ACTBASIS: COMPACT ACTION PARAMETERIZATION FOR FINE-GRAINED NETWORK SCHEDULING

Anonymous authors
Paper under double-blind review

ABSTRACT

Deterministic networks require coordinated routes and precise transmission times so that periodic messages meet deadlines without link conflicts. Higher timing precision increases the number of available start times within a fixed scheduling window, expanding the range of scheduling choices. Deep reinforcement learning can learn these coupled decisions from interaction, yet directly parameterizing an action-selection weight for every time-grid position on each link makes the actor output grow with scheduling precision. This raises the question of whether fine-grained action selection requires a correspondingly large actor output dimension. We propose ACTBASIS, which predicts compact coefficients over a fixed temporal basis to generate action-selection weights across the full scheduling grid. Together with routing weights, these action-selection weights guide the selection of a feasible route and transmission start time. Network experiments under shared feasibility constraints show that timing-score representations support effective action selection at high scheduling precision. ACTBASIS retains this performance level while reducing timing output dimension by up to a factor of 125 relative to direct matrix parameterization. Controlled experiments show that compact parameterization shapes scheduling behavior even before training. The results show that compact actor outputs can support fine-grained action selection while maintaining effective scheduling performance.

1 INTRODUCTION

Industrial automation and avionics rely on timely communication among distributed devices. Deterministic networks coordinate transmissions to deliver messages within specified deadlines (Craciunas et al., 2016). For time-triggered traffic, scheduling determines each message's route and transmission times while preventing conflicts on shared links (Jia et al., 2021). Effective scheduling allows the network to accommodate more messages while meeting these requirements.

Scheduling decisions are coupled across links and messages (Craciunas et al., 2016; Pahlevan et al., 2019). A transmission must fit the available intervals along its route and remain compatible with periodic repetitions. A feasible placement for the current message also changes the transmission opportunities available to later messages. Scheduling quality therefore depends on which feasible placement is selected. Finer timing precision increases the number of available start times within a fixed scheduling window, expanding the range of scheduling choices (Zhang et al., 2026).

Existing methods address this complexity in different ways. Heuristics guide the search through allocation rules (Craciunas et al., 2016; Pahlevan et al., 2019). Reinforcement learning learns scheduling policies from interaction. TTDeep and DeepScheduler divide scheduling into smaller decisions and use structured network representations to guide their choices (Jia et al., 2021; He et al., 2023). For a learned scheduler, the action representation determines how policy outputs guide the selection of transmission placements. When timing choices are represented by independently predicted weights, finer discretization increases the actor output dimension (Jia et al., 2021). This raises the question of whether fine-grained action selection requires a correspondingly large actor output dimension.

We propose ACTBASIS, a compact action parameterization for network scheduling. For each link, the policy predicts coefficients over a fixed, shared temporal basis, which generates action-selection weights across the full scheduling grid. The grid determines the available transmission times, while

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.

the coefficient dimension controls the capacity of the action representation. This separates the number of policy outputs from the granularity of executable timing choices.

In high-precision network experiments, ACTBASIS improves scheduling success by 4.57 percentage points over a list-scheduling heuristic on a 1000-message instance, while staying close to a direct one-weight-per-slot parameterization. With 8000 time-grid positions, it uses only 64 timing outputs per link, a $125\times$ reduction. A controlled single-resource study further shows higher mean test success than this direct parameterization across three grid sizes with $d{=}64$. Capacity controls reveal a trade-off between coefficient count and scheduling quality. Initialization comparisons further show that the advantage of ACTBASIS with 64 coefficients is already present before training, while learning improves both representations.

This work makes three contributions. (1) We develop ACTBASIS, which separates policy output dimension from scheduling precision while retaining action selection over the full scheduling grid under the scheduling constraints. (2) We characterize the representational capacity of the fixed-basis parameterization, derive sufficient conditions for preserving decoded decisions under score perturbations, and quantify output and reconstruction costs. (3) Experiments on network scheduling show that compact action representations retain strong performance at high precision. They reveal a trade-off between representation capacity and scheduling quality, and show that parameterization shapes policy behavior both before and after training.

2 RELATED WORK

2.1 NETWORK SCHEDULING AND LEARNING-BASED SCHEDULERS

Time-triggered scheduling coordinates routing and transmission timing under communication constraints. Formal methods encode these requirements explicitly, as in the gate-control list scheduling formulation of Craciunas et al. (2016). Heuristics guide allocation through scheduling rules, including joint routing and scheduling in the list scheduler of Pahlevan et al. (2019).

Beyond these classical approaches, learning-based schedulers address scheduling constraints through decision decomposition and structured representations. TTDeep (Jia et al., 2021) separates next-link and forwarding-phase selection, followed by earliest-feasible-offset calculation. DeepScheduler (He et al., 2023) combines decision decomposition with graph-based network encoding and flow-aware representations. Other learning approaches adapt scheduling strategies in unknown environments (Zhang et al., 2026). Graph representations, multi-agent collaboration, and joint routing-scheduling have also been studied with continuous-action or deep reinforcement learning (Islam & Muslim, 2024; Chen et al., 2025; García-Cantón et al., 2024). While these learning-based schedulers emphasize how decisions are decomposed or encoded, ACTBASIS addresses how timing action-selection weights are parameterized when the scheduling grid grows with precision.

2.2 LARGE ACTION SPACES AND ACTION REPRESENTATIONS

The timing representation emphasized above connects to broader work on action representations for large decision spaces. Classic value-based methods such as DQN (Mnih et al., 2015) assign one score to each discrete action. When every time-grid position is scored in the same way, the number of outputs grows with the grid size. Action representations more generally separate policy outputs from executable choices. Dulac-Arnold et al. (2015) combine continuous action embeddings with approximate nearest-neighbor retrieval for large discrete action sets. Chandak et al. (2019) learn representations and a mapping to executable actions. These approaches motivate compact internal representations when the set of choices is large. For fine discretization, Tang & Agrawal (2020) study factorized discrete policies and ordinal parameterization for continuous control, and Seo et al. (2025) refine action intervals through coarse-to-fine value-based selection.

Fixed bases provide another way to structure continuous action spaces. Q-functionals represent continuous-action value functions as state-dependent coefficients over Fourier or polynomial bases (Lobel et al., 2023), supporting efficient evaluation over many discrete actions. BEAST encodes action sequences with B-spline bases into compact tokens (Zhou et al., 2025). Unlike these uses for value functions or executed trajectories, ACTBASIS parameterizes link-specific action-selection weights over a fixed temporal basis, so actor outputs need not grow with the timing grid.

3 PROBLEM FORMULATION

We consider sequential message scheduling on a directed network $G = (V, E)$ with $R = |E|$ links. An episode presents M messages. At each decision step, the scheduler considers one message and observes a state s with the current message, occupancy, and process context. The policy and critic receive state features $x = \psi(s)$, where ψ maps the scheduling state to its feature representation.

