LEARNING TO CONSOLIDATE: CANDIDATE-POOL OPTIMIZATION FOR FREIGHT NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We study freight planning in which shipments must follow a single path but can share vehicles on scheduled departures. The large number of feasible paths makes joint optimization difficult, while choosing paths independently can miss opportunities to consolidate freight. We propose WAVECAST, a learning-guided method that builds and expands a pool of candidate paths. A demand-conditioned encoder scores departures, and the scores are mapped to service–mode search charges. Under each resulting charge vector, a path subproblem is solved exactly for every shipment. A restricted mixed-integer master then selects paths and vehicle counts jointly using the original costs and constraints. Loads in the current incumbent guide additional path generation before the master is solved again. On held-out public instances, learned fields achieve lower costs and observed median online times than matched analytic fields. On a 1,602-shipment development instance, the reported run achieves a 3.118% gap to a separate global lower bound in 807.08 seconds from prepared features and frozen models.

1 INTRODUCTION

We study freight planning over a network of scheduled departures. Each shipment has an origin, a destination, a release time, a delivery commitment and a weight, and must be assigned to one spatially and temporally feasible path. Shipments using the same departure can share vehicle capacity. The planner therefore chooses one path for each shipment together with the integer number of vehicles assigned to each service. The objective is to minimize vehicle and shipment-dependent costs while satisfying delivery requirements, service capacities, vehicle limits and, where applicable, terminal handling limits.

The main difficulty is that shipment paths cannot be chosen independently. Two shipments may each have an inexpensive and timely path, yet those paths may require separate vehicles. Assigning the shipments to a common departure can lower total cost even when one follows a longer or more expensive route. This consolidation effect couples path choices through integer vehicle costs and shared capacities. At the same time, each shipment may have many spatial routes and departure-time realizations, making the complete path formulation too large to enumerate directly.

Restricting each shipment to a compact candidate pool reduces the joint problem but risks omitting paths that become useful only through consolidation. Directly predicting one complete route per shipment is also restrictive because a teacher route is only one realization of a shared service design. We therefore learn which departures are useful for the demand instance and use these predictions to guide the generation of multiple feasible paths.

We develop WAVECAST, a learning-guided candidate-pool algorithm for scheduled service network design. The name combines "wave," referring to scheduled departure batches, with "cast" from forecasting; predictions concern useful departures for the given demand, rather than future demand. A demand-conditioned encoder assigns a score to each timetable departure, shared across its available vehicle modes. These scores define nonnegative search charges under which the oracle generates one feasible path per shipment. An original-cost restricted master then coordinates the proposed paths and vehicle decisions. Its incumbent loads reveal where a shipment can reuse residual capacity and where rerouting a group may eliminate a weakly used service. The corresponding alternatives enlarge the pool, after which the master is reoptimized.

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.

Prior work uses learning for solver completion and column selection (Nair et al., 2020; Sun et al., 2023). We make two contributions:

- **Departure-field learning for path proposals.** We learn an instance-level preference over shared departures rather than one complete route per shipment. The scores are mapped to service–mode search charges, under which an exact shipment-level oracle generates feasible path proposals. We compare standard cellwise, set-pooled and latent-attention encoders within this common interface; the contribution is the learning–optimization interface, not a new neural architecture.
- **Feedback-based candidate-pool optimization.** We combine the learned proposals with an original-cost restricted master and incumbent-guided leave-one-out and service-closure neighborhoods. The master solution supplies the loads used to generate further paths, and the enlarged pool is then reoptimized under the original model. The procedure is a primal heuristic: validated incumbents provide upper bounds, but shipment-level exactness does not certify global optimality.

On 108 public test instances from 16 held-out source networks, the learned fields improve average cost by 2.54–2.84% over matched analytic fields and yield lower observed median online times under the same nominal budget. Attention does not outperform the simpler encoders, and complete warm-started MIP obtains lower final costs at the end of these runs. On the 1,602-shipment development instance, the reported run attains a validated cost of $364{,}865.72469$ and a 3.118% gap to a separate global lower bound. Its recorded online time is 807.08 seconds, starting from prepared features and frozen models. Section 5 separates the evidence sources, timing conventions and claim boundaries.

