OT-DISPATCH NET: LEARNING DYNAMIC DISPATCH POLICIES VIA OPTIMAL TRANSPORT

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we study the multiple traveling salesman problem (mTSP) in a dynamic setting, termed DmTSP, where multiple agents jointly serve a set of requests that arrive sequentially and must be assigned without knowledge of future demand. Compared with mTSP, DmTSP poses a key challenge of jointly determining request allocation across agents and individual assignments while accounting for their downstream routing effects, which remains unexplored. To tackle DmTSP, we propose OT-Dispatch Net, a reinforcement learning framework based on optimal transport (OT). Specifically, OT-Dispatch Net uses an agent marginal and transport costs to model request allocation and agent-request preferences, respectively, allowing OT to couple allocation and assignment decisions in a unified dispatch plan. A soft-quota decoder is further developed to map the resulting transport plan to feasible joint assignments. The policy is trained using a potential-based reward function designed to capture the downstream routing effects induced by dispatch decisions. We theoretically characterize the impact of approximation errors on dispatch quality and show that workload stability can be preserved under bounded decision error. Experiments across synthetic and real-world replay settings demonstrate its superior performance over existing learning-based methods.

1 INTRODUCTION

The multiple traveling salesman problem (mTSP) concerns the coordinated service of a shared set of requests by multiple agents, with applications in passenger transport and urban delivery (Xu et al., 2018; Voccia et al., 2019). Exact, heuristic, and neural methods have been developed for static settings in which all requests are known before routes are constructed (Bektas, 2006; Son et al., 2024; Berto et al., 2025; Luttmann & Xie, 2026). However, this assumption does not hold in most real systems. Requests arrive over time, decisions must be made without knowledge of future demand, and each assignment changes agent locations, workloads, and availability. The task therefore becomes a sequence of coupled assignment and routing decisions rather than a single static routing problem. We study the dynamic multiple traveling salesman problem (DmTSP), in which agents receive newly revealed requests at successive decision epochs and update their routes accordingly.

DmTSP couples decisions both across epochs and within each epoch. Across epochs, each assignment changes an agent's route, future location, and availability, so an assignment that appears favorable at the current epoch may increase travel and waiting time at later epochs. Within an epoch, requests cannot be evaluated independently because assigning one request changes the route cost of assigning another to the same agent. Request allocation determines how the requests awaiting assignment are distributed across agents, while individual assignments specify which agent serves each request. Each dispatch decision must coordinate these two choices. An effective policy must update both choices as new requests arrive and agent states evolve, while accounting for their delayed consequences.

Neural combinatorial optimization (NCO) methods learn reusable routing strategies from data and have been increasingly applied to multi-agent routing problems (Son et al., 2024; Berto et al., 2025; Luttmann & Xie, 2026). However, most of these methods address static problems in which all requests are known before route construction. Dynamic routing methods adapt assignments to the current state through learned pair scores, sequential decisions, conflict resolution, or external matching (Xu et al., 2018; Enders et al., 2023; Hoppe et al., 2024). Although these methods determine which agent serves each request, the number of requests assigned to each agent remains implicit in the resulting

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.

assignments. In DmTSP, evolving routes and agent availability may require different request counts even when preferences for specific agent-request pairs remain unchanged. This motivates representing request allocation within the dispatch policy, alongside preferences for individual assignments.

In this work, we propose **OT-Dispatch Net**, a bilevel sequential framework that couples learned dispatch decisions with downstream route optimization. At each decision epoch, the upper-level policy represents request allocation through a learned *agent marginal* and agent-request preferences through learned *transport costs*. Entropic optimal transport (OT) couples these two components into a continuous *transport plan* (Sinkhorn & Knopp, 1967; Cuturi, 2013; Peyré & Cuturi, 2019; Blondel et al., 2020), jointly modeling allocation and assignment within a single dispatch decision. A *soft-quota decoder* converts the transport plan into executable joint assignments, which are passed to a lower-level route optimizer. The resulting routing and service outcomes are incorporated into a potential-based reward, providing dense learning signals at intermediate decision epochs while capturing the downstream effects of dispatch decisions.

Our key contributions are as follows:

- We formulate DmTSP as a bilevel sequential decision process, with dispatch at the upper level and route optimization at the lower level. The formulation connects current assignments to future routes and agent availability, allowing the dispatch policy to learn from service outcomes over complete episodes.
- We propose OT-Dispatch Net, which explicitly parameterizes request allocation through a learned agent marginal and couples it with agent-request preferences using entropic OT. The resulting transport plan coordinates preferences across the entire batch subject to the allocation learned for the current dispatch state.
- We derive explicit bounds on cost sensitivity to the agent marginal and soft-quota decoding, and establish conditional stability under an auxiliary workload model. Experiments across synthetic and New York City (NYC) replay settings demonstrate the policy's effectiveness in reducing request waiting time and makespan across different fleet sizes.

2 Related work

Neural combinatorial optimization with multiple agents. NCO methods learn policies for the traveling salesman problem (TSP) and vehicle routing problem (VRP), with many foundational approaches constructing routes one decision at a time (Bello et al., 2017; Nazari et al., 2018; Kool et al., 2019; Khalil et al., 2017; Kwon et al., 2020). Extensions to multiple agents must additionally coordinate which agent acts and which request it serves. Equity-Transformer formulates routing that minimizes the longest route as sequential generation and introduces an equity context to balance route construction (Son et al., 2024). PARCO (Berto et al., 2025) allows agents to act in parallel through learned communication and resolves conflicting choices by priority, whereas MACSIM (Luttmann & Xie, 2026) samples conflict-free matchings from joint agent-task logits. Other attention architectures coordinate route construction across agents (Cao et al., 2024). Hybrid approaches such as iMTSP learn to assign the requests of a fully revealed instance and then solve the resulting TSP for each agent (Hu et al., 2020; Guo et al., 2024). Despite their different architectures and decoding strategies, these methods do not explicitly model the number of requests assigned to each agent. Instead, the resulting request count for each agent is determined by sequential decoding, conflict resolution, or a downstream solver. OT-Dispatch Net instead learns an agent marginal that specifies the desired request allocation across agents and produces a joint assignment at every epoch.

Dynamic routing and dispatch. Dynamic routing research considers arriving customers (Voccia et al., 2019; Zhang et al., 2023), changing traffic (Chen et al., 2025; Cheong & White, 2012), travel times that depend on departure time (Malandraki & Daskin, 1992), and online routing studied through competitive analysis (Baligács et al., 2023). Dynamic TSP methods often consider a single vehicle (Zhang et al., 2023; Chen et al., 2025), whereas multi-vehicle methods revise dispatch or routing plans as new information arrives (Bent & Van Hentenryck, 2004; Voccia et al., 2019). Xu et al. (2018) estimate the present and future value of each driver-request pair and pass these values to a centralized combinatorial optimizer. This learning-and-matching design has also been deployed in large-scale ride-hailing systems (Qin et al., 2020). Hybrid methods for multiple agents likewise combine learned decisions with weighted bipartite matching to obtain globally coordinated

assignments (Enders et al., 2023; Hoppe et al., 2024). Structured reinforcement learning (SRL) moves this integration inside the actor by embedding a combinatorial optimization layer (Hoppe et al., 2025). In these dispatch methods, learned values or scores guide matching or optimization, while the realized request counts are not specified by a learned agent marginal. Related graph RL work adopts a bilevel architecture in which a learned controller specifies a desired network state and a lower-level convex program realizes the action (Gammelli et al., 2023). Martens (2025) studies a bilevel routing problem with known target trajectories over the full horizon, unlike our online setting with unknown future requests. Our dispatch policy operates without knowledge of future requests and explicitly learns the agent marginal from evolving route and availability information.

Optimal transport for assignment. Optimal transport (OT) computes a cost-minimizing coupling between two distributions subject to marginal constraints. It has been applied to discrete assignment with agent dynamics (Kachar & Gorodetsky, 2022) and ride-hailing dispatch (Lei & Wu, 2022), while Sinkhorn-based formulations enable differentiable learning over permutations and matchings (Mena et al., 2018) and have been used as constraint layers for static neural TSP models (Goh et al., 2022). Differentiable OT methods build on entropic regularization and Sinkhorn scaling (Sinkhorn & Knopp, 1967; Cuturi, 2013; Peyré & Cuturi, 2019). Unlike the cited uses of OT for assignment or matching constraints, OT-Dispatch Net learns the agent marginal as a state-conditioned policy output, allowing request allocation to adapt as routes and availability evolve. OT couples this allocation with agent-request preferences, and a soft-quota decoder produces executable joint assignments.

3 Problem Setup

DmTSP formulation. We consider a metric space $(\mathcal{V}, d)$ with depot ℓ_{depot} and M agents $\mathcal{M} = \{1, \dots, M\}$, initially at the depot. Agents travel at unit speed and serve assigned requests sequentially. Each request $r_j = (\ell_j, \alpha_j)$, $j \in \mathcal{J}$, has location ℓ_j and release time $\alpha_j \in [0, T]$, and is revealed only upon arrival. Dispatch epochs are spaced by Δt, yielding $N_{\text{arr}} = T/\Delta t$ intervals over the *arrival horizon*. A complete episode includes both arrivals up to T and subsequent service until all assigned routes are complete and all agents have returned to the depot.

A solution $\pi = \{\pi_1, \dots, \pi_M\}$ assigns each request to exactly one agent and specifies its service order, with $j_{i,m}$ denoting agent i's mth request. Service start times satisfy $\sigma_{j_{i,m}}(\pi) = \max(\alpha_{j_{i,m}}, \sigma_{j_{i,m-1}}(\pi) + d(\ell_{j_{i,m-1}}, \ell_{j_{i,m}}))$ with $\sigma_{j_{i,0}} = 0$ and $\ell_{j_{i,0}} = \ell_{\text{depot}}$. The waiting time is $w_j(\pi) = \sigma_j(\pi) - \alpha_j$, and the average waiting time is $\overline{W}(\pi) = |\mathcal{J}|^{-1} \sum_{j \in \mathcal{J}} w_j(\pi)$. Let $C_i(\pi_i)$ denote agent i's completion time, including its return to the depot, and define the makespan as $C_{\max}(\pi) = \max_i C_i(\pi_i)$. We seek a nonanticipatory policy minimizing

$$\min_{\pi} \quad \mathbb{E}_{\mathcal{J}}\big[\text{Obj}(\pi)\big], \qquad \text{Obj}(\pi) := \lambda\, C_{\max}(\pi) + (1-\lambda)\overline{W}(\pi), \tag{1}$$

where $\lambda \in [0, 1]$ controls the trade-off between route completion and request waiting time.

Sequential dispatch and routing model. At epoch k, let $\mathcal{B}_k$ denote the unassigned batch, $K_k := |\mathcal{B}_k|$ its size, and $\mathcal{P}_{i,k}$ the requests assigned to agent i awaiting service. The set $\mathcal{W}_k := \mathcal{B}_k \cup \bigcup_i \mathcal{P}_{i,k}$ contains all released requests still waiting. Only $\mathcal{B}_k$ is available for dispatch, since previous assignments remain fixed. We formulate dispatch and routing as a bilevel problem. An upper-level policy generates assignments Y, while a lower-level optimizer selects routes from $\Omega(\mathcal{J}_i(Y))$ for each agent's induced request set $\mathcal{J}_i(Y)$:

$$\min_{\theta} \quad \mathbb{E}_{\mathcal{J}}\mathbb{E}_{Y \sim \pi_\theta}\big[\text{Obj}(\pi)\big] \tag{2a}$$

$$\text{s.t.} \quad Y_k \sim \pi_\theta(\cdot \mid o_k), \quad \forall k \in \{0, \dots, L-1\} \text{ with } K_k > 0, \tag{2b}$$

$$\sum_{i \in \mathcal{M}} Y_{k,ij} = 1, \quad \forall j \in \mathcal{B}_k, \tag{2c}$$

$$\pi_i \in \arg\min_{\pi_i \in \Omega(\mathcal{J}_i(Y))} C_i(\pi_i), \quad \forall i \in \mathcal{M}. \tag{2d}$$

The policy $\pi_\theta(Y_k \mid o_k)$ defines a distribution over assignments $Y_k \in \{0,1\}^{M \times K_k}$ satisfying the coverage constraint (2c). Given these assignments, the lower-level problem in (2d) optimizes the service order of each agent's accumulated requests to minimize its completion time.

4 OT-DISPATCH NET

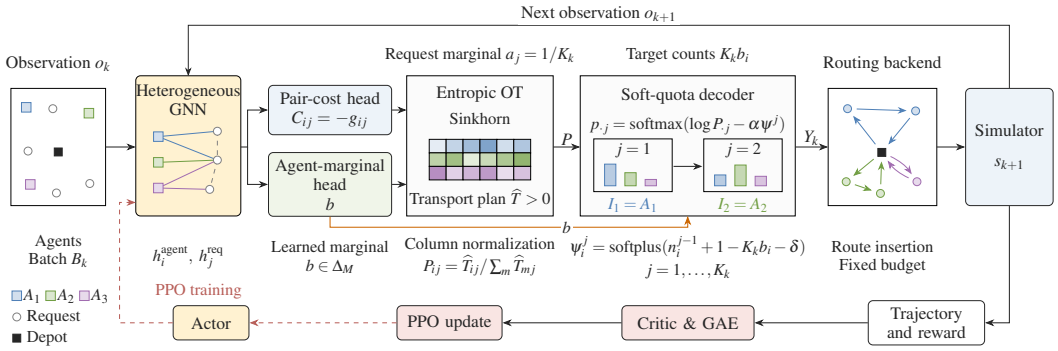

Figure 1: Overview of OT-Dispatch Net.