Transmission times lie in a fixed scheduling window of length T with spacing Δ. Assuming an integral ratio, the number of time-grid positions is

$$H = \frac{T}{\Delta}. \tag{1}$$

Finer timing precision increases H within the same physical window. Let $\mathcal{P}(s)$ denote the route set for the current message, and let $\Omega_p(s)$ denote the feasible start times on route p that satisfy period completion, end-to-end deadlines, and non-overlap with reserved transmissions, and for which all induced transmissions fall within the scheduling window. These start times are offsets from the beginning of a period; hop transmission times follow from the chosen offset and fixed forwarding shifts along the route. The objective is to maximize the expected scheduling success rate:

$$\max_{\pi} \ \mathbb{E}_{\pi}\left[\frac{1}{M}\sum_{i=1}^{M} c_i\right], \tag{2}$$

where $c_i \in \{0, 1\}$ indicates whether message i is successfully scheduled.

3.1 DENSE CONTINUOUS ACTION INTERFACE

Under the continuous action interface studied here, the policy emits routing weights $u \in \mathbb{R}^R$ together with a timing-score matrix

$$A \in \mathbb{R}^{R\times H}, \tag{3}$$

with RH continuous coordinates. Each entry $A_{e,h}$ is a timing score for link e at time-grid position h, serving as an action-selection weight. The single-link case reduces A to a vector of length H. When at least one feasible combination of route and start time exists, a scheduling decoder

$$(p_i, \tau_i) = D_s(u, A) \tag{4}$$

maps (u, A) to a feasible route $p_i \in \mathcal{P}(s)$ and start time $\tau_i \in \Omega_{p_i}(s)$ according to the routing weights and timing scores. If no placement is feasible, the current message is rejected. Feasibility is fixed by the constraints. The action representation determines how the policy fills the H columns.

3.2 COST OF A GROWING COLUMN DIMENSION

Under direct parameterization of this interface, the continuous output has $R + RH$ components and grows linearly with H. For a fully connected head of input width w that emits all RH timing scores, $(w + 1)RH$ parameters are required, including biases, with $O(wRH)$ operations per evaluation. Raising precision therefore enlarges the timing branch of this interface.

Our goal is to reduce the number of timing outputs while retaining the full scheduling grid and effective scheduling performance.

4 METHODOLOGY

ACTBASIS supports fine-grained scheduling decisions through a compact action parameterization. As shown in Figure 1, the actor predicts routing weights and timing coefficients from scheduling-state features. A shared basis maps the coefficients to timing scores over the full scheduling grid. The decoder selects a feasible route and then a transmission time along that route. The policy is trained through the reconstructed action.

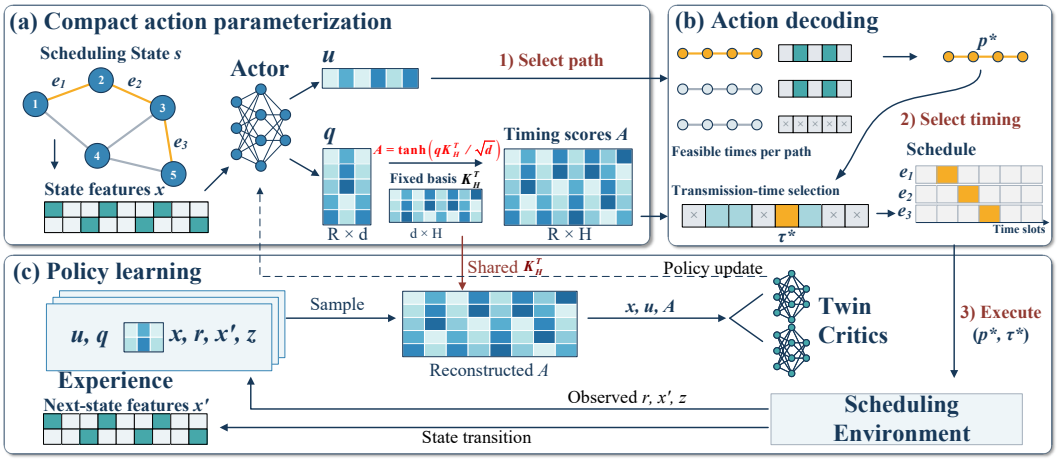

Figure 1: ACTBASIS overview. (a) Given state, the actor predicts routing weights u and compact timing coefficients q. A fixed shared basis K_H maps q to timing scores A over the full scheduling grid. (b) The decoder uses u to select a feasible route p^*, then aggregates timing scores along that route to select a transmission start time τ^*. Schedule rows correspond to links e_1, e_2, e_3 on the selected route. (c) Replay stores behavior outputs (u, q) with transition data and reconstructs A for critic updates. Policy gradients pass through the critics and reconstruction map.

4.1 COMPACT ACTION PARAMETERIZATION

As shown in Figure 1(a), given state features x, the actor outputs $(u, q) = \pi_\theta(x)$, where θ denotes the actor parameters. Here, $u \in \mathbb{R}^R$ contains routing weights and $q \in \mathbb{R}^{R\times d}$ contains d timing coefficients per link. Unlike DQN-style outputs that assign one value to each discrete action (Mnih et al., 2015), our parameterization uses compact coefficients to generate timing scores through a fixed, shared basis $K_H \in \mathbb{R}^{H\times d}$:

$$A = F_H(q) = \tanh(qK_H^\top/\sqrt{d}). \tag{5}$$

Each column of K_H evaluates a basis function over the H time-grid positions. Each row of q combines these functions for one link, and the elementwise $\tanh$ bounds the resulting scores. Ordered time positions allow shared basis functions to describe variation across the scheduling grid, while state-dependent coefficients determine the scores for each decision. Keeping the basis fixed makes the coefficient-to-score mapping consistent across states and transitions.

The grid size H controls how many time-grid positions receive scores. The coefficient count d controls representation capacity. For a fixed basis family and coordinate convention, increasing H evaluates the same functions on a finer grid, while increasing d expands the number of coefficients available to construct each row of A. The actor therefore emits R routing weights and Rd timing coefficients, $R(d+1)$ scalars in total, while the fixed basis expands these coefficients to RH timing scores. At fixed d, refining the grid changes how finely the basis is evaluated, not the number of timing coefficients produced by the actor. Figure 5 in Appendix A illustrates this mapping.

4.2 SCHEDULING DECODER

Given the routing weights u and timing scores A, the decoder converts them into a scheduling action (p^*, τ^*). As shown in Figure 1(b), this proceeds in two stages: route selection and transmission-time selection. Here, τ specifies the first-hop transmission start time relative to the beginning of a period. The decoder first considers routes that admit at least one feasible start time. It scores each route by summing its routing weights and selects the highest-scoring route.

$$L(p; u) = \sum_{e\in p} u_e, \tag{6}$$

$$p^* = \arg\max_{p\in\mathcal{P}(s):\Omega_p(s)\neq\varnothing} L(p; u). \tag{7}$$

For the selected route, each feasible start time determines transmission times across all hops and periodic repetitions. The decoder evaluates each feasible start time by averaging the corresponding