2 Related work

Service network design and path restriction. Service network design coordinates freight routes and capacity to exploit consolidation (Crainic, 2000). Time-expanded formulations expose this coordination but grow with temporal resolution. Boland et al. (2017) iteratively refine partially time-expanded networks for continuous-time service network design, while Van Dyk & Koenemann (2023) use arc-dependent time discretizations, including instances with restricted commodity routes. These methods refine the temporal representation of the optimization problem. WaveCast instead works on a prescribed timetable or exact integer-time network, restricts the shipment path families, and expands them using learned and incumbent-based proposals. For stochastic service network design, Baubaid (2026) learns recourse costs for a surrogate model; our master always evaluates candidates with the original deterministic costs.

Learning-guided combinatorial search. Neural Diving predicts partial integer assignments for solver completion (Nair et al., 2020); neural large-neighborhood search learns a routing repair operator (Hottung & Tierney, 2019); and heatmap-guided search separates learned preferences from combinatorial construction (Fu et al., 2021). Learning has also been used to filter or select columns and to accelerate pricing (Sun et al., 2023; Chi et al., 2022; Lu et al., 2024; Abouelrous et al., 2025; Hijazi et al., 2026). In contrast, WaveCast predicts departure preferences before complete paths are constructed, and its incumbent-based charges generate primal alternatives rather than LP certificates.

Demand-set representations. DeepSets (Zaheer et al., 2017), Set Transformer (Lee et al., 2019), and Perceiver (Jaegle et al., 2021) provide standard mechanisms for unordered inputs and efficient attention. We use cellwise, set-pooled and latent-attention encoders as alternative ways to condition departure scores on demand. The contribution is the interface between these representations, exact path generation and master reoptimization, not a new neural architecture.

3 Problem setting and formulation

We study a deterministic, finite-horizon service network design problem. Each shipment is unsplittable across paths: all of its weight follows one spatially and temporally feasible service sequence.

Capacity is pooled across the vehicles dispatched on the same service–mode option. The planner jointly chooses shipment paths and vehicle counts to minimize vehicle and shipment-dependent costs, subject to service and terminal capacities.

3.1 FEASIBLE SHIPMENT PATHS AND COSTS

Let $\mathcal{H}$ and $\mathcal{G}$ index terminals and planning days, and let $\mathcal{S}$ and $\mathcal{K}$ denote the sets of scheduled service–mode options and shipments. Terminal $h \in \mathcal{H}$ can process at most B_{hg} units of transferred weight on day $g \in \mathcal{G}$, at unit handling cost η_h. Time is discretized at one-minute resolution on a common axis: all service departure and arrival times and all shipment release and deadline times are integer minutes from the start of the planning horizon. For example, if the horizon starts at 00:00 on day 1, then $t = 510$ denotes 08:30 on day 1 and $t = 1{,}515$ denotes 01:15 on day 2. For terminal-capacity accounting, each scheduled arrival is also assigned to its planning day $g \in \mathcal{G}$.

Each service–mode option is represented by the tuple

$$s = \left(i_s, j_s, t_s^{\mathrm{dep}}, t_s^{\mathrm{arr}}, m_s, Q_s, F_s, \bar{y}_s\right) \in \mathcal{S}.$$

Here, $i_s, j_s \in \mathcal{H}$ are the origin and destination terminals, t_s^{dep} and t_s^{arr} are the scheduled departure and arrival times, and m_s is the transportation or ownership mode. In the reference application, $m_s \in \{\mathrm{own}, \mathrm{out}\}$ distinguishes owned and outsourced capacity. The remaining attributes are the capacity $Q_s > 0$ per vehicle, cost $F_s \geq 0$ per dispatched vehicle, and maximum vehicle count $\bar{y}_s \in \mathbb{Z}_+$. Thus, one timetable departure may induce an owned option and an outsourced option with the same $(i_s, j_s, t_s^{\mathrm{dep}}, t_s^{\mathrm{arr}})$ but different $(m_s, Q_s, F_s, \bar{y}_s)$; these are distinct elements of $\mathcal{S}$. When departures are available on an integer time grid rather than a fixed timetable, each feasible arc–time pair is treated as one scheduled option.

Each shipment is represented by the tuple

$$k = \left(i_k, j_k, w_k, r_k, d_k, \bar{d}_k, \pi_k\right) \in \mathcal{K}.$$

Here, $i_k, j_k \in \mathcal{H}$ are the origin and destination, $w_k > 0$ is the weight, r_k is the release time, d_k is the promised arrival time, $\bar{d}_k$ is the hard arrival limit, and π_k is the penalty per day of lateness. We assume $r_k \leq d_k \leq \bar{d}_k$ and remove orders with $i_k = j_k$ before optimization.

