.

E.4 Decoding Budget and Solution Quality

Returns from additional decoding. Table S13 shows lower Gap and longer delivered latency as the sampling budget increases for all six tasks. The gains diminish at larger budgets: increasing UPMSP from 128 to 512 samples lowers Gap by 0.11 percentage points while increasing time from 5.32 to 15.86 seconds. Time includes all sampled trajectories and best-schedule selection, without division by the sample count.

Solution-quality distributions. Figure S3 extends the OA/SC/FS/TC comparison to three sizes per task. Dots show per-instance Gap averaged over seeds 0–4; black markers and vertical bars report mean Gap and 95% bootstrap confidence intervals. Insets show paired mean differences with 95% paired-bootstrap intervals; negative values favor the first representation. Appendix D.5 defines the resampling procedure.

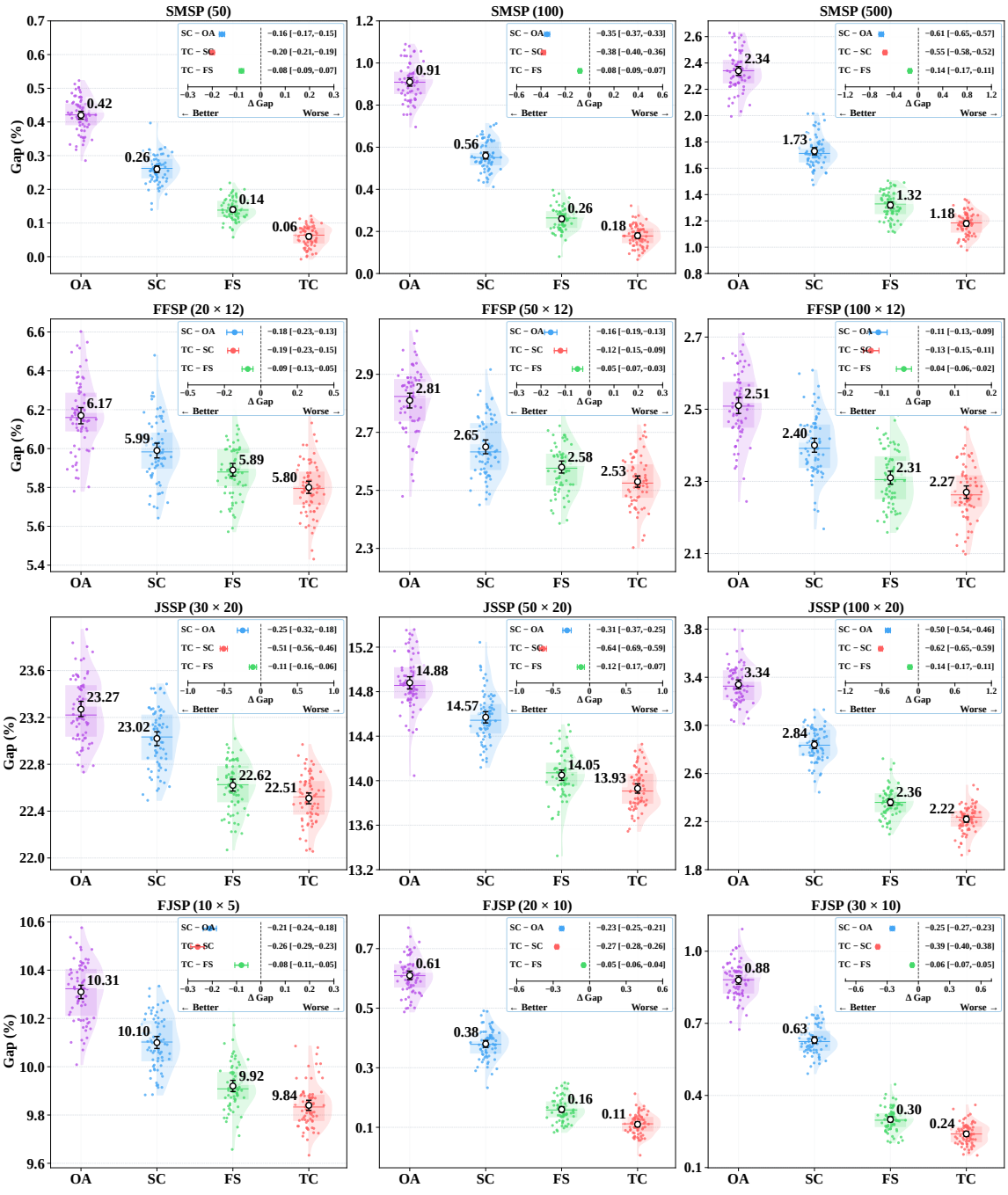


Figure S3: Gap distributions for SMSP, FFSP, JSSP, and FJSP; lower is better. Density envelopes and boxes summarize instance variability. Insets compare SC–OA, TC–SC, and TC–FS in percentage points. Task sizes label each panel; statistical definitions follow Appendix D.5.

Scaling with job and resource counts. Figure S4 varies jobs within panels and resources across columns: UPMSP uses 3/9/12 machines, PFSP/JSSP 10/15/20, FJSP 5/15/20, and FFSP 2/4/6 machines per stage. Plotted time grows with job count; TC+DS is below FS+FRA in these settings. PFSP uses milliseconds, other tasks seconds. The same four representation–encoder combinations are defined in Table S8; timing boundaries and evaluator settings are given in Appendix D.6.