OT-Dispatch Net addresses dynamic dispatch by explicitly coupling request allocation across agents with individual agent-request assignments. We introduce two key components: (1) an optimal transport policy that learns a state-conditioned agent marginal together with agent-request transport costs, enabling entropic OT to represent allocation and assignment within a single dispatch plan; and (2) a soft-quota autoregressive decoder that transforms the continuous transport plan into executable joint assignments while encouraging the realized request allocation to follow the learned agent marginal. A heterogeneous GNN encodes the evolving dispatch state for both policy heads, while downstream route optimization provides the routing outcomes used to train the policy.

4.1 SEQUENTIAL DECISION PROCESS

We first specify the observations, actions, transitions, and rewards through which the dispatch policy interacts with the routing environment. Let s_k denote the full simulator state and $o_k = h(s_k)$ the policy observation. The agent and request features define the observation as

$$\begin{aligned} x^{\mathrm{a}}_{i,k} &:= \big[p_i(k), n_i(k), \hat{C}_i(k), \tau^{\mathrm{rem}}_i(k), \Delta^{\mathrm{depot}}_i\big], \quad i \in \mathcal{M}, \\ x^{\mathrm{r}}_{j,k} &:= \big[\ell_j, \alpha_j, w_j(k), \|\ell_j - \ell_{\mathrm{depot}}\|\big], \quad j \in \mathcal{B}_k, \\ o_k &:= h(s_k) = \Big(\{x^{\mathrm{a}}_{i,k}\}_{i\in\mathcal{M}}, \{x^{\mathrm{r}}_{j,k}\}_{j\in\mathcal{B}_k}\Big), \end{aligned} \tag{3}$$

where $p_i(k)$, $n_i(k)$, and $\hat{C}_i(k)$ denote agent i's current position, pending request count, and estimated route completion time, respectively. The remaining service time is $\tau^{\mathrm{rem}}_i(k)$, while $\Delta^{\mathrm{depot}}_i$ collects statistics relative to the depot. For request j, $w_j(k) = t_k - \alpha_j$ is its waiting time at dispatch time $t_k = k\Delta t$. At each nonempty epoch, Y_k assigns the entire batch under (2c). The shared routing backend inserts the assigned requests while preserving previous assignments. The simulator then advances to s_{k+1}, incorporating requests released during the transition.

We obtain dense feedback from the terminal objective by defining rewards as decreases in a state-based potential. Let $\mathcal{J}_{\le k}$ contain requests released by t_k, with $\sigma_j = +\infty$ until service begins. The estimate $\widehat{C}^b(s_k)$ predicts completion of the committed routes and retains actual depot return times for completed routes. New requests enter s_{k+1} before its potential is evaluated. At termination, $\widehat{C}^b(s_L) = C_{\max}(\pi)$ and $\overline{W}(s_L) = \overline{W}(\pi)$. The definitions below give the current mean waiting time, potential, and reward; their telescoping sum equals the initial potential minus the terminal objective:

$$\overline{W}(s_k) := \frac{1}{\max\{1, |\mathcal{J}_{\le k}|\}} \sum_{j\in\mathcal{J}_{\le k}} \big(\min\{t_k, \sigma_j\} - \alpha_j\big)_+, \tag{4}$$

$$\Phi(s_k) := \lambda \widehat{C}^b(s_k) + (1-\lambda)\overline{W}(s_k), \tag{5}$$

$$r_k := \Phi(s_k) - \Phi(s_{k+1}), \tag{6}$$

$$\sum_{k=0}^{L-1} r_k = \Phi(s_0) - \Phi(s_L) = \Phi(s_0) - \mathrm{Obj}(\pi). \tag{7}$$

4.2 ENCODER

To represent agent availability and the spatial structure of the current request batch, we encode the observation o_k as a sparse heterogeneous graph with agent and request nodes. Their features, $x^{\mathrm{a}}_{i,k}$ and $x^{\mathrm{r}}_{j,k}$ in (3), characterize agent positions and current service commitments, and request locations and waiting information, respectively.

Each agent is connected bidirectionally to its κ_a nearest requests, enabling local demand and agent availability to be exchanged through message passing. Each request is also connected to its κ_r nearest neighboring requests to capture local spatial context relevant to assignment. Separate input maps embed the two node types into a shared latent space, and L graph isomorphism network (GIN) layers (Xu et al., 2019) then update their representations by aggregating information from neighboring nodes. Shared parameters within each node type and sum aggregation preserve permutation equivariance without requiring node identity embeddings.

The resulting agent embeddings $\{h_i^{\mathrm{agent}}\}_{i=1}^{M}$ are passed to the agent-marginal head, which predicts the agent marginal representing request allocation across agents. The pair-cost head jointly uses the agent embeddings and request embeddings $\{h_j^{\mathrm{req}}\}_{j=1}^{K_k}$ to predict transport costs that encode agent-request preferences.

4.3 OPTIMAL TRANSPORT FOR JOINT ASSIGNMENT

We represent request allocation across agents and agent-request preferences through two policy heads and couple their outputs using entropic optimal transport (Cuturi, 2013; Peyré & Cuturi, 2019).

Coupled policy heads. The agent-marginal head maps agent embeddings to a softmax distribution that represents request allocation across agents. The pair-cost head applies a shared multilayer perceptron (MLP) to score all agent-request pairs in $\mathcal{E}_k^{\mathrm{OT}} = [M] \times [K_k]$:

$$g_{ij} = f_\phi\big([h_i^{\mathrm{agent}} \,\|\, h_j^{\mathrm{req}}]\big), \qquad (i,j) \in \mathcal{E}_k^{\mathrm{OT}}, \tag{8}$$

$$b_i = \frac{\exp\big(f_\psi(h_i^{\mathrm{agent}})\big)}{\sum_{i'=1}^{M} \exp\big(f_\psi(h_{i'}^{\mathrm{agent}})\big)}, \qquad b \in \Delta^M. \tag{9}$$

The preference score g_{ij} defines the transport cost as $C_{ij} = -g_{ij}$, so a more preferred agent-request pair has a lower cost. The agent marginal b specifies how the policy aims to distribute the current request batch across agents, with each entry b_i representing the desired fraction allocated to agent i. These target shares translate into target counts $K_k b_i$, which guide the subsequent soft-quota decoding.

Entropic transport layer. We fix the request marginal as $a_j = 1/K_k$, assigning equal transport mass to each request. Entropic OT couples this fixed request marginal with the learned agent marginal b and transport costs $C = -g$ to obtain a continuous transport plan:

$$\begin{aligned} T_\varepsilon^\star &= \arg\min_{T \in \Pi_{\mathcal{E}_k^{\mathrm{OT}}}(b,a)} \big\{\langle -g, T\rangle - \varepsilon H(T)\big\}, \\ \Pi_{\mathcal{E}_k^{\mathrm{OT}}}(b,a) &= \big\{T \in \mathbb{R}_{\geq 0}^{M \times K_k} \,\big|\, T\mathbf{1}_{K_k} = b,\ T^\top \mathbf{1}_M = a\big\}. \end{aligned} \tag{10}$$

Here, $H(T) = -\sum_{i,j} T_{ij} \log T_{ij}$ denotes the entropy, with regularization strength $\varepsilon > 0$. Each entry T_{ij} represents the transport mass associated with an agent-request pair. The row constraints enforce the learned agent marginal, while the column constraints assign equal mass to each request. Under these constraints, the objective balances transport cost and entropy to coordinate agent-request preferences with the desired request allocation. The decoder in Section 4.4 converts this plan into discrete joint assignments. Alternative request marginals are evaluated in Appendix D.2.

Sinkhorn computation. We obtain an transport plan $\widehat{T}$ using N_{SK} differentiable Sinkhorn iterations that alternately scale the rows and columns. Row scaling adjusts the total transport mass assigned to each agent according to the learned marginal b, while column scaling distributes equal total mass to each request according to the uniform marginal a. During policy optimization, gradients propagate through these iterations to both policy heads, allowing allocation targets and assignment preferences to be learned jointly from routing and service outcomes. The decoder then uses the resulting plan to guide individual assignments and the target counts $K_k b_i$ to coordinate their distribution across agents. Appendix A provides the scaling updates and the positivity analysis.

4.4 DECODER

Algorithm 1 converts the transport plan into a joint assignment Y_k with one agent per request. It normalizes transport columns into base probabilities P_{ij} and assigns requests sequentially. A finite softplus penalty, with scale α and offset δ, discourages assignments beyond the target counts $K_k b_i$. Each conditional probability p_{ij} accounts for earlier assignments through the batch count n_i, distinct from the pending route count $n_i(k)$.

Algorithm 1 Soft-quota decoder.

Require: $\widehat{T}$, agent marginal b, $K_k = |\mathcal{B}_k|$, $\alpha, \delta > 0$
Ensure: $Y_k, \log \pi_\theta(Y_k \mid o_k)$
$P_{ij} \leftarrow \widehat{T}_{ij} / \sum_m \widehat{T}_{mj}$ ▷ Normalize
$Y_k \leftarrow \mathbf{0}$; $\log \pi \leftarrow 0$; $n \leftarrow \mathbf{0}$ ▷ Initialize
for $j = 1$ **to** K_k **do**
  $\psi_i^j \leftarrow \mathrm{softplus}(n_i + 1 - K_k b_i - \delta), \quad \forall i$
  $p_{\cdot j} \leftarrow \mathrm{softmax}(\log P_{\cdot j} - \alpha \psi^j)$
  $I_j \sim p_{\cdot j}$ ▷ Sample agent
  $Y_k[I_j, j] \leftarrow 1$; $n_{I_j} \leftarrow n_{I_j} + 1$
  $\log \pi \leftarrow \log \pi + \log p_{I_j, j}$ ▷ Accumulate
end for
return Y_k, $\log \pi$

Positive base probabilities and finite penalties ensure that every agent remains selectable. The decoder returns a feasible joint assignment in

$$\mathcal{Y}_{K_k} := \left\{ Y \in \{0,1\}^{M \times K_k} \mid Y^\top \mathbf{1}_M = \mathbf{1}_{K_k} \right\}.$$

Every $Y \in \mathcal{Y}_{K_k}$ has positive probability, so policies retain common support across updates. For a stored assignment, history counts are fixed, making $\log \pi_\theta(Y_k \mid o_k) = \sum_j \log p_{I_j, j}$ differentiable through the plan and penalty to both policy heads. Appendix B.1 formalizes these properties and the suppression of assignments beyond target counts. Proposition 7 bounds the cost deviation from the base assignment probabilities using target shares, cost ranges, and penalty parameters.

4.5 TRAINING AND INFERENCE-TIME EXTENSIONS

We train the policy using proximal policy optimization (PPO) (Schulman et al., 2017), generalized advantage estimation (GAE) (Schulman et al., 2016), and the potential-based rewards in (6) (configuration in Appendix E). Rollouts sample Y_k from the soft-quota autoregressive distribution; policy updates recompute each sampled assignment's log probability. Episodes continue after the final arrival until all requests are served and agents return to the depot. Rewards from subsequent routing and service outcomes inform the advantage estimates of earlier assignments.

Two optional test-time procedures reuse the trained policy without retraining. For agents with $M \neq M_0$, *hierarchical task clustering* (HTC) recursively partitions requests using the M_0-agent policy and matches partitions to agents (Appendix C). Separately, test-time refinement samples R joint assignments independently, evaluates their routes, and executes the best candidate.

5 THEORETICAL ANALYSIS

The learned agent marginal, approximate transport plan, and soft quota decoder can each increase assignment cost above its minimum at a dispatch epoch. We analyze these errors and identify conditions for workload stability under repeated approximate decisions in an auxiliary workload model. Proposition 3 establishes feasibility, common support, and differentiability of the joint log probability. At a nonempty epoch, let $\mathcal{F}_k$ contain all information available before dispatch. The learned cost $\widehat{C}(k) = -g(k)$ and transport quantities are $\mathcal{F}_k$-measurable. Define $C_{ij}^{\mathrm{drift}}(k) = Q_i(k) d_{ij}(k)$ for nonnegative workload increments $d_{ij}(k)$, where $Q_i(k)$ tracks workload in an auxiliary workload model, not the actual pending request count or remaining route duration (Neely, 2010).

Four residual terms quantify the cost of imposing the agent marginal (Δ_{marg}), finite Sinkhorn error (Δ_{sink}), decoder deviation (Δ_{soft}), and uniform discrepancy between learned and drift costs (Δ_{cal}). The fixed request marginal limits the entropy contribution to $K_k \varepsilon \log M$, giving

$$\Xi_{\mathrm{soft}}(k) := K_k \varepsilon \log M + \Delta_{\mathrm{marg}}(k) + \Delta_{\mathrm{sink}}(k) + \Delta_{\mathrm{soft}}(k) + 2\Delta_{\mathrm{cal}}(k). \tag{11}$$

Theorem 1 (Expected one-step cost gap). *Under the conditions of Proposition 3, the assignment returned at epoch k satisfies*

$$\mathbb{E}\left[\langle C^{\mathrm{drift}}(k), Y^{(k)} \rangle \,\middle|\, \mathcal{F}_k \right] \leq \min_{Z \in \mathcal{Y}_{K_k}} \langle C^{\mathrm{drift}}(k), Z \rangle + \Xi_{\mathrm{soft}}(k). \tag{12}$$

Propositions 6 and 7 give explicit bounds on two residuals. Let $\mathcal{R}(k)$ be the largest cost range across agents in any request column, and let $b^\star(k)$ be the marginal of an assignment minimizing the current learned linear cost. Then

$$\Delta_{\mathrm{marg}}(k) \le \tfrac{1}{2} K_k \mathcal{R}(k) \|b(k) - b^\star(k)\|_1, \qquad \Delta_{\mathrm{soft}}(k) \le B_{\mathrm{dec}}(k), \tag{13}$$

where (36) computes B_{dec} from cost ranges, target shares, and decoder parameters. These terms quantify sensitivity to target shares and decoder penalties under the current learned cost. Appendix B.4 gives the proof and the explicit total bound.