A path $p = (s_1, \ldots, s_{|p|})$ is feasible for shipment k when it is continuous in both space and time:

$$\begin{aligned} \text{spatial:} \quad & i_{s_1} = i_k, \quad j_{s_q} = i_{s_{q+1}}, \quad j_{s_{|p|}} = j_k, \\ \text{temporal:} \quad & t_{s_1}^{\mathrm{dep}} \geq r_k, \quad t_{s_q}^{\mathrm{arr}} \leq t_{s_{q+1}}^{\mathrm{dep}}, \quad t_{s_{|p|}}^{\mathrm{arr}} \leq \bar{d}_k. \end{aligned}$$

Here, $q = 1, \ldots, |p| - 1$ indexes consecutive services. The spatial conditions connect the shipment's origin and destination through adjacent services. The temporal conditions enforce release, feasible transfers, and the hard arrival limit; waiting at an intermediate terminal is allowed. Shipment-specific service permissions and other route restrictions are absorbed into the feasible path family $\mathcal{P}_k$. We assume that the complete joint problem has a feasible solution; a finite candidate pool need not contain one.

For path p, let $a_{ps} = 1$ if p uses service s, and zero otherwise, and let b_{phg} count its intermediate arrivals at terminal h on day g, excluding origin loading and final delivery. Models without terminal limits omit the corresponding constraints. In general, c_{kp} denotes all shipment-dependent costs of assigning shipment k to path p, including variable transportation, handling and lateness costs where applicable. This definition accommodates commodity-specific route costs without changing the master formulation. With $(z)_+ = \max\{z, 0\}$, the reference path cost is

$$c_{kp} = w_k \sum_{h \in \mathcal{H}} \sum_{g \in \mathcal{G}} \eta_h b_{phg} + \pi_k \left\lceil \frac{(t_{s_{|p|}}^{\mathrm{arr}} - d_k)_+}{1{,}440} \right\rceil.$$

The first term charges handling per unit weight for each intermediate arrival, and the second charges π_k for each commenced 1,440-minute interval after d_k. Vehicle costs are excluded from c_{kp} because they depend on aggregate service loads; the formulation below captures them through $\sum_s F_s y_s$. In the reference application, $\bar{d}_k = d_k + 2{,}880$, so a feasible path incurs a lateness penalty of at most $2\pi_k$.

3.2 PATH-BASED MIXED-INTEGER FORMULATION

Let $x_{kp} = 1$ if shipment k uses path $p \in \mathcal{P}_k$, and zero otherwise, and let y_s be the number of vehicles assigned to service–mode option s. The complete path formulation is

$$\min_{x,y} \quad \sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}_k} c_{kp} x_{kp} + \sum_{s\in\mathcal{S}} F_s y_s \tag{1}$$

$$\text{s.t.} \quad \sum_{p\in\mathcal{P}_k} x_{kp} = 1 \qquad \forall k \in \mathcal{K}, \tag{2}$$

$$\sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}_k} w_k a_{ps} x_{kp} \le Q_s y_s \qquad \forall s \in \mathcal{S}, \tag{3}$$

$$\sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}_k} w_k b_{phg} x_{kp} \le B_{hg} \qquad \forall h \in \mathcal{H},\ g \in \mathcal{G}, \tag{4}$$

$$x_{kp} \in \{0,1\} \qquad \forall k \in \mathcal{K},\ p \in \mathcal{P}_k, \tag{5}$$

$$y_s \in \{0,\dots,\bar{y}_s\} \qquad \forall s \in \mathcal{S}. \tag{6}$$

The objective combines shipment-level path costs with service-level vehicle costs. Constraints (2)–(4) respectively enforce one path per shipment, sufficient vehicle capacity on each service, and daily terminal handling capacity. Constraints (5)–(6) impose integrality and vehicle limits. Consolidation arises when several selected paths use the same service–mode option and share its dispatched vehicles.

Because $\mathcal{P}_k$ can be combinatorially large, explicit enumeration quickly becomes impractical. With 1,000 shipments, 100 spatial routes per shipment and 100 timetable realizations per route already yield 10^7 binary path variables. We therefore keep candidate sets $P_k \subseteq \mathcal{P}_k$ and substitute them for the complete path families. Any feasible restricted-master solution extends to the complete model by setting omitted variables to zero, so its cost is a valid upper bound. Restricted-master optimality is not global optimality, however, and its LP value is not generally a valid lower bound because feasible columns are omitted. Section 4 describes how the pools are generated and expanded; Appendix B.1 records an optional bounding construction over the complete path domain.