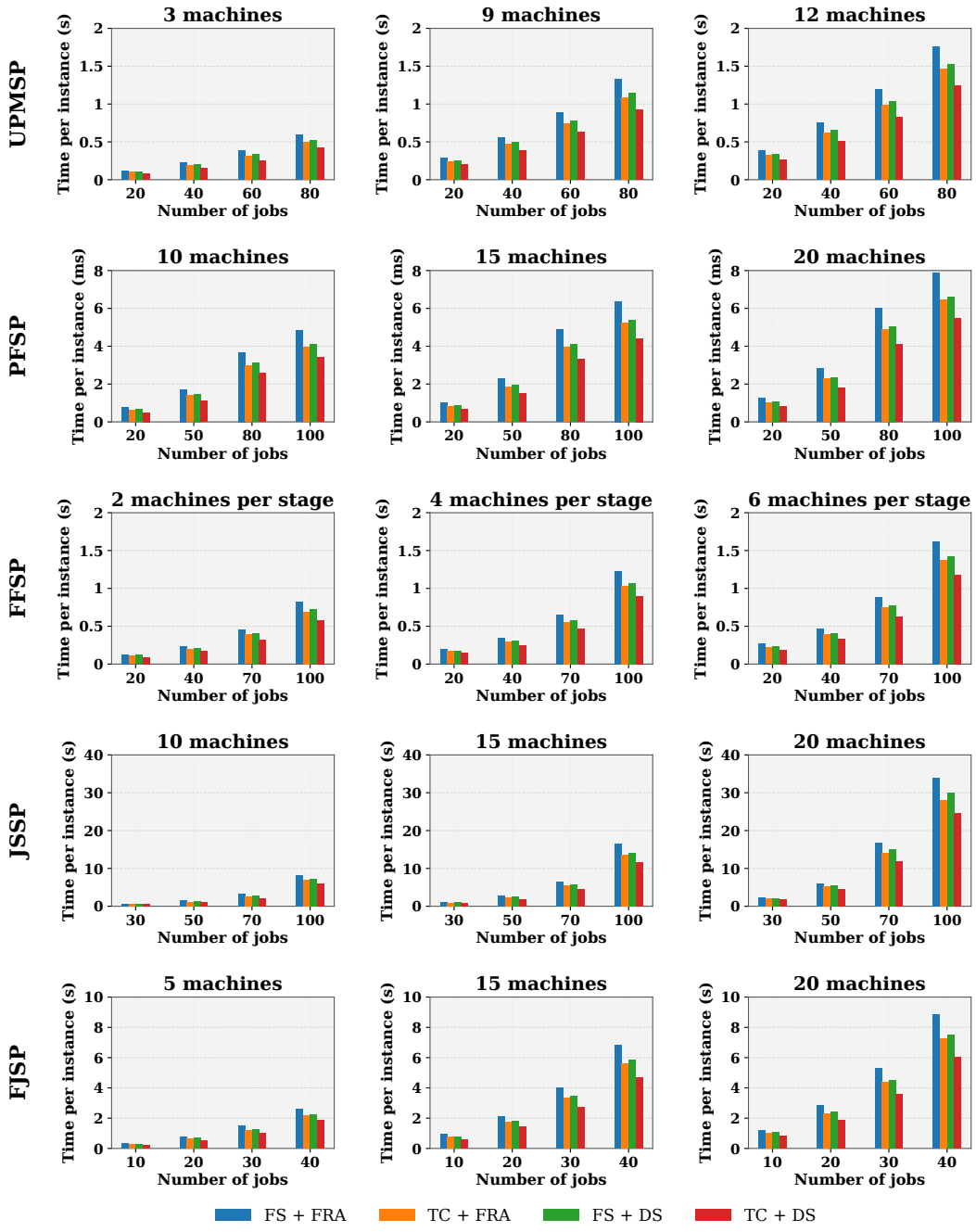


Figure S4: Schedule-construction time across job counts and resource configurations for UPMSP, PFSP, FFSP, JSSP, and FJSP. Bars compare FS+FRA, TC+FRA, FS+DS, and TC+DS. PFSP uses milliseconds per instance; the other rows use seconds per instance. FFSP column labels specify machines per stage, while the other column labels specify machine counts.

Table S12: Computational cost of solution quality at fixed task sizes: Gap, latency decomposition, and peak memory for four representation–architecture combinations.

Configuration	**SMSP** 100					**UPMSP** 100 × 6				
	Gap	C	N	T	M	Gap	C	N	T	M
FS + FRA	0.46%	20	53	90	19.8	8.12%	404	920	1530	46.5
TC + FRA	0.31%	8	42	67	17.2	7.83%	122	895	1210	38.6
FS + DS	0.28%	20	39	76	13.3	7.62%	395	706	1310	29.7
TC + DS	**0.18%**	**8**	**27**	**53**	**10.9**	**7.48%**	**118**	**666**	**1010**	**23.1**
Configuration	**PFSP** 100 × 5					**FFSP** 100 × 12				