The auxiliary workload follows $Q_i(k+1) = [Q_i(k) - \mu_i(k) + A_i(k)]_+$ with $A_i(k) = \sum_j d_{ij}(k) Y_{ij}^{(k)}$ and service $\mu_i(k) \ge 0$. For a comparison assignment $Z \in \mathcal{Y}_{K_k}$, write $A_i^Z(k) = \sum_j d_{ij}(k) Z_{ij}$.

Assumption 1 requires bounded second moments, a feasible reference assignment with uniform negative drift, and $\Xi_{\mathrm{soft}}(k) \le \xi$ uniformly, together with a finite initial second moment.

Theorem 2 (Conditional stability of the workload process)**.** *Under Assumption 1, the workload process is strongly stable in the standard time-average sense (Neely, 2010):*

$$\limsup_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \mathbb{E}\Big[\sum_{i=1}^{M} Q_i(k)\Big] \le \frac{B+\xi}{\eta}. \tag{14}$$

The proof is presented in Appendix B.5, which applies the standard quadratic Lyapunov argument. Under Assumption 1, a uniformly bounded expected one-step cost gap preserves the negative drift of a reference dispatcher from the feasible assignment class. We complement our theoretical analysis with empirical evaluation, detailed in Appendix D.3.

6 Experiments

We compare OT-Dispatch Net with classical and learned baselines on request waiting time, makespan, and dispatch latency across fleet sizes and demand levels. We further assess transfer to larger fleets and NYC arrival traces without retraining. Ablations isolate the contributions of the learned agent marginal, transport coupling, and soft quota decoding to service quality and computational efficiency.

6.1 Experimental setup

Benchmarks. The synthetic benchmark initializes all agents at a central depot and releases planar requests over $T = 300$, with dispatch every $\Delta t = 30$. Requests follow homogeneous Poisson arrivals at rate $\lambda_r = |\mathcal{J}|/T$, where $|\mathcal{J}|$ is the expected request count. We denote demand by $\mathrm{AP} \triangleq |\mathcal{J}|/(MN_{\mathrm{arr}})$, the nominal number of requests per agent per arrival interval. Combining $M \in \{5, 10, 20\}$ with $\mathrm{AP} \in \{6, 9, 12\}$ defines nine settings, and we use $\lambda = 0.5$ in (1) and (6).

Baselines. We compare four classical and three learned dispatchers. Greedy assigns each request to its nearest available agent. Simulated annealing (SA) (Bräysy & Gendreau, 2005) and ant colony optimization (ACO) (Dorigo et al., 2006) search batch assignments under fixed wall-clock budgets, whereas rolling-horizon OR-Tools (Furnon & Perron, 2023) optimizes currently available requests. The learned iMTSP (Guo et al., 2024) policy processes the same request set before the shared routing backend replans the resulting routes. SRL (Hoppe et al., 2025) uses the official actor-critic pipeline with a DmTSP optimization layer and its best validation checkpoint. The learned cost minimum-cost flow baseline (L-MCF) uses the same cost-network architecture but is trained separately with deterministic min-cost-flow assignment (Ahuja et al., 1993).

Protocol and metrics. All methods receive matched arrivals at identical dispatch epochs and use the same post-assignment routing backend, implemented as an OR-Tools route-insertion routine that may change route order but not request ownership. OT-Dispatch Net executes one stochastic assignment ($R = 1$) or selects the assignment with the lowest post-routing objective among ten samples ($R = 10$). SA and ACO each receive 0.5 s per decision, with 16 ants for ACO, and every OR-Tools call has a 0.2 s limit. We report makespan $C_{\max}$, mean request waiting time (AvgWait), and dispatch latency, using AvgWait as the primary quality metric. Dispatch latency excludes the shared route-insertion call; Appendix D reports total decision latency. Baselines average 30 instances; OT-Dispatch Net reports mean and standard deviation (SD) across five training seeds.

Table 1: Synthetic results across nine settings. *Ours*: mean ± seed-level SD over five training seeds; baselines: means over 30 matched instances. Time is dispatch latency (ms).

$M = 5$

Method	AP = 6			AP = 9			AP = 12		
Metric	$C_{\max}\downarrow$	AvgWait ↓	Time ↓	$C_{\max}\downarrow$	AvgWait ↓	Time ↓	$C_{\max}\downarrow$	AvgWait ↓	Time ↓
ACO	**618.6**	167.3	215.9	795.9	238.2	239.5	839.6	272.5	277.4
SA	692.1	193.1	535.1	818.2	257.7	584.2	933.4	313.0	658.3
Greedy	863.8	282.2	**0.6**	1044.6	383.9	**0.9**	1192.6	467.0	**1.1**
OR-Tools	759.5	202.9	200.9	985.8	325.3	201.2	1086.9	366.7	201.4
iMTSP	847.3	225.0	1.8	972.3	278.8	1.9	1074.3	321.0	2.0
SRL	776.8	254.4	22.2	930.1	331.2	21.6	1067.5	404.4	28.7
L-MCF	696.5	185.5	2.4	770.3	232.7	2.5	845.8	268.8	2.7
Ours ($R = 1$)	653.2±11.5	162.5±2.2	3.8	**749.0±11.5**	**224.4±3.4**	4.0	789.8±10.5	249.2±1.6	4.1
Ours ($R = 10$)	648.7±10.3	**161.9±2.0**	52.7	749.6±11.9	224.5±3.2	78.5	**780.9±11.1**	**246.1±1.7**	114.6

$M = 10$

Method	AP = 6			AP = 9			AP = 12		
Metric	$C_{\max}\downarrow$	AvgWait ↓	Time ↓	$C_{\max}\downarrow$	AvgWait ↓	Time ↓	$C_{\max}\downarrow$	AvgWait ↓	Time ↓
ACO	701.3	176.5	258.8	805.3	221.6	353.9	838.8	252.2	461.2
SA	702.1	170.4	581.7	752.6	203.0	694.0	829.4	245.4	863.1
Greedy	819.8	247.9	**1.6**	1007.3	347.6	**2.5**	1165.3	434.2	**3.3**
OR-Tools	702.6	175.7	201.4	877.7	228.9	202.0	1150.3	365.5	501.3
iMTSP	777.4	189.9	6.0	946.8	238.8	6.4	1045.1	286.3	6.6
SRL	845.2	287.0	34.8	991.0	362.4	52.6	1147.5	429.5	93.6
L-MCF	703.1	165.0	4.8	774.5	199.2	5.3	820.2	250.6	5.4
Ours ($R = 1$)	672.6±14.2	156.4±3.3	4.9	726.1±6.1	187.7±2.9	5.4	787.0±14.9	217.8±3.9	5.5
Ours ($R = 10$)	**666.4±12.0**	**155.4±3.3**	75.9	**722.9±8.2**	**185.7±2.6**	123.3	**785.4±11.3**	**217.7±3.0**	190.7

$M = 20$ **(HTC from $M_0 = 5$)**

Method	AP = 6			AP = 9			AP = 12		
Metric	$C_{\max}\downarrow$	AvgWait ↓	Time ↓	$C_{\max}\downarrow$	AvgWait ↓	Time ↓	$C_{\max}\downarrow$	AvgWait ↓	Time ↓
ACO	794.2	191.3	435.1	1012.9	261.7	795.1	1745.8	423.2	1428.2
SA	700.1	128.6	704.3	643.3	154.1	1000.5	833.9	195.8	1452.8
Greedy	768.2	196.8	**5.4**	960.3	300.0	8.2	1122.3	382.4	10.8
OR-Tools	657.6	132.0	2002.1	897.8	181.0	4002.7	1004.4	315.0	4004.1
iMTSP	725.0	141.5	7.7	853.5	179.6	8.0	946.2	215.2	8.6
SRL	1123.5	393.6	106.1	1411.0	491.5	207.7	1664.3	584.7	394.8
L-MCF	645.7	124.8	6.4	735.1	157.2	**6.8**	772.6	188.4	7.5
Ours ($R = 1$)	603.2±9.7	121.1±1.5	6.4	639.8±8.6	144.4±1.9	6.9	735.5±7.3	**178.0±1.7**	**7.5**
Ours ($R = 10$)	**602.4±8.4**	**121.0±1.4**	113.6	**639.3±7.8**	**143.6±1.3**	201.0	**734.6±6.0**	178.1±1.5	329.9

6.2 Comparison results

Service quality. With $R = 1$, OT-Dispatch Net achieves lower mean AvgWait than all baselines across the nine synthetic settings in Table 1. Reductions against the best baseline per setting range from 2.9% to 11.2%. At the two training scales, the advantage increases with demand, from 2.9% to 7.3% for $M = 5$ and from 5.2% to 11.2% for $M = 10$. Thus, the waiting-time advantage persists and strengthens as more requests arrive per agent. These waiting-time gains generally do not sacrifice route completion: the $R = 1$ policy also outperforms all baselines in makespan in eight settings. Only at $(M, \mathrm{AP}) = (5, 6)$ does ACO achieve lower makespan but higher AvgWait.

Comparison with learned baselines. L-MCF uses the same cost-network architecture but is trained separately with deterministic min-cost-flow assignment. Compared with this baseline, the $R = 1$ policy reduces AvgWait by 3.0%-13.1% across all nine settings, with comparable dispatch latency (3.8-7.5 versus 2.4-7.5 ms). These results show improved waiting-time performance over a separately trained flow-based policy with a similar computational cost. The $R = 1$ policy also reduces AvgWait by 14.4%-27.8% relative to iMTSP and 32.2%-70.6% relative to SRL. The SRL comparison concerns our DmTSP adaptation, whose optimization layer permits deferred assignment.

Dispatch efficiency and scale transfer. Increasing the sample count from $R = 1$ to $R = 10$ yields small, nonuniform changes in service quality while increasing dispatch latency from 3.8-7.5 ms to 52.7-329.9 ms. The largest mean AvgWait reduction from ten-sample selection is only 1.24%, and two settings show slight increases. Thus, the baseline advantage is already present with a single sampled assignment. This trade-off supports $R = 1$ as the default, with $R = 10$ reserved for deployments that permit additional decoding and route evaluation. With hierarchical task clustering,

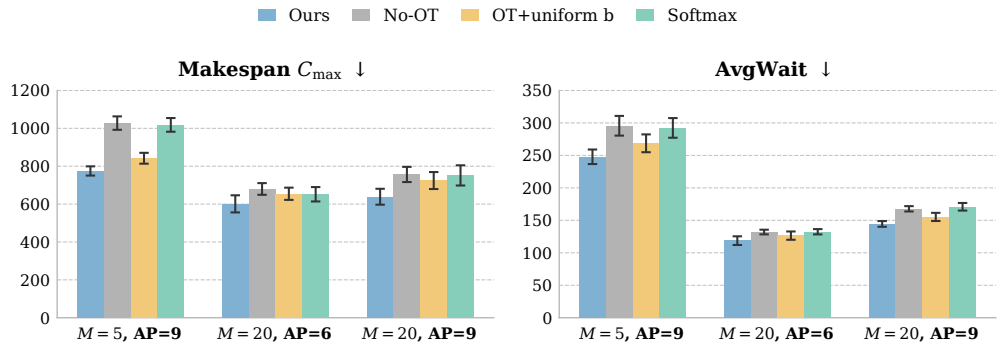

Figure 2: Component ablation at $(M, \mathrm{AP}) \in \{(5, 9), (20, 6), (20, 9)\}$. Bars and error bars show means and instance-level sample SDs over 30 matched instances. *No OT* assigns greedily; *OT+uniform b* fixes b; *Softmax* replaces Sinkhorn with columnwise softmax.

the policy trained at $M_0 = 5$ transfers to $M = 20$ without retraining. At $R = 1$, it outperforms all baselines in AvgWait and makespan at each demand level; latency includes clustering.

NYC replay. On chronologically held-out NYC arrival traces, frozen policies achieve the lowest mean AvgWait across 30 matched replays per fleet size (Table 2; Appendix E.6). Reductions relative to the strongest baseline are 12.1%, 11.3%, and 2.3% for $M = 5, 10, 20$, respectively. OT-Dispatch Net also achieves the lowest mean makespan for $M = 10$ and $M = 20$, whereas ACO performs better for $M = 5$. These results support zero-shot transfer to real pickup traces.

6.3 Ablation Study

Figure 2 compares transport coupling and the learned agent marginal. No OT uses nearest-agent assignment as a deterministic reference. OT+uniform b fixes $b_i = 1/M$; Softmax replaces OT with columnwise probabilities while retaining soft-quota decoding. Appendix E.4 gives training details.

Across all three settings, the full model achieved lower AvgWait and makespan than every variant (Figure 2). OT+uniform b was the strongest variant, suggesting that transport coupling remains useful with a fixed marginal. Its gap from the full model supports learning the agent marginal, while Softmax's poorer performance supports retaining OT constraints. The probability maps likewise show that OT concentrates assignment probability on fewer agents than Softmax (Figure 3).

At $(M, \mathrm{AP}) = (5, 12)$, soft-quota decoding matched Hungarian projection in AvgWait (256.84 versus 257.33) and makespan, but cut dispatch latency by 75.8% (Table 4). Gumbel-Sinkhorn and independent sampling had similar latency but much higher AvgWait and makespan. Thus, soft quota preserved quality comparable to Hungarian projection at lower latency.