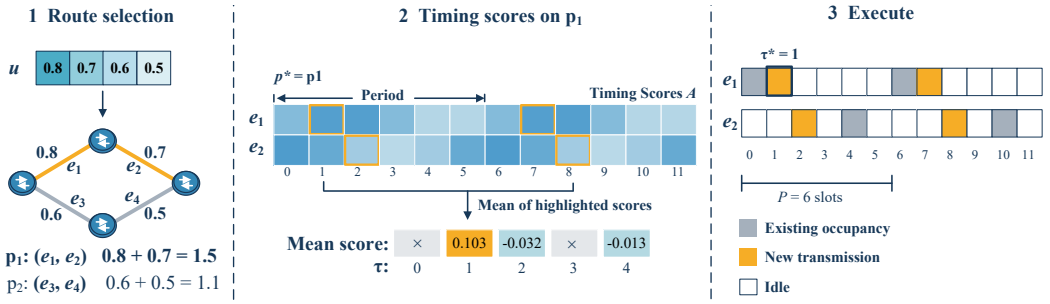

Figure 2: Route and transmission-time selection for one request with $H = 12$, period $P = 6$, and hop shifts $(0, 1)$. Both routes admit feasible start times. Routing-weight sums select $p_1 = (e_1, e_2)$. The middle panel shows timing scores on the selected path. Averaging over both links and periodic repetitions selects $\tau^* = 1$ from the feasible set $\{1, 2, 4\}$. Orange outlines mark the four gathered scores for $\tau = 1$.

timing scores at these positions. For the current message period P_f, let $\mathcal{I}_f = \{j \in \mathbb{Z}_{\geq 0} : jP_f < H\}$. Let $\sigma_{p,e}$ be the cumulative forwarding shift to link e along route p. The route length $|p|$ is the number of links in p. All timing quantities are measured in slots. Conditional on p^*,

$$\mu(p^*, \tau; A) = \frac{\sum_{j \in \mathcal{I}_f} \sum_{e \in p^*} A_{e, jP_f + \tau + \sigma_{p^*,e}}}{|\mathcal{I}_f|\,|p^*|}, \tag{8}$$

$$\tau^* = \arg\max_{\tau \in \Omega_{p^*}(s)} \mu(p^*, \tau; A). \tag{9}$$

Ties are broken separately for the two stages. Routing ties are resolved by a fixed route order. Timing ties are resolved by selecting the earliest feasible start time. If no route admits a feasible start time, the message is rejected. Figure 2 illustrates this two-stage decoding.

In the example, the routing-weight sums are 1.5 for $p_1 = (e_1, e_2)$ and 1.1 for $p_2 = (e_3, e_4)$, selecting p_1. For $\tau = 1$, with period $P = 6$ and hop shifts $(0, 1)$, the decoder gathers scores at positions 1 and 7 on e_1 and positions 2 and 8 on e_2. Their mean score is the highest among the feasible start times, so these four transmission slots are reserved.

For a single resource without periodic aggregation, let $\Omega(s)$ contain the feasible start times on the sole resource. Decoding reduces to

$$\tau^* = \arg\max_{\tau \in \Omega(s)} F_H(q)_{1,\tau}. \tag{10}$$

4.3 Policy learning with parameterized actions

The parameterization and decoder above define how actions are generated and executed. As shown in Figure 1(c), we train this policy with TD3 (Fujimoto et al., 2018). The actor produces routing weights u and timing coefficients q, and twin critics evaluate the resulting representation through $C_{\phi_i}(x, u, F_H(q))$, where ϕ_i denotes the parameters of critic i.

During data collection, exploration and clipping produce behavior outputs $(\tilde{u}, \tilde{q})$. The decoder maps $(\tilde{u}, F_H(\tilde{q}))$ to a schedule, and replay stores $(x, \tilde{u}, \tilde{q}, r, x', z)$. Here, r is the reward, x' contains the next-state features, and $z \in \{0, 1\}$ indicates episode termination. Successfully scheduling the current message yields reward $1/M$, and otherwise the reward is zero. With unit discount, the sum of these step rewards equals the fraction of successfully scheduled messages in Equation 2. When a transition is sampled, the same fixed basis reconstructs $F_H(\tilde{q})$, providing the critics with the timing scores associated with the observed transition.

The critics are trained with the TD3 temporal-difference target

$$y = r + \gamma(1 - z) \min_{i=1,2} C_{\phi_i^-}(x', u', F_H(q')). \tag{11}$$

where γ is the discount factor and (u', q') are target-actor outputs from x' after smoothing and clipping. Here, ϕ_i^- denotes the parameters of target critic i. Delayed actor updates maximize

$$J(\theta) = \mathbb{E}_{x \sim \mathcal{D}} \left[C_{\phi_1}(x, u_\theta(x), F_H(q_\theta(x))) \right]. \tag{12}$$

Algorithm 1 Training ACTBASIS

Require: Fixed basis K_H, actor π_θ, critics C_{ϕ_1}, C_{ϕ_2}, target networks, replay buffer $\mathcal{D}$

for each environment step **do**
  Observe features $x = \psi(s)$
  Sample behavior outputs $(\tilde{u}, \tilde{q})$ from $\pi_\theta(x)$ with exploration and clipping
  Reconstruct $A = F_H(\tilde{q})$ and decode (p^*, τ^*) as in Section 4.2
  Execute or reject the request, observe reward r, next features x', and terminal flag z
  Store $(x, \tilde{u}, \tilde{q}, r, x', z)$ in $\mathcal{D}$
  if $\mathcal{D}$ contains at least one minibatch of transitions **then**
    Sample a minibatch from $\mathcal{D}$ and reconstruct behavior timing scores with F_H
    Update critics with the TD target in Equation 11
    if the delayed actor-update schedule triggers **then**
      Update θ by maximizing Equation 12
      Soft-update the target networks
    end if
  end if
end for

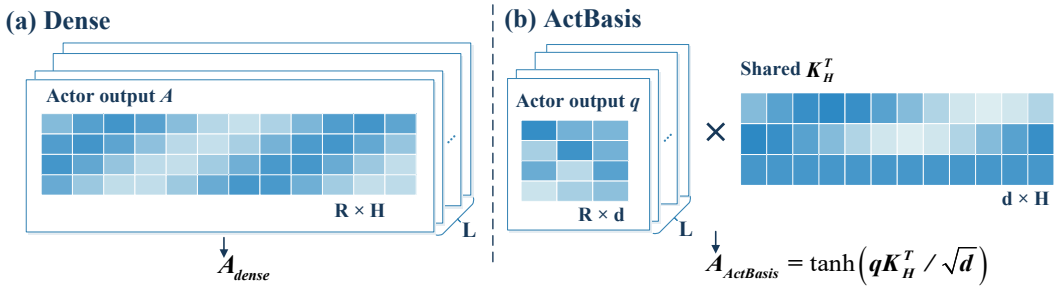

Figure 3: Timing-action parameterization across L scheduling decisions. Each card is the timing output for one decision. (a) Dense emits an $R \times H$ timing-score matrix. (b) ACTBASIS emits $R \times d$ coefficients and reconstructs timing scores with a fixed shared basis. Both use the same scheduling grid and scheduling decoder. The routing branch is omitted.

Here, $\mathcal{D}$ is the replay distribution over state features. Gradients for the timing branch propagate through F_H to the actor. Algorithm 1 summarizes the overall training loop.

4.4 REPRESENTATION CAPACITY AND COMPUTATIONAL COST

The compact parameterization reduces policy outputs by restricting the family of timing scores the actor can generate. Its effectiveness therefore depends on whether this family supports useful scheduling choices. The fixed basis and coefficient dimension d determine this family. Before the elementwise nonlinearity, each link's timing profile lies in the span of the basis functions. Since different score matrices can decode to the same schedule, representation quality depends on preserving the distinctions needed to select effective feasible start times.