	Gap	C	N	T	M	Gap	C	N	T	M
FS + FRA	0.09%	1.10	2.22	4.38	28.4	2.46%	132	840	1290	47.3
TC + FRA	-0.04%	0.24	1.75	3.12	24.6	2.21%	49	751	1030	42.8
FS + DS	-0.06%	1.02	1.80	3.61	15.2	2.29%	127	632	1040	32.1
TC + DS	**-0.16%**	**0.22**	**1.12**	**2.46**	**11.8**	**2.27%**	**45**	**564**	**900**	**27.3**
Configuration	**JSSP** 100 × 20					**FJSP** 40 × 10				
	Gap	C	N	T	M	Gap	C	N	T	M
FS + FRA	2.94%	5700	24600	35100	245.8	0.75%	750	3400	5180	164.7
TC + FRA	2.66%	2140	22240	28300	222.4	0.63%	290	3080	4120	137.5
FS + DS	2.38%	5610	19840	29900	74.6	0.47%	720	2860	4620	98.2
TC + DS	**2.22%**	**2010**	**17920**	**24500**	**58.7**	**0.39%**	**268**	**2450**	**3600**	**74.5**

Gap: percent; C/N/T: compilation/network/total milliseconds per complete schedule; M: mean per-instance peak allocated memory (MiB). Configurations follow Table S8. Total includes initialization, construction, selection, and execution; separately profiled C and N need not sum to T.