7 Conclusion

We introduced OT-Dispatch Net for DmTSP to make request allocation across agents an explicit policy decision rather than an implicit consequence of individual assignments. This explicit control allows the policy to redistribute newly arriving requests as routes and agent availability evolve, thereby accounting for the coupling between current allocation and future service. Soft-quota decoding produces executable joint assignments while preserving policy gradient learning through the marginal. Within the bilevel framework, potential-based rewards connect dispatch decisions to routing and service outcomes over complete episodes. Our analysis bounds assignment cost errors and establishes conditional stability under an auxiliary workload model. Experiments across synthetic and NYC replay settings show lower request waiting time and generally lower makespan than the evaluated baselines. Together, these results support learning request allocation alongside individual assignment preferences as routes and agent availability evolve. Future work will extend the framework to agents with different speeds and service capacities, and to road networks with travel times that vary over time. Another direction is to connect the auxiliary workload model to remaining route duration and derive stability conditions that account for route updates and changing demand.

AI USE STATEMENT

In this work, we used generative AI tools, specifically OpenAI Codex, for consistency checking of mathematical proofs. Additionally, we used OpenAI Codex for LaTeX migration, consistency checks of mathematical notation, language polishing, and formatting checks. We have reviewed all AI-assisted work. In particular, the authors independently verified all mathematical claims and proof steps. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the information needed to reproduce the formulation, implementation, and evaluation of OT-Dispatch Net throughout the paper and appendices. Section 3 specifies the DmTSP objective and sequential environment, while Section 4 and Algorithm 1 define the policy and decoding procedure. Appendix E reports the architecture, training objective, hyperparameters, decoder ordering, and ablation configurations. Appendix E.6 documents synthetic instance generation, NYC data processing and splits, evaluation seeds, and checkpoint protocol. Section 6.1 specifies the matched instances, shared routing backend, evaluation metrics, and dispatch latency convention used to compare methods. Complete assumptions and proofs are given in Appendices B and E.5; Appendix C details test-time adaptation, while Appendix D reports additional comparisons, ablations, and latency measurements.

REFERENCES

Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ, 1993. ISBN 9780136175490.

Júlia Baligács, Yann Disser, Farehe Soheil, and David Weckbecker. Tight analysis of the lazy algorithm for open online dial-a-ride. In *Algorithms and Data Structures*, volume 14079, pp. 43–64, 2023.

Tolga Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, 2006.

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 Track*, 2017.

Russell W Bent and Pascal Van Hentenryck. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987, 2004.

Federico Berto, Chuanbo Hua, Laurin Luttmann, Jiwoo Son, Junyoung Park, Kyuree Ahn, Changhyun Kwon, Lin Xie, and Jinkyoo Park. PARCO: Parallel autoregressive models for multi-agent combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 38, pp. 167130–167160, 2025.

Mathieu Blondel, André F. T. Martins, and Vlad Niculae. Learning with Fenchel–Young losses. *Journal of Machine Learning Research*, 21(35):1–69, 2020.

Olli Bräysy and Michel Gendreau. Vehicle routing problem with time windows, part II: metaheuristics. *Transportation Science*, 39(1):119–139, 2005.

Yuhong Cao, Zhanhong Sun, and Guillaume Sartoretti. DAN: decentralized attention-based neural network for the minmax multiple traveling salesman problem. In *Distributed Autonomous Robotic Systems*, volume 28, pp. 202–215, 2024.

Dawei Chen, Christina Imdahl, David Lai, and Tom Van Woensel. The dynamic traveling salesman problem with time-dependent and stochastic travel times: A deep reinforcement learning approach. *Transportation Research Part C: Emerging Technologies*, 172:105022, 2025.

Taesu Cheong and Chelsea C White. Dynamic traveling salesman problem: Value of real-time traffic information. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):619–630, 2012.

Marco Cuturi. Sinkhorn distances: lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, pp. 2292–2300, 2013.

Marco Dorigo, Mauro Birattari, and Thomas Stützle. Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4):28–39, 2006.

Tobias Enders, James Harrison, Marco Pavone, and Maximilian Schiffer. Hybrid multi-agent deep reinforcement learning for autonomous mobility on demand systems. In *Proceedings of the 5th Annual Learning for Dynamics and Control Conference*, volume 211, pp. 1284–1296, 2023.

Vincent Furnon and Laurent Perron. OR-Tools Routing Library. Version 9.8, 2023. URL https://developers.google.com/optimization/routing/.

Daniele Gammelli, James Harrison, Kaidi Yang, Marco Pavone, Filipe Rodrigues, and Francisco C. Pereira. Graph reinforcement learning for network control via bi-level optimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 10587–10610, 2023.

Yong Liang Goh, Wee Sun Lee, Xavier Bresson, Thomas Laurent, and Nicholas Lim. Combining reinforcement learning and optimal transport for the traveling salesman problem. *arXiv preprint arXiv:2203.00903*, 2022.

Yifan Guo, Zhongqiang Ren, and Chen Wang. iMTSP: Solving min-max multiple traveling salesman problem with imperative learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 10245–10252, 2024.

Heiko Hoppe, Tobias Enders, Quentin Cappart, and Maximilian Schiffer. Global rewards in multi-agent deep reinforcement learning for autonomous mobility on demand systems. In *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*, volume 242, pp. 260–272, 2024.

Heiko Hoppe, Léo Baty, Louis Bouvier, Axel Parmentier, and Maximilian Schiffer. Structured reinforcement learning for combinatorial decision-making. In *Advances in Neural Information Processing Systems*, volume 38, pp. 173726–173761, 2025.

Yujiao Hu, Yuan Yao, and Wee Sun Lee. A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs. *Knowledge-Based Systems*, 204:106244, 2020.

Koray Gabriel Kachar and Alex Arkady Gorodetsky. Dynamic multiagent assignment via discrete optimal transport. *IEEE Transactions on Control of Network Systems*, 9(1):151–162, 2022.

Elias B. Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6348–6358, 2017.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

Harold W Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, 1955.

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.

Dechao Lei and Yuanshan Wu. Order-dispatching strategy induced by optimal transport plan for an online ride-hailing system. *Transportation Research Record*, 2676(6):156–169, 2022.

Laurin Luttmann and Lin Xie. Multi-action self-improvement for neural combinatorial optimization. In *International Conference on Learning Representations*, 2026.

Chryssi Malandraki and Mark S Daskin. Time dependent vehicle routing problems: formulations, properties and heuristic algorithms. *Transportation Science*, 26(3):185–200, 1992.

Björn Martens. A bi-level approach for a dynamic multiple traveling salesman problem. *Journal of Optimization Theory and Applications*, 207(3):47, 2025.

Gonzalo E. Mena, David Belanger, Scott W. Linderman, and Jasper Snoek. Learning latent permutations with Gumbel–Sinkhorn networks. In *International Conference on Learning Representations*, 2018.

Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takac. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*, volume 31, pp. 9839–9849, 2018.

Michael J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*, volume 3 of *Synthesis Lectures on Communication Networks*. Morgan & Claypool Publishers, 2010. ISBN 9781608454556.

Gabriel Peyré and Marco Cuturi. Computational optimal transport: with applications to data science. *Foundations and Trends in Machine Learning*, 11(5–6):355–607, 2019.

Zhiwei Tony Qin, Xiaocheng Tang, Yan Jiao, Fan Zhang, Zhe Xu, Hongtu Zhu, and Jieping Ye. Ride-hailing order dispatching at DiDi via reinforcement learning. *INFORMS Journal on Applied Analytics*, 50(5):272–286, 2020.

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.

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

Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

Jiwoo Son, Minsu Kim, Sanghyeok Choi, Hyeonah Kim, and Jinkyoo Park. Equity-Transformer: solving NP-hard min-max routing problems as sequential generation with equity context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 20265–20273, 2024.

Stacy A Voccia, Ann Melissa Campbell, and Barrett W Thomas. The same-day delivery problem for online purchases. *Transportation Science*, 53(1):167–184, 2019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Zhe Xu, Zhixin Li, Qingwen Guan, Dingshui Zhang, Qiang Li, Junxiao Nan, Chunyang Liu, Wei Bian, and Jieping Ye. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 905–913, 2018.

Zizhen Zhang, Hong Liu, MengChu Zhou, and Jiahai Wang. Solving dynamic traveling salesman problems with deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(4):2119–2132, 2023.

A Entropic optimal transport background

This appendix reviews the entropic transport formulation underlying Section 4.3.

Consider an agent marginal $b \in \Delta^M$, a request marginal $a \in \Delta^K$, and a finite cost matrix $C \in \mathbb{R}^{M\times K}$. Here, K denotes the batch size and equals $K_k = |\mathcal{B}_k|$ at dispatch epoch k. We suppress the epoch index when it is unnecessary. The Kantorovich problem minimizes the linear cost over nonnegative plans with these marginals (Peyré & Cuturi, 2019):

$$\begin{aligned} T^\star &= \arg\min_{T\in\Pi(b,a)} \langle C, T\rangle, \\ \Pi(b,a) &= \left\{T \in \mathbb{R}^{M\times K}_{\geq 0} \;\middle|\; T\mathbf{1}_K = b,\ T^\top \mathbf{1}_M = a\right\}. \end{aligned} \tag{15}$$

The feasible set $\Pi(b,a)$ is the transport polytope, with rows representing agents and columns representing requests. Its constraints require row i to sum to b_i and column j to sum to a_j. Here, b_i is agent i's target share of the current batch and a_j is request j's transport mass. For uniform $a_j = 1/K$, the corresponding target count is Kb_i; discrete decoding need not realize this count.

For positive marginals, a strictly positive feasible plan is

$$T^0 = ba^\top > 0, \qquad T^0\mathbf{1}_K = b, \qquad (T^0)^\top \mathbf{1}_M = a. \tag{16}$$

For positive marginals and $\varepsilon > 0$, entropic regularization gives a unique plan of the form $T^\star_\varepsilon = \mathrm{diag}(u)K_\varepsilon \mathrm{diag}(v)$, where $K_\varepsilon = \exp(-C/\varepsilon)$ (Cuturi, 2013). The objective and alternating scaling updates are

$$T^\star_\varepsilon = \arg\min_{T\in\Pi(b,a)} \{\langle C, T\rangle - \varepsilon H(T)\}, \quad H(T) = -\sum_{i,j} T_{ij}\log T_{ij}, \tag{17}$$

$$u^{(t+1)} = \frac{b}{K_\varepsilon v^{(t)}}, \qquad v^{(t+1)} = \frac{a}{K_\varepsilon^\top u^{(t+1)}}. \tag{18}$$

For finite costs, the kernel K_ε is strictly positive. Starting from $v^{(0)} = \mathbf{1}$, the updates preserve positive scaling vectors and hence give $\widehat{T}_{ij} > 0$ after every finite iteration in exact arithmetic. Under these conditions, the iterates converge to the unique entropic solution, although a finite-iteration plan need not satisfy both marginals exactly.

Each iteration consists of differentiable matrix scaling, with divisions performed elementwise. After N_{SK} iterations, the resulting plan $\widehat{T}$ approximates the entropic solution. Automatic differentiation propagates gradients through these iterations to the encoder and both policy heads. This path remains available when PPO evaluates the soft-quota log probability of a stored assignment; the sampling operation itself is not differentiated.

B Theory: OT relaxation and stability

This appendix develops the policy properties and cost comparisons used in Section 5, followed by the conditional stability proof. Unless stated otherwise, we consider a fixed dispatch epoch and use the convention $0\log 0 = 0$. Positivity and differentiability statements are understood in exact arithmetic.

B.1 Soft-quota policy properties

Consider a nonempty batch of size K with observation o, agent marginal $b \in \mathrm{int}(\Delta^M)$, and positive finite-iteration plan $\widehat{T}$. Each assignment $Y \in \mathcal{Y}_K$ determines a sequence $(I_1, \ldots, I_K)$ in canonical request order. For the preceding agent choices $I_{<j}$, let $n_i^{j-1} = \sum_{t<j} \mathbb{I}[I_t = i]$ count requests

assigned to agent i within this batch. These history counts are not the pending route counts $n_i(k)$. With finite $\alpha, \delta > 0$, the base probabilities, conditional probabilities, and joint policy are

$$P_{ij} := \frac{\widehat{T}_{ij}}{\sum_{r=1}^{M} \widehat{T}_{rj}}, \tag{19}$$

$$p_{ij} := \frac{P_{ij} \exp[-\alpha \operatorname{softplus}(n_i^{j-1} + 1 - Kb_i - \delta)]}{\sum_{r=1}^{M} P_{rj} \exp[-\alpha \operatorname{softplus}(n_r^{j-1} + 1 - Kb_r - \delta)]}, \tag{20}$$

$$\pi_\theta(Y \mid o) := \prod_{j=1}^{K} p_{I_j j}. \tag{21}$$

Proposition 3 (Feasibility, common support, and differentiability)**.** *Suppose the pairwise scores are finite, $b \in \operatorname{int}(\Delta^M)$, and Sinkhorn uses a positive kernel and positive initial scaling. The soft-quota decoder returns $Y \in \mathcal{Y}_K$ almost surely and assigns positive probability to every $Y \in \mathcal{Y}_K$. For any fixed replayed action, its log probability is differentiable with respect to the pairwise scores and b. Any two admissible parameter values thus share support $\mathcal{Y}_K$ and yield a finite PPO likelihood ratio for every stored action.*

Proof of Proposition 3. Positive conditionals and feasibility. Positive agent marginals and finite pairwise scores ensure that the Sinkhorn kernel and all scaling vectors remain strictly positive. Every finite iteration preserves positivity, giving $\widehat{T}_{ij} > 0$ and, by (19), $P_{ij} > 0$. Finite counts and penalty parameters make every corrected logit in (20) finite. It follows that

$$0 < p_{ij} < 1 \qquad \text{for every } i, j \text{ and every history } I_{<j} \tag{22}$$

when $M > 1$; if $M = 1$, the sole probability equals one. Each categorical draw assigns exactly one agent to the current request, so the resulting action lies in $\mathcal{Y}_K$. Conversely, every $Y \in \mathcal{Y}_K$ determines a sequence whose probability in (21) is a finite product of positive terms. Hence the policy has full support $\mathcal{Y}_K$ for every admissible parameter value.