If each timing score changes by at most ϵ, the aggregated score of each feasible start time also changes by at most ϵ. For fixed feasibility sets and unchanged routing weights, a unique selected start time is therefore preserved whenever its aggregated score exceeds every alternative by more than 2ϵ. Thus, preserving the selected start time does not require reproducing every timing score exactly. The corresponding analysis and coefficient-space bounds are given in Appendix A.

Figure 3 illustrates the H/d reduction in timing outputs across successive decisions when $d < H$. For a linear timing output layer with input width w, the parameter count falls from $(w+1)RH$ to $(w+1)Rd$, while constructing the full timing-score matrix costs $O(RHd)$ operations per decision. The resulting saving is in actor output dimension and output-layer parameters, while action selection still uses the full scheduling grid.

5 EXPERIMENTS

We evaluate whether compact timing actions retain effective scheduling decisions while reducing the timing output dimension.

5.1 EXPERIMENTAL SETUP

Common feasibility interface. Preliminary runs with unconstrained probabilistic proposals achieved very low scheduling success, obscuring differences in placement quality, in line with reports that standard RL without masking constraints performs poorly in this setting (Zhang et al., 2026). We therefore use a common feasibility-matching and execution interface for all compared methods. Method-specific mappings select a legal placement whenever one is available, and messages with an empty feasible set are rejected. This shared feasibility handling contributes to the high success rates. The comparison therefore focuses on how each method selects among feasible placements (Figure 1(b), Appendix B).

Network study. We use an A380-style topology (24 nodes, 62 directed links), as considered in prior scheduling studies on this aircraft network (Maletić et al., 2023). It provides routing, multi-hop, and periodic scheduling constraints within a fixed network, allowing us to examine action representations across timing precisions and traffic loads. We focus on 500-message and 1000-message loads. A 100-message prefix reaches full scheduling success under the shared feasibility interface for all compared methods, so it offers little contrast for comparing placement quality and is omitted from the main comparison. Preliminary experiments on smaller topologies showed little separation among methods under the shared feasibility interface, even with increased message counts. Their shorter routes reduced the scope for placement choices to affect subsequent scheduling. We use $H \in \{400, 8000\}$, corresponding to coarse and fine slot sizes (20 μs and 1 μs) within the same physical scheduling window. These settings provide a 20-fold change in grid size, contrasting coarse and fine timing choices. This setting tests whether compact timing outputs remain competitive when placement choices couple across links and later messages. Policies are optimized on each instance and evaluated deterministically at the selected checkpoint. Heuristics include Earliest (Jia et al., 2021), List (Pahlevan et al., 2019), and Center (Zhu et al., 2025). Reference RL methods include GCN-TD3 (Islam & Muslim, 2024), MADDPG (Chen et al., 2025), and Joint DQN (García-Cantón et al., 2024). Dense (Figure 3(a)) and ACTBASIS (d=64; Figure 3(b)) share the encoder, TD3 learner, and decoder and differ only in the timing branch. A matched PPO pair (Schulman et al., 2017) uses the same shortest-path-first (SPF) routing rule. During training, PPO-Gaussian samples feasible start times from a Gaussian-induced discrete distribution, while PPO-ACTBASIS samples basis coefficients.

Controlled study. To isolate timing parameterization from routing coupling, a single-resource task with 100 requests per instance over a fixed physical window varies $H \in \{400, 2000, 8000\}$ and $d \in \{16, 32, 64\}$ across three seeds, with validation-based checkpoint selection and an independent test set. Increasing H refines the set of executable start times. The geometrically increasing coefficient dimensions allow us to examine how additional representation capacity affects scheduling quality and the compression ratio at each grid size.

Metrics. We report the scheduling success rate (the percentage of input messages scheduled under all routing and timing constraints) together with the timing output dimension. Method definitions, group-specific training budgets, and remaining protocol details appear in Appendix B.

Table 1: A380 scheduling success rates (%). Learning results report mean±standard deviation (n=3). Rows report 500-message and 1000-message loads. The bottom row averages method means within each group for H=8000 and 1000 messages.

		Heuristics			Reference RL				Timing-score RL		
H	M	Earliest	List	Center	GCN-TD3	MADDPG	Joint DQN	PPO-G	Dense	**ACTBASIS**	PPO-ACTBASIS
400	500	83.8	83.8	84.6	84.87±0.12	84.67±0.12	84.73±0.42	84.33±0.50	**86.87±0.31**	86.07±1.03	86.67±0.12
	1000	70.1	70.3	68.5	69.40±0.30	68.97±0.15	**70.57±1.19**	69.77±0.29	67.40±0.83	67.90±0.35	66.27±1.04
8000	500	95.8	95.8	99.8	99.87±0.12	99.93±0.12	98.13±0.46	98.60±1.51	99.40±0.20	**100.00±0.00**	99.73±0.46
	1000	90.4	90.3	93.8	93.80±0.10	93.73±0.12	92.60±0.62	91.10±0.61	**95.30±1.05**	94.87±0.90	94.93±0.61
Group mean (8000, 1000)		91.50			92.81				95.03		

5.2 NETWORK SCHEDULING PERFORMANCE

Table 1 reports success under the shared feasibility interface. Within that interface, performance depends on which feasible placement is selected and how that choice affects subsequent messages.

Heuristics select feasible placements through prescribed rules, such as choosing the earliest available time or a position near the center of a gap. Reference RL methods instead learn state-dependent proposals that are converted into legal placements through their respective mappings. Timing-score policies learn scores across links and time-grid positions, allowing the decoder to compare feasible start times by aggregating scores over the transmissions each choice would induce.

Timing-score selection performs well in the high-precision 1000-message setting, where the timing-score group averages 95.03% success, compared with 91.50% for heuristics and 92.81% for reference RL. All three timing-score methods achieve higher mean success rates than the other evaluated methods in this setting. Since feasible placements are available to all methods through the shared interface, this advantage reflects how effectively their selection policies accommodate subsequent messages. The results suggest that comparing timing scores across the transmissions induced by each placement helps preserve future scheduling opportunities. Pairwise statistical comparisons are reported in Appendix B.4. Under the shared feasibility interface, several methods approach 100% success on the high-precision 500-message setting, leaving less room for performance differences.

Achieving this timing-score performance does not require emitting an independent score for every time-grid position. In the high-precision 1000-message setting, ACTBASIS reaches 94.87% success compared with Dense's 95.30%, while reducing timing outputs per link from 8000 to 64, a factor of 125. Its 4.57-percentage-point gain over List corresponds to approximately 46 additional scheduled messages. Across the evaluated network settings, its mean success rates differ from Dense by at most 0.8 percentage points. Once common state fields are included, the archived A380 1000-message buffer still records a 9.81× mean payload reduction and about 10× more filled transitions under the same MiB budget (Table 2). ACTBASIS thus retains effective fine-grained scheduling while substantially reducing the number of timing outputs.

Table 2: Measured A380 replay compression at H=8000, M=1000.