Table S13: Quality gained from additional decoding effort. The two panels use identical checkpoints, test instances, and budgets; they pair mean Gap (%) with time (s) to return the selected schedule.

Gap (%)

Task	G	S16	S64	S128	S512
SMSP	0.18%	-0.28%	-0.63%	-0.80%	-0.96%
UPMSP	7.48%	6.93%	6.42%	6.18%	6.07%
PFSP	-0.16%	-0.25%	-0.34%	-0.39%	-0.45%
FFSP	2.27%	0.86%	0.02%	-0.42%	-0.62%
JSSP	2.22%	1.90%	1.58%	1.43%	1.24%
FJSP	0.39%	0.29%	0.21%	0.16%	0.12%

Time (s)

Task	G	S16	S64	S128	S512
SMSP	0.053	0.089	0.176	0.294	0.923
UPMSP	1.010	1.680	3.450	5.320	15.860
PFSP	0.00246	0.0037	0.0058	0.0076	0.0208
FFSP	0.900	1.520	2.820	4.160	12.780
JSSP	24.500	44.900	92.600	154.800	503.600
FJSP	3.600	8.300	15.600	22.800	63.000

G: Greedy; Sk: best of k complete sampled schedules. Representative sizes: SMSP 100; UPMSP 100 × 6; PFSP 100 × 5; FFSP 100 × 12; JSSP 100 × 20; FJSP 40 × 10. Time includes all trajectories and best-schedule selection, without division by k. Quality comparisons are made within a task, not by averaging gaps across different references.

E.5 Stability across Training Seeds

Stability across independent training runs. Table S14 reports paired w/o TC minus Full TADS Gap differences across five training seeds. Mean differences favor Full TADS in all six tasks under both decoding modes. FFSP exhibits negative differences for some individual seeds and larger variation relative to its mean gain. Thus, the average benefit of TC does not imply an improvement in every independent training run.

Table S14: Training-seed stability of TC gains at fixed sizes. Paired ΔGap (percentage points) equals w/o TC minus Full TADS; positive values favor Full TADS.

Task	Decoding	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4	Mean	SD
SMSP	Greedy	0.61	0.80	0.68	0.87	0.69	0.73	0.104
	Sampling	0.42	0.61	0.53	0.70	0.49	0.55	0.108
UPMSP	Greedy	1.12	1.51	1.28	1.40	1.34	1.33	0.145
	Sampling	0.85	1.17	0.96	1.08	0.99	1.01	0.121
PFSP	Greedy	0.52	0.74	0.63	0.70	0.66	0.65	0.084
	Sampling	0.38	0.56	0.47	0.52	0.47	0.48	0.067
FFSP	Greedy	0.05	0.35	-0.03	0.49	0.34	0.24	0.220
	Sampling	-0.08	0.21	0.02	0.31	0.19	0.13	0.157
JSSP	Greedy	0.93	1.30	1.03	1.24	1.10	1.12	0.151
	Sampling	0.69	0.98	0.80	0.94	0.89	0.86	0.116
FJSP	Greedy	0.59	0.83	0.68	0.79	0.71	0.72	0.094
	Sampling	0.40	0.62	0.48	0.57	0.53	0.52	0.085

Sizes: Table S13. Each seed column contains a paired mean difference over the test instances. Sampling uses 128 schedules per variant. Mean and sample SD summarize the five seed-wise paired means.

E.6 Detailed Public-Benchmark Results

Table S15: Public-benchmark objective, Gap (%), and runtime (s) for frozen TADS checkpoints.