Differentiability and common support. For a stored action, the indicators defining the history counts n_i^{j-1} are fixed. Finite Sinkhorn scaling, column normalization, softplus, and log-softmax are differentiable on their respective domains. Their composition makes the joint log probability differentiable with respect to the scores and b in the simplex interior. Both policies assign positive probability to the stored action, so $\pi_\theta(Y \mid o)/\pi_{\theta_{\text{old}}}(Y \mid o)$ is finite. This is the joint-action likelihood ratio used in PPO rollout and replay. □

The following result bounds the probability of further assignments once an agent's assigned count exceeds its continuous target count.

Proposition 4 (Assignment probability beyond the target count)**.** *Fix a decoder step j and suppose the base log probabilities satisfy*

$$|\log P_{ij} - \log P_{rj}| \le G \qquad \textit{for all } i, r \tag{23}$$

for some finite G. If $n_i^{j-1} - Kb_i \ge r$ for an agent i and some $r \ge 0$, then

$$p_{ij} \le \min\{1, \exp\big(G - \alpha\big[\operatorname{softplus}(r + 1 - \delta) - \operatorname{softplus}(1 - \delta)\big]\big)\}. \tag{24}$$

Proof. The count identity $\sum_h n_h^{j-1} = j - 1 < K = \sum_h Kb_h$ gives an agent h with $n_h^{j-1} < Kb_h$. Monotonicity of softplus and cancellation of the conditional normalizer yield

$$\begin{aligned} &\psi_h^j < \operatorname{softplus}(1 - \delta), \qquad \psi_i^j \ge \operatorname{softplus}(r + 1 - \delta), \\ &\frac{p_{ij}}{p_{hj}} = \frac{P_{ij}}{P_{hj}} \exp[-\alpha(\psi_i^j - \psi_h^j)] \\ &\qquad \le \exp(G - \alpha[\operatorname{softplus}(r + 1 - \delta) - \operatorname{softplus}(1 - \delta)]). \end{aligned}$$

The last step uses (23). Multiplying by $p_{hj} \le 1$ and applying $p_{ij} \le 1$ proves (24). The softplus difference grows linearly for large r, giving exponential decay at rate $\exp(-\alpha r)$ up to a constant factor. □

Global occupancy regularizer. Here, occupancy means the number of requests assigned within the current batch, $N_i(Y) = \sum_j Y_{ij}$, not the agent's existing route workload. Define the accumulated penalty and its increment by

$$\Phi_i(N; b_i) := \sum_{r=1}^{N} \mathrm{softplus}(r - Kb_i - \delta),$$
$$\Omega_b(Y) := \sum_{i=1}^{M} \Phi_i(N_i(Y); b_i), \tag{25}$$
$$\Phi_i(n_i^{j-1} + 1; b_i) - \Phi_i(n_i^{j-1}; b_i) = \psi_i^j. \tag{26}$$

Thus each decoder penalty is the marginal increase in Ω_b. Request order and sampling determine the sequential policy, with their cost effects included in Δ_{soft}.

B.2 Agent-marginal and entropy comparisons

Fix $a = K^{-1}\mathbf{1}_K$ and a finite cost matrix C. Let $T_0^\star(b)$ and $T_\varepsilon^\star(b)$ minimize the unregularized and entropic objectives, respectively, over the same polytope $\Pi(b, a)$.

Proposition 5 (Cost of the agent marginal)**.** *For every finite $C \in \mathbb{R}^{M\times K}$ and $b \in \mathrm{int}(\Delta^M)$,*

$$\Delta_{\mathrm{marg}} := K\langle C, T_0^\star(b)\rangle - \min_{Z\in\mathcal{Y}_K} \langle C, Z\rangle \geq 0. \tag{27}$$

The corresponding entropic solution satisfies the following upper bound on linear cost:

$$K\langle C, T_\varepsilon^\star(b)\rangle \leq \min_{Z\in\mathcal{Y}_K} \langle C, Z\rangle + \Delta_{\mathrm{marg}} + K\varepsilon \log M. \tag{28}$$

Proof. For any $T \in \Pi(b, a)$, each column of $X = KT$ is a probability vector. Its cost is a convex combination of the agent costs for that request, giving

$$K\langle C, T\rangle = \sum_{j=1}^{K}\sum_{i=1}^{M} C_{ij}X_{ij} \geq \sum_{j=1}^{K} \min_i C_{ij} = \min_{Z\in\mathcal{Y}_K} \langle C, Z\rangle. \tag{29}$$

Substituting $T_0^\star(b)$ into this inequality establishes the nonnegativity claimed in (27).

For every feasible plan, $X = KT$ has probability-vector columns, so

$$H(T) = \log K + \frac{1}{K}\sum_j H(X_{\cdot j}), \qquad 0 \leq H(X_{\cdot j}) \leq \log M.$$

Combining this entropy range with optimality of $T_\varepsilon^\star$ gives the continuous comparison and its scaled assignment bound:

$$\begin{aligned}\langle C, T_\varepsilon^\star\rangle &\leq \langle C, T_0^\star\rangle + \varepsilon[H(T_\varepsilon^\star) - H(T_0^\star)]\\ &\leq \langle C, T_0^\star\rangle + \varepsilon \log M,\\ K\langle C, T_\varepsilon^\star\rangle &\leq \min_{Z\in\mathcal{Y}_K} \langle C, Z\rangle + \Delta_{\mathrm{marg}} + K\varepsilon \log M.\end{aligned}$$

The final line uses (27) and establishes (28). □

Proposition 6 (Sensitivity to the agent marginal)**.** *Fix a finite cost matrix C and $a = K^{-1}\mathbf{1}_K$. Define*

$$V_C(b) := \min_{T\in\Pi(b,a)} \langle C, T\rangle, \qquad \mathcal{R}_C := \max_j\big(\max_i C_{ij} - \min_i C_{ij}\big).$$

For any b, b' in the closed probability simplex,

$$|V_C(b) - V_C(b')| \leq \frac{\mathcal{R}_C}{2}\|b - b'\|_1. \tag{30}$$

Let $Y^\star$ minimize $\langle C, Y\rangle$ over $\mathcal{Y}_K$ and set $b^\star = K^{-1}Y^\star\mathbf{1}_K$. Then

$$0 \leq \Delta_{\mathrm{marg}} \leq \frac{K\mathcal{R}_C}{2}\|b - b^\star\|_1. \tag{31}$$

Proof. Take an optimal plan for b. In each row with $b_i > b'_i$, remove mass $b_i - b'_i$, distributing the removal among its columns without making any entry negative. Reassign the removed mass within the same columns to rows with $b_i < b'_i$ until each deficit is filled. Total surplus equals total deficit, $m = \frac{1}{2}\|b - b'\|_1$, so this construction gives a plan with row marginal b' and unchanged column marginal a. Moving mass within column j increases cost by at most its mass times $\max_i C_{ij} - \min_i C_{ij}$. Thus $V_C(b') \le V_C(b) + m\mathcal{R}_C$. Interchanging b and b' proves (30). The plan $Y^\star/K$ is feasible for $b^\star$, and (29) implies $KV_C(b^\star) = \langle C, Y^\star\rangle$. Substitution gives (31). □

For fixed C, Equation (30) quantifies the sensitivity of the optimal transport cost to the agent marginal. The comparison accepts any marginal $b^\star$ induced by an optimal assignment for the current linear cost, including marginals on the simplex boundary.

B.3 Explicit cost bound for soft-quota decoding

Fix an observation, a positive plan $\widehat{T}$, and the canonical request order. Let P be its column normalization in (19). For a finite learned cost matrix $\widehat{C}$, define

$$\begin{aligned} \mathcal{R}_j &:= \max_i \widehat{C}_{ij} - \min_i \widehat{C}_{ij}, & D_j &:= \max_i \psi_i^j - \min_i \psi_i^j, \\ b_{\min} &:= \min_i b_i, & b_{\max} &:= \max_i b_i, \\ \overline{D}_j &:= \operatorname{softplus}(j - Kb_{\min} - \delta) - \operatorname{softplus}(1 - Kb_{\max} - \delta). \end{aligned} \tag{32}$$

Here $\psi_i^j = \operatorname{softplus}(n_i^{j-1} + 1 - Kb_i - \delta)$ depends on the sampled history, whereas $\overline{D}_j$ does not. Since $0 \le n_i^{j-1} \le j - 1$, every history satisfies $0 \le D_j \le \overline{D}_j$.

Proposition 7 (Cost deviation under soft-quota decoding)**.** *Under the conditions of Proposition 3, let $p_{\cdot j}$ be the conditional distribution in (20) and define $\mathrm{TV}(p, P) := \frac{1}{2}\sum_i |p_i - P_i|$. The conditional deviation for every history and the resulting expected cost deviation satisfy*

$$\mathrm{TV}(p_{\cdot j}, P_{\cdot j}) \le \tanh(\alpha D_j/4), \tag{33}$$

$$\begin{aligned} \left|\mathbb{E}\langle \widehat{C}, Y\rangle - \langle \widehat{C}, P\rangle\right| &\le \sum_{j=1}^{K} \mathcal{R}_j\, \mathbb{E}\tanh(\alpha D_j/4) \\ &\le \sum_{j=1}^{K} \mathcal{R}_j \tanh(\alpha \overline{D}_j/4). \end{aligned} \tag{34}$$

Expectations are over decoder sampling conditional on the observation.

Proof. Fix j and its history, and write $w_i = \exp(-\alpha\psi_i^j)$, $l = \min_i w_i$, $u = \max_i w_i$, and $z = \sum_i P_{ij} w_i$. Then $p_{ij} = P_{ij} w_i / z$. If $l = u$, the distributions coincide. Otherwise, convexity of $|w - z|$ on $[l, u]$ bounds it by the line joining its endpoint values. Averaging this bound under $P_{\cdot j}$ gives

$$\begin{aligned} \mathrm{TV}(p_{\cdot j}, P_{\cdot j}) &= \frac{1}{2z}\sum_i P_{ij}|w_i - z| \\ &\le \frac{(u - z)(z - l)}{z(u - l)} \le \frac{\sqrt{u} - \sqrt{l}}{\sqrt{u} + \sqrt{l}} = \tanh(\alpha D_j/4). \end{aligned} \tag{35}$$

The second inequality follows by maximizing over $z \in [l, u]$, with maximum at $z = \sqrt{lu}$. Subtracting $\min_i \widehat{C}_{ij}$ from all costs in column j does not change the difference of their expectations. The resulting costs lie in $[0, \mathcal{R}_j]$, hence

$$\left|\sum_i \widehat{C}_{ij}(p_{ij} - P_{ij})\right| \le \mathcal{R}_j\, \mathrm{TV}(p_{\cdot j}, P_{\cdot j}).$$

Take expectations over histories and sum over requests, using the tower property and the triangle inequality. Monotonicity of $\tanh$ and $D_j \le \overline{D}_j$ complete the proof. □

To compare with the finite transport plan, define the computable normalization term and decoder bound

$$E_{\mathrm{norm}} := |\langle \widehat{C}, P - K\widehat{T}\rangle|,$$
$$B_{\mathrm{dec}} := E_{\mathrm{norm}} + \sum_{j=1}^{K} \mathcal{R}_j \tanh(\alpha \overline{D}_j/4). \tag{36}$$

The triangle inequality gives $\Delta_{\mathrm{soft}} \le B_{\mathrm{dec}}$. When $\widehat{T}^\top \mathbf{1}_M = K^{-1}\mathbf{1}_K$, column normalization gives $P = K\widehat{T}$ and $E_{\mathrm{norm}} = 0$. This holds for at least one exact Sinkhorn iteration ending in column scaling with the prescribed request marginal. Numerical offsets or approximate normalization require retaining E_{norm}.

The bound identifies how the penalty scale, target shares, batch size, and cost ranges control deviation from the base assignment probabilities. With exact column normalization, the bound approaches zero as $\alpha \to 0$. The history-dependent bound in (34) further accounts for the penalty spread at each decoding step. Both bounds measure deviation under the learned linear cost.

B.4 Proof of Theorem 1

At epoch k, let $T_0^\star(k;b)$ and $T_\varepsilon^\star(k;b)$ denote the unregularized and entropic solutions in $\Pi(b(k), a(k))$. Let $\widehat{T}(k)$ denote the finite-iteration plan. The four residuals in (11) are defined by

$$\Delta_{\mathrm{marg}}(k) := K_k \langle \widehat{C}(k), T_0^\star(k;b)\rangle - \min_{Z \in \mathcal{Y}_{K_k}} \langle \widehat{C}(k), Z\rangle, \tag{37}$$
$$\Delta_{\mathrm{sink}}(k) := K_k \big[\langle \widehat{C}(k), \widehat{T}(k)\rangle - \langle \widehat{C}(k), T_\varepsilon^\star(k;b)\rangle\big]_+, \tag{38}$$
$$\Delta_{\mathrm{soft}}(k) := \big[\mathbb{E}[\langle \widehat{C}(k), Y^{(k)}\rangle \mid \mathcal{F}_k] - K_k \langle \widehat{C}(k), \widehat{T}(k)\rangle\big]_+, \tag{39}$$
$$\Delta_{\mathrm{cal}}(k) := \sup_{Z \in \mathcal{Y}_{K_k}} \left|\langle \widehat{C}(k) - C^{\mathrm{drift}}(k), Z\rangle\right|. \tag{40}$$

All four residuals are nonnegative. Proposition 5 establishes this property for Δ_{marg}; positive parts define Δ_{sink} and Δ_{soft}, while Δ_{cal} uses an absolute value.