Measurement	Dense	ACTBASIS
Bytes / transition	2,191,440	223,312
Transitions in 256 MiB	122	1,202
Transitions in 512 MiB	244	2,404
Payload ratio	9.81×	

The matched PPO comparison further supports this finding within the same learning algorithm. In the high-precision 1000-message setting, PPO-ACTBASIS achieves 94.93% success, compared with 91.10% for PPO-Gaussian. This result supports generating timing scores from compact coefficients for fine-grained placement selection. Deployment timings, including the modest overhead of ACTBASIS relative to Dense, are reported in Appendix B.5.

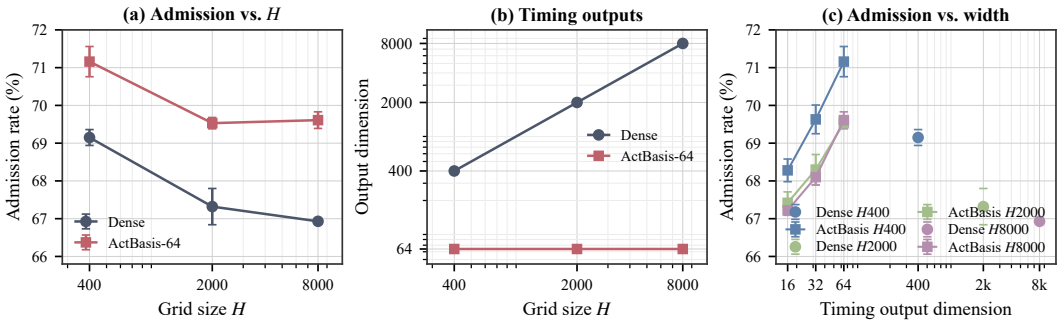

Figure 4: Single-resource results. (a) Success rate versus grid size. (b) Timing output dimension versus grid size. (c) Success rate versus timing output dimension. Error bars indicate standard deviation across three training seeds.

5.3 ACTION PRECISION AND COMPACT REPRESENTATION

The network results show that compact timing parameterization can support effective scheduling under routing and multi-hop constraints. We next isolate the relationship between scheduling precision and representation capacity in a single-resource task. For each fixed coefficient dimension, increasing H refines the scheduling grid without increasing the timing output dimension of ACTBASIS. With d=64, ACTBASIS achieves higher mean test success rates than Dense at all three grid sizes, while its timing output dimension remains fixed. As H increases from 400 to 8000, the output reduction relative to Dense grows from $6.25\times$ to $125\times$ (Figure 4).

Compact parameterization nevertheless requires sufficient capacity to distinguish useful timing choices. At fixed H, increasing d expands the family of timing profiles available to the policy, and the mean test success rate increases across the tested coefficient dimensions. These results indicate that finer scheduling grids do not require independently predicting a score for every time-grid position. Shared basis functions allow compact coefficients to express useful relationships across timing positions, while the coefficient count controls the flexibility of those relationships. The required output dimension therefore depends on the representation capacity needed for effective scheduling, rather than on the number of time-grid positions alone.

5.4 COMPRESSION AND LEARNING CONTROLS

These controls separate structural output compression from numerical storage precision and from training. FP16 reduces the bytes used to store each timing value, whereas ACTBASIS reduces the number of policy outputs. At H=8000, FP16 halves the Dense timing payload from 32000 to 16000 bytes without changing its output dimension (Table 3). Dense FP16 remains close to Dense FP32 in mean success rate, while ACTBASIS with 64 coefficients achieves higher means at both grid sizes. Reducing numerical precision therefore does not reproduce the performance observed with compact parameterization.

Table 3: Numerical-precision control on the single-resource task. Success rates (%) are mean±standard deviation over three seeds.

Method	H=400	H=8000
Dense FP32	69.15±0.21	66.93±0.05
Dense FP16	69.30±0.07	67.29±0.55
ACTBASIS-64	71.16±0.40	69.61±0.22

With d=64, ACTBASIS achieves higher mean success rates than Dense at initialization across all three grids, and training further improves the mean success rates of both policies (Table 4). These results suggest that compact parameterization provides a useful structural inductive bias for scheduling. Its role extends beyond reducing output size to shaping the placement preferences on which subsequent learning builds.

Table 4: Untrained and test success rates (%) on the single-resource task. Results are mean±standard deviation over three seeds. Test results use validation-selected checkpoints.

H		Dense	ACTBASIS-16	ACTBASIS-32	ACTBASIS-64
400	Untr.	67.99±0.58	67.71±0.25	68.50±0.12	70.45±0.39
	Test	69.15±0.21	68.28±0.30	69.63±0.38	71.16±0.40
2000	Untr.	65.64±0.37	67.23±0.21	67.58±0.07	69.17±0.68
	Test	67.32±0.48	67.42±0.29	68.30±0.40	69.53±0.15
8000	Untr.	65.51±0.56	67.31±0.19	67.39±0.29	68.94±0.59
	Test	66.93±0.05	67.22±0.13	68.10±0.21	69.61±0.22

6 CONCLUSION

This paper studied how reinforcement learning can support fine-grained network scheduling with compact policy outputs. We introduced ACTBASIS, which generates link-specific action-selection weights from coefficients over a fixed temporal basis for schedule selection under constraints. The network experiments demonstrate the effectiveness of timing-score selection among feasible placements, particularly at finer scheduling precision. ActBasis retains this scheduling capability with substantially fewer outputs. These findings suggest that the actor output dimension should reflect the capacity needed to distinguish useful actions, rather than mirror the granularity of the scheduling grid. In network scheduling, ACTBASIS demonstrates that this distinction can be preserved through compact coefficients while retaining fine-grained action selection. Future work will investigate adaptive basis representations and examine when compact parameterizations retain effective decisions in other structured action spaces.

REFERENCES

Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip Thomas. Learning action representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 941–950, 2019.

Changsheng Chen, Anrong Zhao, Zhihao Zhang, Tao Zhang, and Chao Fan. Research on multi-agent collaborative scheduling planning method for time-triggered networks. *Electronics*, 14(13): 2575, 2025. doi: 10.3390/electronics14132575.

Silviu S. Craciunas, Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, pp. 183–192, 2016. doi: 10.1145/2997465.2997470.

Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

Sergi García-Cantón, Cristina Cervelló-Pastor, David Rincón, and Sebastià Sallent. Deep reinforcement learning-based joint scheduling and routing for time-sensitive networks. In *24th International Conference on Transparent Optical Networks (ICTON)*, pp. 1–4, 2024. doi: 10.1109/ICTON62926.2024.10647521.

Xiaowu He, Xiangwen Zhuge, Fan Dang, Wang Xu, and Zheng Yang. DeepScheduler: Enabling flow-aware scheduling in time-sensitive networking. In *IEEE INFOCOM*, pp. 1–10, 2023. doi: 10.1109/INFOCOM53939.2023.10228875.

Syed Tasnimul Islam and Anas Bin Muslim. AI-based dynamic schedule calculation in time sensitive networks using GCN-TD3. In *IFIP Networking Conference*, 2024. doi: 10.23919/IFIPNetworking62109.2024.10619807.

Hongyu Jia, Yu Jiang, Chunmeng Zhong, Hai Wan, and Xibin Zhao. TTDeep: Time-triggered scheduling for real-time Ethernet via deep reinforcement learning. In *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2021. doi: 10.1109/GLOBECOM46510.2021.9685850.