Task	Group	Greedy			Sampling		
		Obj.	Gap (%)	Time (s)	Obj.	Gap (%)	Time (s)
PFSP	20×5	1229.49	0.58%	0.00110	1226.19	0.31%	0.0050
	50×5	2747.89	0.42%	0.00185	2741.05	0.17%	0.0068
	100×5	5263.90	0.37%	0.00310	5251.84	0.14%	0.0098
JSSP	30×20	2255.87	15.84%	1.81	2173.30	11.60%	32.90
	50×20	3146.21	10.63%	4.72	3017.38	6.10%	88.60
	100×20	5527.74	3.02%	25.40	5416.67	0.95%	479.20
FJSP	Brandimarte	181.79	8.21%	2.14	178.33	6.15%	13.70
	Hurink (edata)	1122.16	9.16%	4.85	1098.42	6.85%	29.80
	Hurink (rdata)	988.83	5.87%	4.38	975.28	4.42%	27.40
	Hurink (vdata)	926.63	0.83%	5.61	923.32	0.47%	34.60

Baselines follow Table 2. PFSP and JSSP use Taillard groups; FJSP uses Brandimarte and Hurink. Sampling budgets are 128, 512, and 100, respectively. Both decoding modes use the same checkpoints, instances, and references. Values are within-group means; runtimes are seconds per instance.

Public-benchmark detail. Table S15 reports objective, Gap, and latency for frozen checkpoints. Sampling improves Gap in every benchmark group at greater inference cost. Remaining gaps are largest in the smaller JSSP groups and Hurink edata; JSSP also incurs a substantial sampling-time increase. Objectives are interpreted within groups because sizes, processing-time scales, and reference values differ.

E.7 Optimization and Architecture Sensitivity

Table S16: Optimization and architecture sensitivity under greedy decoding. Each configuration is independently trained with seeds 0–4, and each entry reports the mean test Gap across the five trained models.

Setting	Value	SMSP	UPMSP	PFSP	FFSP	JSSP	FJSP
Default		0.18%	7.48%	-0.16%	2.27%	2.22%	0.39%
Width	64	0.31%	7.96%	-0.04%	2.43%	2.58%	0.55%
	256	0.20%	7.44%	-0.14%	2.31%	2.18%	0.41%
Global layers	1	0.47%	7.73%	0.00%	2.19%	2.67%	0.53%
	4	0.16%	7.46%	-0.15%	2.30%	2.19%	0.40%
Local layers	1	0.34%	8.25%	0.08%	2.51%	2.45%	0.72%
	4	0.18%	7.39%	-0.17%	2.24%	2.20%	0.38%
Replay length	32	0.30%	7.63%	-0.10%	2.26%	2.41%	0.46%
	128	0.17%	7.43%	-0.16%	2.30%	2.21%	0.37%
Initial LR	5×10^{-5}	0.23%	7.60%	-0.12%	2.33%	2.30%	0.44%
	2×10^{-4}	0.28%	7.74%	-0.09%	2.41%	2.39%	0.49%

Sizes follow Table S13. Defaults: width 128, three global and three local layers, replay length 64, and initial learning rate 10^{-4}. Each non-default configuration changes one setting relative to the default while keeping the remaining training protocol fixed. The Default row uses the corresponding five Full TADS runs.

Sensitivity to optimization and architecture. Table S16 varies one setting at a time and reports five-seed mean Gaps under greedy decoding. The default is within 0.09 percentage points of the best tested Gap for every task. Reducing width or local depth increases mean Gap on all six tasks; a fourth local layer improves five and matches SMSP at the reported precision, with gains of at most 0.09 percentage points. Wider embeddings, deeper global networks, and longer replay yield mixed effects. Halving or doubling the initial learning rate increases mean Gap on every task. The consistent degradation from reducing width or local depth suggests that sufficient capacity for candidate representation and comparison is important under the tested protocol. Further increases in capacity or replay length offer limited or task-dependent gains. The default remains close to the lowest observed mean Gaps, although the best configuration varies by task.