Proof. Fix k and abbreviate $K = K_k$, $C^d = C^{\mathrm{drift}}(k)$, $\widehat{C} = \widehat{C}(k)$, $Y = Y^{(k)}$, and $T_\varepsilon = T_\varepsilon^\star(k;b)$. Suppress k on the residuals and transport plan. Feasibility of Y and the uniform calibration bound, followed by the decoder, Sinkhorn, and entropy comparisons, yield

$$\mathbb{E}[\langle C^d, Y\rangle \mid \mathcal{F}_k] \le \mathbb{E}[\langle \widehat{C}, Y\rangle \mid \mathcal{F}_k] + \Delta_{\mathrm{cal}} \tag{41}$$
$$\le K\langle \widehat{C}, \widehat{T}\rangle + \Delta_{\mathrm{soft}} + \Delta_{\mathrm{cal}} \tag{42}$$
$$\le K\langle \widehat{C}, T_\varepsilon\rangle + \Delta_{\mathrm{sink}} + \Delta_{\mathrm{soft}} + \Delta_{\mathrm{cal}} \tag{43}$$
$$\le \min_{Z \in \mathcal{Y}_K} \langle \widehat{C}, Z\rangle + K\varepsilon \log M + \Delta_{\mathrm{marg}} + \Delta_{\mathrm{sink}} + \Delta_{\mathrm{soft}} + \Delta_{\mathrm{cal}}. \tag{44}$$

The first three inequalities use the residual definitions in (40), (39), and (38); the last uses Proposition 5. For every $Z \in \mathcal{Y}_K$, calibration also gives $\langle \widehat{C}, Z\rangle \le \langle C^d, Z\rangle + \Delta_{\mathrm{cal}}$. Minimizing over Z and substituting into the chain gives

$$\min_{Z \in \mathcal{Y}_K} \langle \widehat{C}, Z\rangle \le \min_{Z \in \mathcal{Y}_K} \langle C^d, Z\rangle + \Delta_{\mathrm{cal}}, \tag{45}$$
$$\mathbb{E}[\langle C^d, Y\rangle \mid \mathcal{F}_k] \le \min_{Z \in \mathcal{Y}_K} \langle C^d, Z\rangle + \Xi_{\mathrm{soft}}(k),$$

where $\Xi_{\mathrm{soft}}(k)$ combines the four residual terms and the entropy contribution. □

For each epoch, apply Proposition 6 with $C = \widehat{C}(k)$ and Proposition 7 conditional on $\mathcal{F}_k$. Theorem 1 also holds with $\Xi_{\mathrm{soft}}(k)$ replaced by

$$\overline{\Xi}_{\mathrm{soft}}(k) := K_k \varepsilon \log M + \tfrac{1}{2} K_k \mathcal{R}(k) \|b(k) - b^\star(k)\|_1 + \Delta_{\mathrm{sink}}(k) + B_{\mathrm{dec}}(k) + 2\Delta_{\mathrm{cal}}(k). \tag{46}$$

Here $\mathcal{R}(k) = \max_j(\max_i \widehat{C}_{ij}(k) - \min_i \widehat{C}_{ij}(k))$. A minimizing assignment defining $b^\star(k)$ is obtained by selecting a minimum-cost agent in each column, with fixed tie-breaking. Thus the marginal and decoder terms can be bounded without evaluating the expected decoded cost. The remaining terms account for finite Sinkhorn computation and the conversion from learned costs to drift costs. A uniform bound on $\overline{\Xi}_{\mathrm{soft}}$ is sufficient for the error assumption in Theorem 2, since $\Xi_{\mathrm{soft}} \le \overline{\Xi}_{\mathrm{soft}}$.

B.5 Auxiliary workload model and proof of Theorem 2

Auxiliary workload model. Let $\mathcal{F}_k$ contain the information available immediately before dispatch. The auxiliary queues $Q_i(k)$, increments $d_{ij}(k)$, and batch size K_k are nonnegative and $\mathcal{F}_k$-measurable. Service satisfies $\mu_i(k) \ge 0$, with conditional law unchanged by the comparison assignment. Define

$$L(Q(k)) := \tfrac{1}{2}\sum_i Q_i(k)^2, \tag{47}$$

$$Q_i(k+1) = [Q_i(k) - \mu_i(k) + A_i(k)]_+, \quad A_i(k) := \sum_j d_{ij}(k) Y_{ij}^{(k)}, \tag{48}$$

$$A_i^Z(k) := \sum_j d_{ij}(k) Z_{ij}, \tag{49}$$

$$\mathcal{A}_k := \mathcal{Y}_{K_k}. \tag{50}$$

Assumption 1 (Moment, capacity, and approximation conditions)**.** *Consider an infinite-horizon extension using the soft-quota policy at every nonempty epoch. For constants $B < \infty$, $\eta > 0$, and $\xi < \infty$, assume that for every k there is an $\mathcal{F}_k$-measurable reference assignment $\widetilde{Y}(k) \in \mathcal{A}_k$ such that*

$$\mathbb{E}[L(Q(0))] < \infty, \tag{51}$$

$$\tfrac{1}{2}\sum_i \mathbb{E}[\mu_i(k)^2 + A_i(k)^2 \mid \mathcal{F}_k] \le B, \tag{52}$$

$$\sum_i Q_i(k) A_i^{\widetilde{Y}}(k) \le \sum_i Q_i(k)\mathbb{E}[\mu_i(k) \mid \mathcal{F}_k] - \eta \sum_i Q_i(k), \tag{53}$$

$$\Xi_{\mathrm{soft}}(k) \le \xi \quad \textit{almost surely}. \tag{54}$$

The moment conditions control the quadratic drift, the reference assignment provides a uniform service margin, and the approximation bound controls the assignment cost gap. All residuals are $\mathcal{F}_k$-measurable, with the decoder residual defined through a conditional expectation.

Proof of Theorem 2. Quadratic drift. Applying $([x]_+)^2 \le x^2$ to the workload recursion gives, for each agent i,

$$\begin{aligned} Q_i(k+1)^2 &= \big([Q_i(k) - \mu_i(k) + A_i(k)]_+\big)^2 \\ &\le \big(Q_i(k) - \mu_i(k) + A_i(k)\big)^2 \\ &= Q_i(k)^2 + \mu_i(k)^2 + A_i(k)^2 + 2Q_i(k)\big(A_i(k) - \mu_i(k)\big) - 2\mu_i(k)A_i(k) \\ &\le Q_i(k)^2 + \mu_i(k)^2 + A_i(k)^2 + 2Q_i(k)\big(A_i(k) - \mu_i(k)\big), \end{aligned} \tag{55}$$

where the last step uses $\mu_i(k)A_i(k) \ge 0$. Summing over agents, dividing by two, and applying the conditional second-moment bound yields

$$\begin{aligned} &\mathbb{E}[\,L(Q(k+1)) - L(Q(k))\,|\,\mathcal{F}_k] \\ &\quad \le B + \mathbb{E}\left[\sum_{i=1}^M Q_i(k)A_i(k) \,\middle|\, \mathcal{F}_k\right] - \sum_{i=1}^M Q_i(k)\mathbb{E}[\mu_i(k) \mid \mathcal{F}_k]. \end{aligned} \tag{56}$$

Comparison with a reference assignment. The drift cost satisfies

$$\sum_{i=1}^M Q_i(k)A_i(k) = \langle C^{\mathrm{drift}}(k), Y^{(k)}\rangle. \tag{57}$$

Using Theorem 1, the uniform error bound, and the feasible reference assignment $\widetilde{Y}(k)$, we obtain

$$\mathbb{E}\left[\sum_i Q_i(k)A_i(k) \mid \mathcal{F}_k\right] \le \min_{Z\in\mathcal{A}_k} \sum_i Q_i(k)A_i^Z(k) + \xi \tag{58}$$

$$\le \sum_i Q_i(k)A_i^{\widetilde{Y}}(k) + \xi$$

$$\le \sum_i Q_i(k)\mathbb{E}[\mu_i(k) \mid \mathcal{F}_k] - \eta \sum_i Q_i(k) + \xi. \tag{59}$$

The last inequality is the slack condition (53). Substitution into (56) cancels the service terms and yields

$$\mathbb{E}[L(Q(k+1)) - L(Q(k)) \mid \mathcal{F}_k] \le B + \xi - \eta \sum_i Q_i(k). \tag{60}$$

Time-average bound. Taking expectations and summing (60) over $k = 0, \dots, N-1$ gives

$$\mathbb{E}[L(Q(N))] - \mathbb{E}[L(Q(0))] \le N(B+\xi) - \eta \sum_{k=0}^{N-1} \mathbb{E}\left[\sum_{i=1}^{M} Q_i(k)\right]. \tag{61}$$

Since $L(Q(N)) \ge 0$, rearrangement gives the following finite-horizon bound on average workload:

$$\frac{1}{N}\sum_{k=0}^{N-1} \mathbb{E}\left[\sum_{i=1}^{M} Q_i(k)\right] \le \frac{B+\xi}{\eta} + \frac{\mathbb{E}[L(Q(0))]}{\eta N}. \tag{62}$$

The initial moment condition (51) makes the final term vanish as $N \to \infty$. Taking the limit superior gives the claimed bound (14):

$$\limsup_{N\to\infty} \frac{1}{N}\sum_{k=0}^{N-1} \mathbb{E}\left[\sum_{i=1}^{M} Q_i(k)\right] \le \frac{B+\xi}{\eta}. \tag{63}$$

□

C TEST-TIME ADAPTATION ALGORITHMS

Hierarchical task clustering HTC uses the pretrained M_0-agent policy as a recursive partition operator, as summarized in Algorithm 2. An initial split partitions the current request set, after which the cluster with the largest estimated workload is split repeatedly. A budget B limits neural evaluations, and angular bisection supplies any remaining partitions. Bipartite matching then assigns these partitions to physical agents.

Algorithm 2 Hierarchical task clustering for deployment with a different agent count.

Require: π_{M_0}, $\mathcal{B}_k$, target agent count M, budget B
$\mathcal{C} \leftarrow \mathrm{Split}_{M_0}(\mathcal{B}_k)$ ▷ Initial split
while $|\mathcal{C}| - 1 + M_0 \le M$ and budget B remains **do**
  $C_{\text{heavy}} \leftarrow \arg\max_{C\in\mathcal{C}} \mathrm{WorkloadProxy}(C)$
  $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C_{\text{heavy}}\}) \cup \mathrm{Split}_{M_0}(C_{\text{heavy}})$ ▷ Split
end while
if $|\mathcal{C}| < M$ **then**
  $\mathcal{C} \leftarrow \mathrm{AngularBisection}(\mathcal{C}, M)$ ▷ Fallback
end if
$\sigma \leftarrow \mathrm{BipartiteMatching}(\mathcal{C}, \mathrm{PhysicalAgents})$
$Y_{ij} \leftarrow \mathbb{I}[r_j \in \mathcal{C}_{\sigma(i)}]$
return Y

D ADDITIONAL EXPERIMENTAL RESULTS

This appendix supplements Section 6 with demand comparisons, NYC replay results, decoder ablations, workload diagnostics, and latency component measurements. Appendix E.6 specifies the data and evaluation protocols.

Table 2: NYC results on 30 matched replays per agent count from the chronologically held-out test partition. Makespan and AvgWait are means±sample standard deviations across episodes. Time is mean dispatch latency in milliseconds.

Setting	Method	Makespan ↓	AvgWait ↓	Time ↓
$M = 5$	Greedy	626.31±34.92	170.10±10.60	**1.1**
	SA	567.15±29.20	134.58±7.69	615.8
	ACO	**538.11±38.86**	131.32±19.77	554.3
	OR-Tools	679.99±63.47	163.60±27.12	201.1
	iMTSP	1090.67±95.51	373.47±53.47	2.5
	SRL	686.15±37.14	230.53±24.83	29.5
	L-MCF	555.49±27.31	132.92±8.93	2.6
	Ours	540.81±23.18	**115.41±8.82**	16.3
$M = 10$	Greedy	598.67±23.91	141.21±8.39	**2.3**
	SA	520.55±34.56	96.74±5.17	629.3
	ACO	603.67±65.91	138.14±25.14	625.2
	OR-Tools	1048.93±80.30	310.50±35.72	1001.7
	iMTSP	600.21±38.26	110.90±10.43	3.0
	SRL	693.09±32.82	230.49±31.01	45.3
	L-MCF	526.83±34.05	95.95±10.63	5.0
	Ours	**512.67±28.27**	**85.08±6.42**	23.4
$M = 20$	Greedy	560.18±20.90	111.44±6.62	7.9
	SA	509.69±21.31	67.49±2.74	777.7
	ACO	780.07±138.03	187.14±45.04	1018.1
	OR-Tools	1644.33±92.37	511.78±33.86	2001.9
	iMTSP	561.10±44.06	80.75±5.49	**4.4**
	SRL	771.68±55.29	279.24±26.15	163.8
	L-MCF	549.74±37.92	70.37±6.53	6.8
	Ours	**496.63±17.60**	**65.91±3.35**	42.9

Table 3: Request-marginal designs at two settings $(M, |\mathcal{J}|)$. Uniform, waiting-weighted, and distance-weighted marginals are compared using makespan and AvgWait.