Samuel Lobel, Sreehari Rammohan, Bowen He, Shangqun Yu, and George Konidaris. Q-functionals for value-based continuous control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8932–8939, 2023. doi: 10.1609/aaai.v37i7.26073.

Želimir Maletić, Miloš Ljubojević, and Mihajlo Savić. Comparative analysis of time-sensitive network scheduling and traffic shaping mechanisms for A380 aircraft network topology. In *31st Telecommunications Forum (TELFOR)*, pp. 1–4, 2023. doi: 10.1109/TELFOR59449.2023.10372723.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. doi: 10.1038/nature14236.

Maryam Pahlevan, Nadra Tabassam, and Roman Obermaisser. Heuristic list scheduler for time triggered traffic in time sensitive networks. *ACM SIGBED Review*, 16(1):15–20, 2019. doi: 10.1145/3314206.3314208.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Younggyo Seo, Jafar Uruç, and Stephen James. Continuous control with coarse-to-fine reinforcement learning. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2866–2894, 2025.

Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5981–5988, 2020.

W. Zhang, J. Wu, A. Zhao, Q. Wang, C. Chen, T. Zhang, and P. Zhang. Multi-source strategy adaptation network for time-triggered flow scheduling in unknown environments. *Applied Soft Computing*, 186:113988, 2026. doi: 10.1016/j.asoc.2025.113988.

Hongyi Zhou, Weiran Liao, Xi Huang, Yucheng Tang, Fabian Otto, Xiaogang Jia, Xinkai Jiang, Simon Hilber, Ge Li, Qian Wang, Ömer Erdinç Yağmurlu, Nils Blank, Moritz Reuss, and Rudolf Lioutikov. BEAST: Efficient tokenization of B-splines encoded action sequences for imitation learning. In *Advances in Neural Information Processing Systems*, 2025.

Haowen Zhu, Zhen Li, Jinwei Cheng, and Zhonghe Jin. Research on a hybrid scheduling algorithm based on critical-link optimization for large-scale time-triggered Ethernet. *Sensors*, 25(20):6347, 2025. doi: 10.3390/s25206347.

A Analysis of Compact Action Parameterization

A compact action parameterization must express useful timing actions and preserve their meaning during replay. We analyze its representational capacity, the stability of decoded scheduling decisions, and the resulting storage requirements.

A.1 Illustration of the shared temporal basis

For illustration, let $\xi \in [0, 1]$ and define $k(\xi) = (\sin(2\pi\xi), \cos(2\pi\xi))$. A coefficient vector $q = (a, b)$ produces $A(\xi) = \tanh((a \sin(2\pi\xi) + b \cos(2\pi\xi))/\sqrt{2})$. Figure 5 uses the same basis for all three coefficient vectors and the same vertical scale for all output curves. This two-function example illustrates action reconstruction. The basis used in the resolution study is defined in Equation 19.

A.2 Temporal representation capacity

Each transposed pre-activation row of $Z = qK_H^\top/\sqrt{d}$ lies in $\mathrm{col}(K_H)$, whose dimension is at most d. With unrestricted coefficients, a timing matrix $A \in (-1, 1)^{R\times H}$ is exactly representable if and only if every transposed row of $\mathrm{arctanh}(A)$ lies in this subspace. The parameterization therefore controls the family of temporal profiles through the basis and coefficient dimension. The elementwise nonlinearity maps these profiles to bounded timing scores.

A.3 Decision equivalence and stability

At a fixed state and routing vector, define $A \sim_{s,u} A'$ when $D_s(u, A) = D_s(u, A')$. Timing actions in the same equivalence class produce the same execution decision. The following result quantifies a neighborhood of decision equivalence using the margins between competing routes and offsets.

Proposition 1 (decision stability under perturbations). Fix feasibility sets and aggregation indices. Suppose $\|u' - u\|_\infty \le \epsilon_u$ and $\|A' - A\|_\infty \le \epsilon_A$. A unique winning route p^* remains selected if, for every other feasible p,

$$L(p^*; u) - L(p; u) > \|\chi_{p^*} - \chi_p\|_1 \epsilon_u, \tag{13}$$

where χ_p is the link-incidence vector used in the route sum. For simple routes, $\|\chi_{p^*} - \chi_p\|_1 = |p^* \triangle p|$, since shared links cancel. Conditional on this route, a unique winning offset remains selected if every competing offset has a score gap greater than $2\epsilon_A$. The change in a route-score difference is $(\chi_{p^*} - \chi_p)^\top (u' - u)$, bounded by $\|\chi_{p^*} - \chi_p\|_1 \epsilon_u$. A mean of fixed entries changes by at most ϵ_A, proving the offset statement. The aggregation uses fixed nonempty index sets with normalized nonnegative weights.

Corollary (coefficient perturbations). Let k_h be row h of the fixed basis and define $\kappa = \max_h \|k_h\|_2$. If $\max_e \|q'_e - q_e\|_2 \le \eta$, then

$$\|F_H(q') - F_H(q)\|_\infty \le \frac{\kappa\eta}{\sqrt{d}}. \tag{14}$$

Indeed, $\tanh$ is 1-Lipschitz and Cauchy-Schwarz bounds each pre-activation change by $\|q'_e - q_e\|_2 \|k_h\|_2/\sqrt{d}$. With unchanged routing weights and decoder context, an offset margin greater than $2\kappa\eta/\sqrt{d}$ is sufficient to preserve a unique decision. The bound connects coefficient perturbations directly to a sufficient decision margin.

For the paired harmonic basis in Equation 19, $\|k_h\|_2^2 = d/2$ at every grid location. Equation 14 therefore becomes $\|F_H(q') - F_H(q)\|_\infty \le \eta/\sqrt{2}$, independent of H. Under this construction, a fixed coefficient perturbation yields an H-independent bound on the reconstructed timing action. For other basis families, the relevant scale is $\kappa/\sqrt{d}$. Comparisons should specify basis scaling and row-norm bounds together with coefficient bounds, since equal coefficient noise need not yield equal timing-action perturbations.

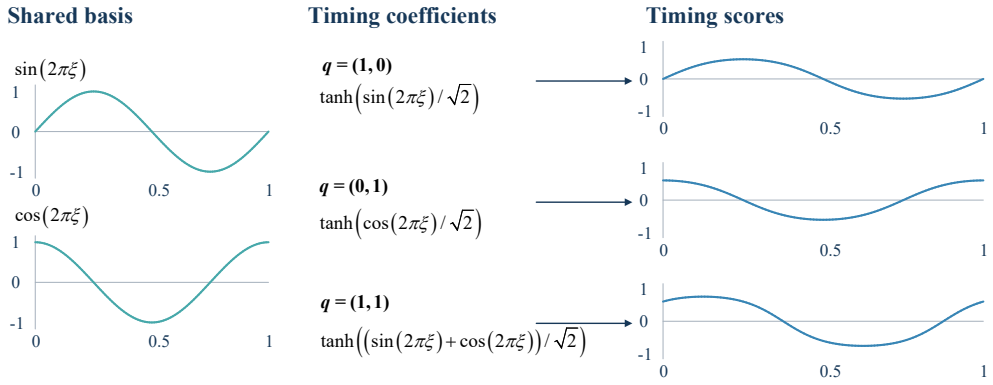

Figure 5: Illustration of the shared temporal basis. Two fixed basis functions, $\sin(2\pi\xi)$ and $\cos(2\pi\xi)$, generate timing scores for three coefficient vectors through $A(\xi) = \tanh((a\sin(2\pi\xi) + b\cos(2\pi\xi))/\sqrt{2})$, where $q = (a, b)$. Curves show the continuous basis expansion, sampled at $\xi_h = h/H$, $h = 0, \ldots, H-1$, for the scheduling grid.