Setting	a design	Makespan ↓	AvgWait ↓
$(10, 900)$	Uniform	**742.55**	**183.40**
$(10, 900)$	Waiting weighted	743.11	183.61
$(10, 900)$	Distance weighted	**742.55**	183.52
$(20, 1800)$	Uniform	**643.34**	**143.18**
$(20, 1800)$	Waiting weighted	652.69	144.15
$(20, 1800)$	Distance weighted	647.62	144.07

D.1 Deployment results and additional comparisons

Effects of AP. AvgWait increased with AP for every evaluated method at each agent count. OT-Dispatch Net achieved the lowest mean AvgWait at all three demand levels. Its margin over the strongest baseline varied with M and AP, showing that the magnitude of the advantage depends on the operating setting.

Latency. Table 1 reports dispatch latency, while Table 6 reports insertion and replanning times separately.

NYC trace replay. Table 2 reports 30 matched, chronologically held-out replays for each $M \in \{5, 10, 20\}$. Episode construction preserves pickup order, relative arrival intervals, and spatial heterogeneity within the common simulation horizon. OT-Dispatch Net achieved the lowest mean AvgWait in all three settings and the lowest makespan for $M = 10$ and $M = 20$. ACO achieved a lower makespan for $M = 5$. All learned checkpoints were frozen before evaluation, so these results assess zero-shot transfer to NYC traces without adaptation.

Table 4: Decoder comparison at $(M, |\mathcal{J}|) = (5, 600)$,All evaluated assignments were feasible; ST denotes the straight-through estimator.

Decoder	Makespan ↓	AvgWait ↓	Dispatch (ms) ↓
Soft-quota decoder (ours)	**819.52**	**256.84**	4.47
Hungarian projection	820.55	257.33	18.47
Gumbel-Sinkhorn (ST)	1218.99	380.95	5.63
Independent sampling	1178.98	375.76	**4.18**

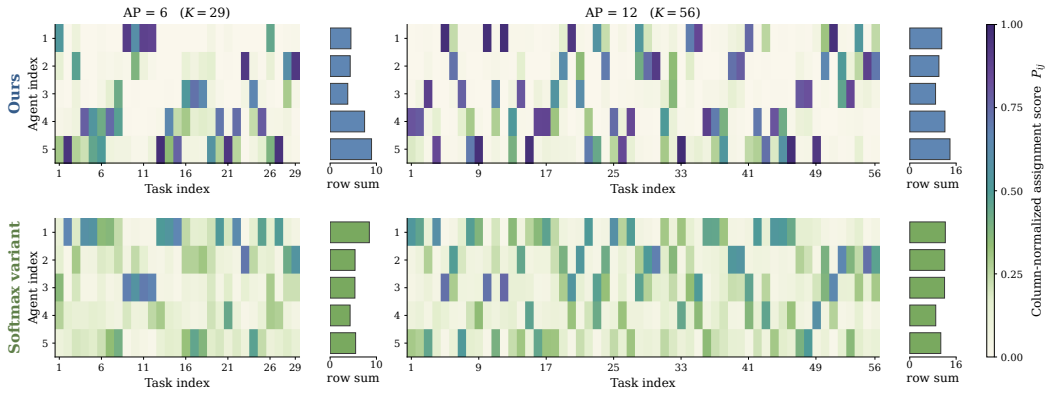

Figure 3: Base assignment probabilities P_{ij} for OT-Dispatch Net (top) and Softmax (bottom) at the same epoch, corresponding to Figure 2. Both settings use $M = 5$, and the bars on the right show row sums. OT concentrates most columns on one or two agents, whereas Softmax distributes mass more diffusely.

D.2 Additional ablations and transport plans

The ablations distinguish architectural changes within the stochastic policy from changes to discrete execution. OT+uniform b and Softmax retain autoregressive decoding and the PPO schedule. No OT, L-MCF, and deterministic decoder alternatives instead modify the execution rule.

Effects of OT coupling and the agent marginal. Figure 2 shows that No OT and Softmax increased mean AvgWait and makespan, whereas a uniform agent marginal gave intermediate results in all three settings.

Effects of request-marginal design. Table 3 compares uniform, waiting-weighted, and distance-weighted request marginals. Uniform weighting yielded the lowest AvgWait in both settings, although the numerical differences between designs were small.

Effects of decoder choice. Table 4 compares soft-quota decoding with Hungarian projection (Kuhn, 1955), Gumbel-Sinkhorn (Mena et al., 2018), and independent sampling. Soft-quota decoding and Hungarian projection achieved similar quality, with lower latency for soft-quota decoding (4.5 versus 18.5 ms). Gumbel-Sinkhorn and independent sampling had comparable dispatch latency (5.6 and 4.2 ms) but increased both makespan and AvgWait.

Transport structure. The base assignment probabilities in Figure 3 illustrate the effect of transport coupling. OT concentrates each request's probability on fewer agents and produces agent-dependent row sums, whereas Softmax distributes probability more diffusely.

D.3 Diagnostics related to the stability analysis

We examine auxiliary workload and the one-step drift-cost gap at $(M, \mathrm{AP}) = (5, 12)$ and $(10, 6)$. Gap summaries include only soft-quota epochs with $K_k \geq 3$, because per-request values are unstable

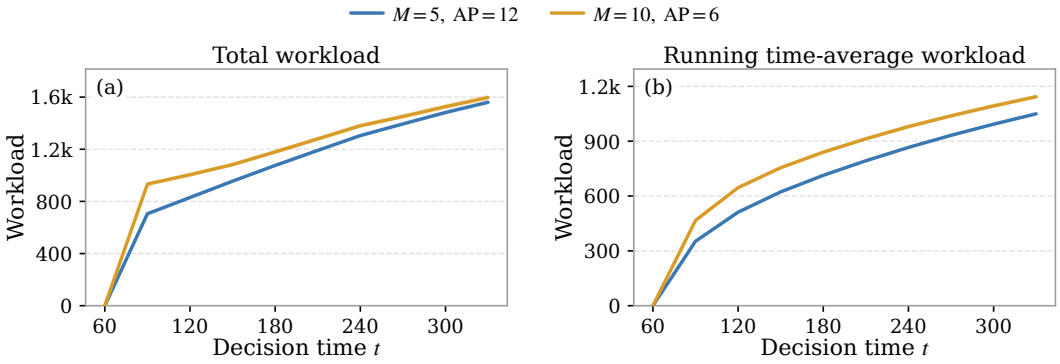

Figure 4: Auxiliary workload $\sum_i Q_i(k)$ (left) and its running average $\frac{1}{k+1}\sum_{l=0}^{k}\sum_i Q_i(l)$ (right). The running-average slope decreases in both settings over the evaluated finite horizon.

Table 5: One-step drift-cost gaps across the included epochs. P90 denotes the 90th percentile; the final column reports the median gap per request.

Setting	Included epochs	Median g_{drift}	P90 g_{drift}	Median g_{drift}/request
Full $(M = 5, \text{AP} = 12)$	300	7,737.3	18,073	130.55
Full $(M = 10, \text{AP} = 6)$	300	19,327	38,774	325.16

for very small batches. Batches with $0 < K_k < 3$ use the same policy but are excluded from these summaries. The analyzed gap is

$$g_{\text{drift}}(k) := \mathbb{E}\Big[\langle C^{\text{drift}}(k), Y^{(k)}\rangle \mid \mathcal{F}_k\Big] - \min_{Z\in\mathcal{Y}_{K_k}} \langle C^{\text{drift}}(k), Z\rangle, \tag{64}$$

This quantity subtracts the comparator cost from the conditional expected cost in Theorem 1. Dividing by K_k gives the corresponding per-request gap.

Table 5 reports median gaps of 7,737.3 and 19,327 and P90 values of 18,073 and 38,774 for $(M, \text{AP}) = (5, 12)$ and $(10, 6)$, respectively. These statistics summarize the distribution of drift-cost gaps over the evaluated finite horizon.

D.4 Latency components per epoch

Table 6: Dispatch, insertion, and replanning latency per epoch in milliseconds at three representative synthetic settings.

Setting $(M, \lvert\mathcal{J}\rvert)$	Dispatch $(R = 1)$	Dispatch $(R = 10)$	Insert	Replan
$(5, 600)$	4.09	114.64	176.29	1003.11
$(10, 900)$	5.43	123.25	116.47	2005.29
$(20, 1800)$	6.91	201.00	190.59	4010.06

Table 6 separates dispatch latency from insertion and replanning. Dispatch values agree with Table 1 after rounding, while replanning is the largest measured component in all three settings.

E Implementation and training details

Figure 5 details the policy architecture summarized in Figure 1. The following sections specify message passing, PPO training, decoder ordering, ablation variants, and evaluation protocols.

E.1 Encoder message passing

At epoch k, agent node v_i^{agent} carries the feature vector

$$x_i^{\text{agent}} = \big(p_i(k),\, n_i(k),\, \hat{C}_i(k),\, \tau_i^{\text{rem}}(k),\, \Delta_i^{\text{depot}}\big) \in \mathbb{R}^{d_a}, \tag{65}$$

where $\tau_i^{\text{rem}}(k)$ is the remaining service time and Δ_i^{depot} contains statistics relative to the depot. Request node v_j^{req} carries the corresponding request feature vector

$$x_j^{\text{req}} = \big(\ell_j,\, \alpha_j,\, w_j(k),\, \|\ell_j - \ell_{\text{depot}}\|\big) \in \mathbb{R}^{d_r}. \tag{66}$$

Pending-request and remaining-route features summarize previously assigned requests, which are not reintroduced as request nodes. Separate maps ϕ_a and ϕ_r embed the two node types in a common d-dimensional space:

$$h_i^{\text{agent},(0)} = \phi_a\big(x_i^{\text{agent}}\big), \qquad h_j^{\text{req},(0)} = \phi_r\big(x_j^{\text{req}}\big). \tag{67}$$

The graph $\mathcal{E}_k^G$ connects each agent to its $\min(\kappa_a, K_k)$ nearest requests bidirectionally and each request to its $\min(\kappa_r, K_k - 1)$ nearest peers. At layer ℓ, agent nodes aggregate request messages through

$$m_{i\leftarrow j}^{(\ell)} = \text{ReLU}\Big(h_j^{\text{req},(\ell-1)} + W_{e,ar}^{(\ell)}\, e_{ij}\Big), \tag{68}$$

$$h_i^{\text{agent},(\ell)} = \text{MLP}_a^{(\ell)}\Big(\big(1+\epsilon_a^{(\ell)}\big)h_i^{\text{agent},(\ell-1)} + \sum_{j\in\mathcal{N}_i^{ar}} m_{i\leftarrow j}^{(\ell)}\Big), \tag{69}$$

Distance ties are resolved lexicographically by location, release time, and immutable request identifier. Here, e_{ij} is the edge length and $W_{e,ar}^{(\ell)}$ projects it into the embedding space. The learnable coefficient $\epsilon_a^{(\ell)}$ weights the node's own embedding. Request nodes aggregate messages from both neighboring agents and neighboring requests:

$$\begin{aligned} h_j^{\text{req},(\ell)} = \text{MLP}_r^{(\ell)}\Big(\big(1+\epsilon_r^{(\ell)}\big)h_j^{\text{req},(\ell-1)} &+ \sum_{i\in\mathcal{N}_j^{ra}} \tilde{m}_{j\leftarrow i}^{(\ell)} \\ &+ \sum_{j'\in\mathcal{N}_j^{rr}} \bar{m}_{j\leftarrow j'}^{(\ell)}\Big). \end{aligned} \tag{70}$$

Each relation uses the same ReLU message form with a separate edge projection. Within each node type, all nodes share the MLPs and relevant projections. These shared maps and sum aggregation preserve permutation equivariance under node reordering, with no identity embeddings. Reordering nodes therefore reorders their output embeddings in the same way.

Graph construction evaluates $O(MK_k + K_k^2)$ pairwise distances. For fixed hidden dimension, each GNN layer processes $O(|\mathcal{E}_k^G|)$ relations, and the pair-cost head evaluates $O(MK_k)$ pairs. Sinkhorn scaling requires $O(N_{\text{SK}}MK_k)$ operations, while autoregressive decoding requires $O(MK_k)$ operations.