A.4 Behavior reconstruction and resource costs

Proposition 2 (behavior reconstruction). Fix H, d, K_H, and the reconstruction map. If replay preserves the behavior coefficients q_b without loss, then in exact arithmetic $A_{\text{replay}} = F_H(q_b) = A_{\text{behavior}}$. If routing weights, decoder context, route and time-slot ordering, and tie-breaking are also unchanged, the decoded action is identical. This follows by substitution into the deterministic reconstruction map and decoder.

Persistent memory. For b bytes per scalar and replay capacity N, dense and factorized timing payloads require $bNRH$ and $bNRd$ bytes, respectively. Caching one basis adds bHd bytes. If S_H denotes common per-transition state, routing, and metadata storage, the simplified full-replay costs are

$$B_{\text{dense}} = N(S_H + bRH), \tag{15}$$

$$B_{\text{compact}} = N(S_H + bRd) + B_{\text{basis}}. \tag{16}$$

The timing-action payload is reduced by a factor of H/d. Total replay memory additionally includes common transition fields and the actual basis caches across devices and processes. State storage may itself grow with H.

Equal-budget replay capacity. For a replay budget $B_{\max} \geq B_{\text{basis}}$, excluding fixed learner memory and under the additive storage model above, the capacities are

$$N_{\text{dense}} = \left\lfloor \frac{B_{\max}}{S_H + bRH} \right\rfloor, \tag{17}$$

$$N_{\text{compact}} = \left\lfloor \frac{B_{\max} - B_{\text{basis}}}{S_H + bRd} \right\rfloor. \tag{18}$$

At equal N, compact replay saves memory only when $NbR(H-d) > B_{\text{basis}}$. If common state storage has the form $S_H = cH + S_0$ with $c > 0$, then at fixed N, R, d, b, S_0 and ignoring basis caches the full-replay ratio tends to $(c + bR)/c$ as H grows, whereas the timing-only ratio H/d keeps increasing. Basis caches further reduce this benefit. These expressions motivate measuring actual allocations when assessing training capacity or speed.

Computation. Explicit basis reconstruction costs $O(RHd)$ operations and materializes $O(RH)$ scores. A batch of size B may require $O(BRH)$ score storage. These costs describe the trade-off between persistent storage and reconstruction work in the full-grid critic interface. We therefore account for reconstruction workspace and latency alongside replay bytes.

Table 5: Shared TD3 and network settings for the resolution study. Dense and compact policies use the same values.

Setting	Value
Discount factor γ	1
Actor / critic learning rate (Adam)	3×10^{-4}
Soft-update rate	5×10^{-3}
Behavior exploration noise (std.)	0.15
Target smoothing noise (std.)	0.2
Target noise clip	0.5
Policy delay (critic steps)	2
Batch size	32
Hidden width	64
Replay capacity N	3000
Environment steps	100,000

B Experimental Configurations and Additional Results

B.1 Single-resource configuration

The single-resource study uses a physical window of $8000\,\mu$s with 100 requests per instance, 200 training instances, 100 validation instances, and 200 test instances. Temporal resolutions $H \in \{400, 2000, 8000\}$ correspond to slot durations of 20, 4, and $1\,\mu$s. Compact policies use $d \in \{16, 32, 64\}$. A dense policy supplies one reference per resolution and training seed. The workload follows the `default_v1` generation recipe used in the archived resolution study. Request durations and timing constraints are sampled once per instance and held fixed across compared methods.

The common training budget is 100,000 environment steps with replay capacity N=3000. Validation uses all 100 instances every 5000 steps, including step zero. The checkpoint with highest mean validation scheduling success rate is selected, with ties resolved in favor of the earlier checkpoint. Final checkpoints are also evaluated. The comparison uses three training seeds, each covering all resolution and dimension settings. Table 5 summarizes the shared TD3 and network hyperparameters used by dense and compact policies.

A nested basis for the resolution study. To vary temporal resolution and representation dimension independently, we use normalized slot-start coordinates $\xi_h = h/H$, corresponding to physical times $t_h = hT/H$. For even d, the controlled study uses ordered harmonic pairs

$$[K_{H,d}]_{h,2j-1} = \sin(2\pi j \xi_h), \qquad [K_{H,d}]_{h,2j} = \cos(2\pi j \xi_h), \tag{19}$$

where $h = 0, \ldots, H-1$ and $j = 1, \ldots, d/2$. Increasing H samples the same temporal features more finely, and increasing d adds higher-frequency pairs. The smaller bases are column prefixes of the largest basis, with a common unit scale. This construction gives the dimension sweep a consistent ordering of temporal subspaces.

Memory and runtime measurement. Transition payloads include stored states, actions, rewards, terminal flags, and metadata. Shared basis caches are counted separately for their actual process and device copies. Peak host and device memory additionally include learner parameters, optimizer state, allocator overhead, and temporary tensors. Runtime measurements distinguish policy evaluation, action reconstruction, feasibility decoding, replay sampling, and actor/critic updates. Actual update counts are retained alongside the environment-step budget.

B.2 Archived reconstruction and buffer measurements

The archived A380 reconstruction checks use the fixed shared basis implementation at $H = 400$ and $H = 8000$, corresponding to slot durations of 20 and $1\,\mu$s. The maximum absolute error across the recorded checks is zero at both resolutions. These measurements support the behavior-action reconstruction property used by compact replay.

Table 6: Network training budgets for Table 1. Learning methods keep the best verified success rate within the budget. Time and episode limits stop training when either is reached.

Family	Setting	Stop budget
Heuristics	all	—
Dense / ACTBASIS / PPO-G / PPO-ACTBASIS	H=8000	8 episodes
	H=400, M=500	8 episodes
	H=400, M=1000	24 episodes
GCN-TD3 / MADDPG / Joint DQN	H=400	180 s or 40 episodes
	H=8000	600 s or 8 episodes

For the A380 instance with 1000 messages and $H = 8000$, actual buffer fills record a mean transition payload of 2,191,440 bytes for dense actions and 223,312 bytes for compact actions, a $9.81\times$ reduction (Table 2 in the main text). The payload includes common transition fields. Shared basis caches and process/device overhead are accounted for separately from these measurements.

B.3 NETWORK SETTING AND COMPARED METHODS

The A380 network has 24 nodes and 62 directed links, with periodic transmissions, forwarding shifts, and link conflicts. The main comparison uses nested prefixes with 500 and 1000 messages. A 100-message prefix is fully scheduled by all compared methods and is omitted from the main tables because it offers little contrast. Grid sizes $H = 400$ and $H = 8000$ correspond to slot durations of 20 and 1 μs, respectively. Network ACTBASIS and PPO-ACTBASIS use the nested harmonic construction in Equation 19 with the same $\xi_h = h/H$ coordinates and $d = 64$. Policies are optimized on each instance. For Dense and ACTBASIS, the checkpoint with the highest verified scheduling success rate observed on that instance during the training budget is retained. The selected checkpoint is then evaluated deterministically.