E.2 PPO training objective and configuration

The actor is trained by minimizing the clipped PPO surrogate loss

$$\mathcal{L}_{\text{PPO}}(\theta) = -\mathbb{E}_k\Big[\min\big(\rho_k(\theta)\hat{A}_k,\ \text{clip}(\rho_k(\theta), 1-\epsilon,\, 1+\epsilon)\hat{A}_k\big)\Big], \tag{71}$$

where $\rho_k(\theta) = \pi_\theta(Y_k \mid o_k)/\pi_{\theta_{\text{old}}}(Y_k \mid o_k)$ is the joint-action likelihood ratio and $\hat{A}_k$ is the advantage estimate. For a completed rollout $s_0, \ldots, s_L$ with $o_k = h(s_k)$, generalized advantage estimation (Schulman et al., 2016) gives

$$\hat{A}_k = \sum_{l=0}^{L-k-1} (\gamma\lambda_{\text{GAE}})^l \delta_{k+l}, \qquad \delta_k = r_k + \gamma V_\phi(o_{k+1}) - V_\phi(o_k), \tag{72}$$

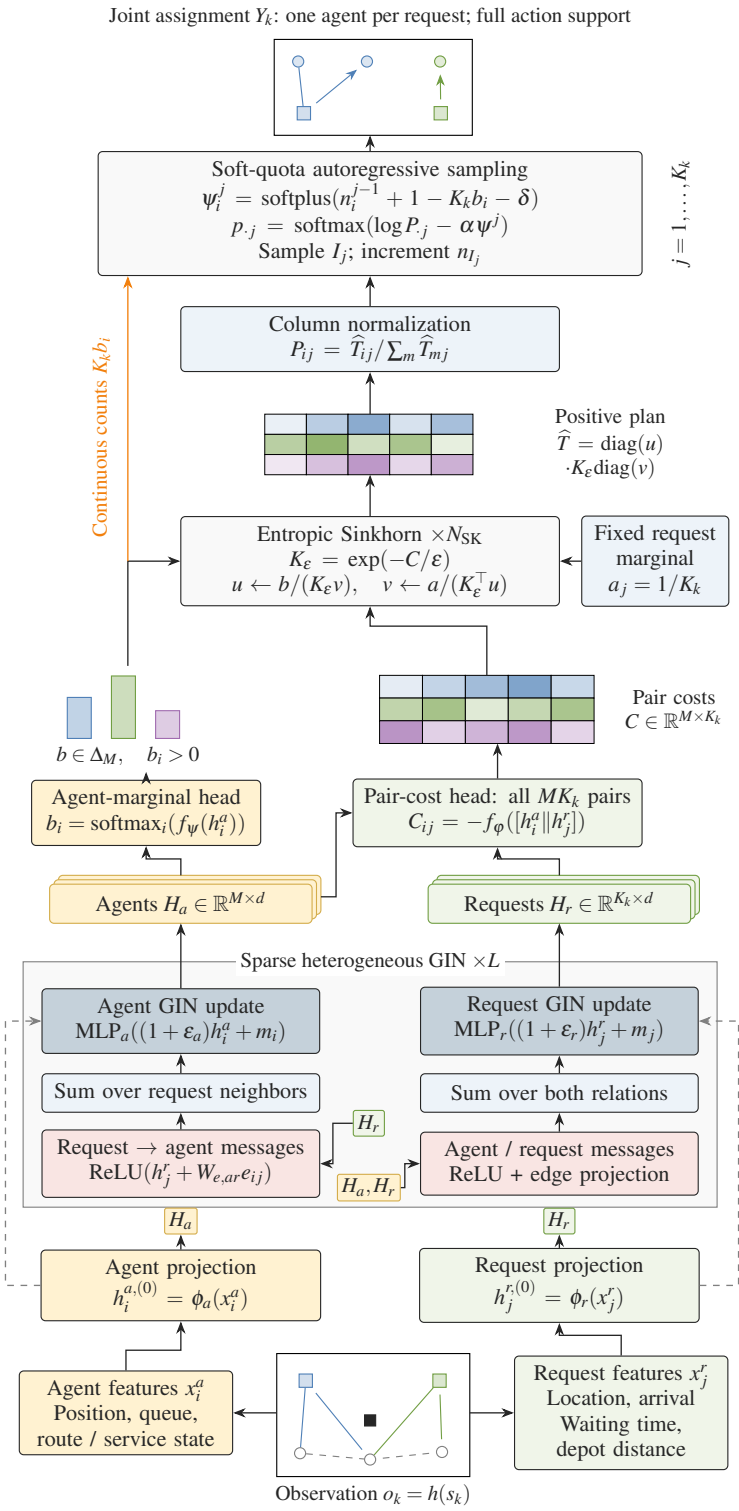

Figure 5: Detailed policy architecture of OT-Dispatch Net. The sparse heterogeneous encoder produces agent and request embeddings for the pair-cost and agent-marginal heads. These heads parameterize entropic transport with a fixed request marginal. Base assignment probabilities and continuous target counts $K_k b$ guide stochastic soft-quota decoding, which assigns one agent per request with full action support.

where the terminal value is $V_\phi(o_L) = 0$. The critic has the actor encoder's GNN architecture, with separate parameters and its own output head. A 20-epoch critic warmup precedes actor updates to reduce noise in the initial advantage estimates.

Transitions without dispatchable requests, including those after the final arrival, are advanced internally, with their potential change attached to the preceding decision transition. Table 7 lists the training configuration. Returns and advantages use the PPO discount $\gamma = 0.99$. Without discounting, cumulative reward equals the initial potential minus the terminal objective in (7); discounting introduces temporal weighting and breaks this exact telescoping identity.

Soft-quota action replay. The rollout buffer stores the sampled assignment Y_k and its joint log probability $\log \pi_{\theta_{\text{old}}}(Y_k \mid o_k)$. During replay, the current actor recomputes the marginal $b_\theta(o_k)$, the finite-iteration transport plan, and the conditionals in Algorithm 1. The stored assignment fixes each history count n_i^{j-1}. Summing the selected conditional log probabilities gives the current joint log probability $\log \pi_\theta(Y_k \mid o_k)$.

Proposition 3 ensures that old and current stochastic policies share support $\mathcal{Y}_{K_k}$, so $\rho_k(\theta)$ compares the same joint action and propagates gradients through transport and the soft penalty to the agent marginal. Rollout and replay use the same canonical request order, and the entropy bonus is averaged over the K_k requests to control its dependence on batch size. These likelihood properties concern the stochastic training policy; greedy autoregressive decoding remains a separate test-time rule.

Table 7: Training configuration shared by models trained with $M_0 \in \{5, 10\}$. Deployment at $M = 20$ uses the $M_0 = 5$ checkpoint through HTC.

Component	Setting
Encoder layers	3
Hidden dimension	128
Entropic regularization ε	0.05
Sinkhorn iterations N_{SK}	30
Algorithm	PPO with generalized advantage estimation
PPO epochs	200
Batch size	32
Gradient steps for each update	10
PPO discount γ	0.99
GAE λ	0.95
Clip range ϵ_{clip}	0.2
Actor learning rate	1×10^{-5}
Critic learning rate	3×10^{-4}
Value function coefficient	0.5
Entropy coefficient	0.01
Max gradient norm	0.5
Optimizer	Adam
Soft-quota penalty scale α	10
Soft-quota offset δ	0.5
Reward	decrease in the state-based potential in (6)
Rollout span	arrival horizon plus service completion
Terminal condition	all requests served and all agents back at the depot
Critic	GNN with the actor architecture and separate parameters
Critic warmup epochs	20
Candidate count during training R_{train}	1
Hardware	RTX 4090 GPU and i9-13900K CPU

E.3 Decoder ordering and permutation equivariance

The encoder is equivariant to agent and request reorderings (Section 4.2). The decoder instead follows the environment's fixed first-in, first-out (FIFO) order, with immutable request identifiers breaking arrival-time ties (Algorithm 1):

- **Arrival order.** Requests are sorted by $(\text{arrival time}, \text{request identifier})$. Each identifier is assigned on arrival and remains fixed throughout the episode.
- **Policy inputs.** Identifiers determine the sampling sequence but are not embedded as policy features. The encoder and both heads use the observation features specified in Section 4.2.
- **Joint probability.** The factorization $\pi_\theta(Y_k \mid o_k) = \prod_j p_{I_j,j}$ follows this visiting order. Each conditional depends on the prefix counts through the soft-quota penalty, so changing order generally changes the distribution.

Every ordering retains feasibility and full support for the stochastic policy. Each step assigns one agent with positive probability, and the prefix counts satisfy $\sum_i n_i^{j-1} = j - 1$. Thus all orderings share support $\mathcal{Y}_{K_k}$, although they need not assign the same probabilities to its elements. Using the canonical order in both rollout and replay makes the PPO likelihood ratio consistent.

Table 8: Decoder-order sensitivity for one $M = 5$ checkpoint over 30 matched instances (mean±sample SD). All variants use greedy autoregressive decoding and the shared routing backend. Canonical order is FIFO; permutation averaging evaluates 10 orders.

AP	Order	Makespan ↓	AvgWait ↓	Latency (ms) ↓
6	Canonical	726.42±30.96	203.97±12.57	4.98±0.09
6	Random	727.97±35.24	204.06±15.64	4.98±0.09
6	Perm. avg.	**694.15±32.31**	**191.40±10.34**	107.28±6.08
12	Canonical	921.80±30.69	301.41±18.47	6.18±0.12
12	Random	921.87±34.36	301.96±15.54	6.16±0.12
12	Perm. avg.	**896.29±36.22**	**292.90±16.58**	196.63±10.02

Table 8 compares deterministic greedy decoding under canonical and randomized request orders. At $\text{AP} = 6$ and 12, one random permutation changed mean AvgWait by only 0.04% and 0.18% and makespan by 0.21% and 0.01%, respectively. These small differences indicate limited sensitivity for the evaluated checkpoint over matched instances.

Permutation averaging applies greedy autoregressive decoding to ten request orders and executes the candidate with the highest probability under their uniform mixture. At $\text{AP} = 6$ and 12, it reduced AvgWait by 6.16% and 2.82% and makespan by 4.44% and 2.77%, respectively. These gains required substantially higher latency, leaving canonical decoding as the efficient default. Both decoding rules produce feasible joint assignments.

E.4 Ablation variant training details

The ablation study includes two stochastic architectural variants and one deterministic execution baseline. OT+uniform b and Softmax share the full model's encoder, initialization, and PPO schedule (Table 7), with the following modifications:

- **No OT:** removes Sinkhorn and greedily assigns each request to its nearest agent. This baseline changes both transport coupling and the execution rule.
- **OT+uniform b:** fixes $b_i = 1/M$ for every agent, replacing the learned agent marginal while retaining Sinkhorn and soft-quota decoding.
- **Softmax:** replaces Sinkhorn with columnwise probabilities $P_{\cdot j} = \text{softmax}(g_{\cdot j}/\tau)$, equivalently $\text{softmax}(-C_{\cdot j}/\tau)$, where $\tau > 0$. It retains the soft-quota penalty but removes the transport marginal constraints.

OT-Dispatch Net, OT+uniform b, and Softmax are trained through log probabilities of executable assignments. In OT-Dispatch Net and Softmax, the soft-quota penalty provides a gradient path to the

agent-marginal head; OT+uniform b fixes this marginal. No OT and L-MCF execute deterministically through nearest-agent assignment and minimum-cost flow over learned pair costs and marginals, respectively. All variants use matched instances and the shared routing backend.

E.5 Discussion of theoretical assumptions

Theorem 1 combines a cost decomposition with explicit marginal and decoder bounds; Theorem 2 requires additional assumptions for auxiliary workload stability.

Execution gap (Theorem 1). Equation (11) separates entropy regularization, the learned agent marginal, finite Sinkhorn scaling, soft decoding, and calibration between learned and drift costs. The fixed request marginal reduces the entropy contribution to $K_k \varepsilon \log M$. Equation (46) replaces the marginal and decoder residuals by explicit bounds; the remaining terms account for finite Sinkhorn error and cost calibration.

Residuals and comparison class. The comparator ranges over $\mathcal{Y}_{K_k}$, the full support of the stochastic decoder. Under learned costs, Δ_{marg} measures the cost of imposing the agent marginal. Proposition 6 bounds this cost by the distance to a marginal optimal for the current learned linear cost. Proposition 7 controls Δ_{soft} through cost ranges, target shares, and penalty parameters, including a normalization term when needed. Uniform calibration over $\mathcal{Y}_{K_k}$ permits substitution between learned and drift costs on both sides of the comparison.

Stability (Theorem 2). The result assumes bounded second moments, uniformly bounded total error, and a feasible reference assignment with uniform negative drift. The comparison class $\mathcal{Y}_{K_k}$ is independent of the learned agent marginal. Under these conditions, the auxiliary queues $Q_i(k)$ satisfy the time-average workload bound $(B + \xi)/\eta$ in (14).

E.6 Data generation and evaluation protocol

Synthetic benchmark. All agents begin at the central depot located at $(50, 50)$, and request locations are sampled uniformly from the square $[0, 100]^2$. Arrivals follow a homogeneous Poisson process with rate $\lambda_r = |\mathcal{J}|/T$, where $|\mathcal{J}|$ is the nominal request count and the realized count fluctuates around it. The arrival horizon is $T = 300$, and the decision interval $\Delta t = 30$ yields $N_{\mathrm{arr}} = 10$ arrival intervals. After arrivals cease, the simulation continues until all requests have been served, all routes have been completed, and every agent has returned to the depot, subject to a safety cutoff of $10T$. Agents travel at one distance unit per time unit. Demand is indexed by $\mathrm{AP} := |\mathcal{J}|/(MN_{\mathrm{arr}})$, the nominal number of requests per agent per arrival interval.

NYC trace benchmark. We use November 2025 yellow taxi records and Taxi Zone files from the NYC Taxi and Limousine Commission (TLC).[1] Pickup coordinates are Taxi Zone centroids within the Manhattan bounding box $[-74.03, -73.90] \times [40.70, 40.88]$. We jointly map both coordinates to $[0, 100]^2$, fix the depot at $(50, 50)$, and use Euclidean distance. We partition calendar dates before constructing episodes: November 1-20 for training, November 21-25 for validation, and November 26-30 for testing. Replays contain 600, 900, or 1800 pickups for $(M, \mathrm{AP}) = (5, 12)$, $(10, 9)$, or $(20, 9)$, respectively. Within each partition, timestamp-sorted pickups form non-overlapping consecutive blocks whose endpoints are mapped to 0 and $T = 300$, preserving relative arrival intervals and pickup locations. For each M, the dataset contains 120 training, 30 validation, and 30 test replays. Source-row identifiers verify that records are neither reused within nor shared across partitions for that setting. All methods evaluate the same 30 test replays in Table 2. Learned checkpoints are frozen beforehand, and no NYC partition is used for adaptation or checkpoint selection.

Evaluation. Synthetic evaluations use 30 shared Poisson realizations with seeds $42 + 100r$, $r = 0, \ldots, 29$; NYC evaluations use 30 shared, chronologically held-out replays per agent count. Every episode starts from its fixed central depot. Table 1 reports baseline means over matched instances and OT-Dispatch Net means and sample SDs over five training seeds; appendix captions flag instance-level variation. Table 2 reports replay-level sample SDs for one checkpoint per learned method.

[1] https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page