All compared methods use a common feasibility-matching and execution mechanism. Policy outputs are mapped to scheduling proposals and checked against the feasible-placement mask. Only feasible placements are written into the schedule (Figure 1(b)). When a method-specific proposal is infeasible, the mechanism projects it onto a legal placement if one exists. Otherwise the message is rejected under the shared execution rule. Under a given state and route, feasible start times are determined by the same rules. Dense and ACTBASIS select among feasible start times using timing scores. They share the encoder architecture, TD3 learner, critic interface $C(x, u, A)$, and decoder, differing in timing-branch parameterization. The decoder first selects a route that admits a feasible placement using routing weights, then selects the feasible start time with the highest aggregated timing score along that route. The message is rejected if no route admits a feasible placement. PPO-Gaussian and PPO-ACTBASIS use a matched PPO protocol with the same fixed SPF routing rule and evaluation. PPO-Gaussian predicts state-dependent $(\mu, \log\sigma)$ for a one-dimensional Gaussian on the real line, assigns to each period slot the Gaussian probability mass of the corresponding normalized interval, renormalizes these masses on the feasible offset set Ω_p, samples an offset during training, and selects the $\arg\max$ offset on that distribution at deployment. After selecting (p^*, h) it constructs an input that makes the decoder return the chosen route and start time, rather than sampling a dense score matrix. This Ω_p-conditioned version is the protocol behind Table 1. Proposals are therefore drawn only from legal offsets. Budgeted pilots of unconstrained probabilistic proposals without feasible-set conditioning produced very low success rates and were not retained as formal baselines. PPO-ACTBASIS samples temporal basis coefficients q from a diagonal Gaussian, reconstructs timing scores with the fixed basis, and evaluates the policy log-probability at the sampled coefficients q. At deployment, PPO-ACTBASIS uses the mean coefficients and reconstructs timing scores for decoding. Feasible selection then follows the same score-based decoder as Dense and ACTBASIS.

Earliest follows the SPF-early preference for the earliest feasible slot (Jia et al., 2021). List uses the joint routing-scheduling rule of Pahlevan et al. (2019). Center prefers placements near the middle of an available gap, following the idea of center-out slot search for balanced TT placement (Zhu et al., 2025). For the reference RL implementations of GCN-TD3 (Islam & Muslim, 2024), MADDPG (Chen et al., 2025), and Joint DQN (García-Cantón et al., 2024), continuous proposals that fall

Table 7: Success-rate comparisons at H=8000, M=1000 (n=3).

Comparison	Test	p
Dense vs. List	one-sample t	0.014
ACTBASIS vs. List	one-sample t	0.013
PPO-ACTBASIS vs. List	one-sample t	0.006
ACTBASIS vs. GCN-TD3	Welch t	0.18
ACTBASIS vs. MADDPG	Welch t	0.16
ACTBASIS vs. Joint DQN	Welch t	0.069
ACTBASIS vs. PPO-G	Welch t	0.027
Dense vs. ACTBASIS	Welch t	0.64
Dense vs. PPO-ACTBASIS	Welch t	0.65
ACTBASIS vs. PPO-ACTBASIS	Welch t	0.93

outside the feasible mask are projected to the nearest legal slot (GCN-TD3 and MADDPG), while Joint DQN selects among at most 24 pre-filtered legal $(\text{path}, \text{offset})$ pairs. Empty masks or empty feasible sets skip the message under the same execution verifier. Illegal actions are therefore never written into the schedule. Dense, ACTBASIS, and PPO-ACTBASIS form the timing-score RL group. PPO-Gaussian is grouped with the other reference RL methods. Table 6 summarizes family-specific training budgets. Methods are compared within their respective protocols.

For Dense and ACTBASIS, a deterministic evaluation is run every 4 episodes on the training instance; the checkpoint with the highest scheduling success rate is retained. PPO-G and PPO-ACTBASIS keep the best deployment scheduling success rate observed under the same episode budgets. Adapted baselines stop at the first of the wall-time and episode caps above.

B.4 STATISTICAL COMPARISONS FOR THE NETWORK TABLE

We compare each timing-score method with the deterministic List baseline using a two-sided one-sample t-test across three training seeds. We also report ACTBASIS against the adapted reference RL methods, and pairwise comparisons among Dense, ACTBASIS, and PPO-ACTBASIS, using Welch t-tests on the reported means and standard deviations. Table 7 lists unadjusted p-values. Non-significance does not establish equivalence.

Table 8: Time to schedule one complete request sequence on the same machine (seconds) on A380. Avg./msg is $1000\times\text{Time}/M$ in milliseconds. The italic row averages the timing-score RL method means.

	(a) H=400				(b) H=8000			
	M=500		M=1000		M=500		M=1000	
Method	Time	/msg	Time	/msg	Time	/msg	Time	/msg
PPO-Gaussian	7.7	15.3	12.0	12.0	178.9	357.8	277.6	277.6
Timing-score RL (group mean)	7.0	14.1	12.6	12.6	129.4	258.7	218.0	218.0
Dense	6.4	12.8	12.5	12.5	100.0	200.0	179.3	179.3
ACTBASIS	6.9	13.8	12.5	12.5	115.3	230.6	188.2	188.2
PPO-ACTBASIS	7.8	15.6	12.7	12.7	172.8	345.6	286.6	286.6

B.5 NETWORK SOLVE-TIME MEASUREMENTS

Deployment timing is a supplementary evaluation and is not treated as a primary claim alongside scheduling success or timing-branch width. For Dense, ACTBASIS, and the matched PPO pair we report wall-clock time for one noise-free deterministic evaluation of the selected checkpoint on the same host, excluding training, model loading, and logging. The per-message average is $1000\times$ that time divided by the number of offered messages (including rejected ones). Cross-machine totals and short-training wall clocks are omitted. Table 8 shows that reducing timing outputs does not shorten deployment: ACTBASIS includes an additional basis-reconstruction step and has modestly higher measured deployment time than Dense, and the PPO pair is typically slower than the TD3 pair.

B.6 Additional measurements

Equal-memory learning comparisons and alternative basis families were not run for this submission and are omitted. Initialization and test rates appear in Table 4. The numerical-precision control is Table 3.

C TD3 Training Settings

Table 5 lists the shared TD3 and network hyperparameters used by dense and compact policies. The training loop and update equations appear in Section 4.3.

D AI Use Statement

The core research idea and initial experiments were developed by the authors. GPT-6 Astra assisted with drafting and revising the manuscript and with feedback on the experiments, results, and related literature. Grok 4.7 assisted with reviewing related literature, organizing experimental logs, calculating evaluation metrics, consistency checks, and language editing. The AI-assisted literature review, log processing, and metric calculations were checked by an author. Kimi 2.7 Code assisted with code writing. The authors reviewed and revised the AI-assisted material and take full responsibility for the methods, results, citations, and final manuscript.