4 WAVECAST: LEARNING-GUIDED CANDIDATE-POOL OPTIMIZATION

WAVECAST alternates between proposing shipment paths and selecting them jointly. Offline, an encoder learns demand-conditioned departure scores (Section 4.1). Online, the scores are mapped to search-charge vectors that guide single-shipment path oracles, and a restricted master selects paths and vehicle counts using the original model (Section 4.2). Incumbent loads then guide further path generation and master reoptimization (Section 4.3). Search charges affect proposals only; they never replace the original master objective.

4.1 DEMAND-CONDITIONED DEPARTURE FIELDS

Input representation. Partition the shipments into demand contexts, $\mathcal{K} = \dot{\bigcup}_{q=1}^{J} \mathcal{K}_q$. A context may group shipments with common origin, destination and time attributes, or contain one commodity when its costs and route permissions must remain distinct. Its feature vector u_q describes these attributes and the size and weight distribution of the group; v_q is its aggregate weight in the encoder's units. Appendices A and E specify the two feature pipelines and their scaling.

A *departure cell* $\ell \in \mathcal{L}$ groups the service–mode options sharing $(i_s, j_s, t_s^{\mathrm{dep}}, t_s^{\mathrm{arr}})$; $\ell(s)$ is the cell of option s. Its fixed-length feature vector e_ℓ describes the timetable and network location, summarizes the available modes, and includes input-derived demand-pressure features such as reachable shipment mass and temporal flexibility. Thus, u_q describes a demand context, whereas e_ℓ describes one physical departure shared by its available modes.

Alternative encoders. The public benchmark and Reference-1602 use activation-supervised fields. Each of the following separately trained architectures outputs one real-valued activation logit z_ℓ per cell:

- **CellMLP** computes $z_\ell = f_{\text{cell}}(e_\ell)$. Demand enters through the demand-pressure features, without learned aggregation of the context set.
- **DeepSets** maps each context to $h_q = \phi(u_q)$ and summarizes the complete, unordered demand set as
$$\mathbf{g} = \rho\left(\left[\frac{1}{J}\sum_q h_q,\ \frac{\sum_q v_q h_q}{\sum_q v_q},\ \log(1+J),\ \log\left(1+\sum_q v_q\right)\right]\right),$$
then computes $z_\ell = f_{\text{set}}([e_\ell, \mathbf{g}])$. The unweighted mean represents a typical context, the weighted mean a context seen by a typical unit of demand, and the log terms retain the number of contexts and total demand scale. Every departure receives the same summary $\mathbf{g}$.
- **Latent attention** forms $h_q = \phi(u_q)$ and lets each departure query R latent demand representations:
$$\{b_i\}_{i=1}^R = A_{\text{latent}}(\{h_q\}_{q=1}^J), \qquad g_\ell^{\text{att}} = A_{\text{query}}(e_\ell, \{b_i\}_{i=1}^R), \qquad z_\ell = f_{\text{att}}(g_\ell^{\text{att}}).$$
A_{latent} compresses the demand contexts through learned latent queries, and A_{query} lets each departure retrieve a different summary from those latents.

Here, ϕ and ρ are multilayer perceptrons (MLPs), the f functions include scalar output heads, and $[\cdot,\cdot]$ denotes vector concatenation.

Offline supervision. For a training instance and validated feasible teacher D, let $r_\ell(D) = 1$ if D uses an option in cell ℓ whose mode belongs to the supervised set $\mathcal{M}_{\text{sup}}$, and zero otherwise. The reference uses $\mathcal{M}_{\text{sup}} = \{\text{own}\}$; the public benchmark has one mode. Teachers need not be optimal. The encoder is trained as an imbalanced binary classifier; Appendices A and E specify losses, features and checkpoint selection. Online inference uses frozen logits without differentiation through the oracle or master. The nine-case study instead uses an inverse-supervised field variant of the same candidate-pool framework (Appendix I.1).

Online charge construction. With sigmoid $\sigma(x) = (1+\exp(-x))^{-1}$, score center $\mu \in \mathbb{R}$, temperature $T > 0$ and shaping parameter $\gamma > 0$, define the smooth search-charge vector
$$\alpha_s = F_s\left[1 - \sigma\left(\frac{z_{\ell(s)} - \mu}{T}\right)\right]^\gamma, \qquad 0 \le \alpha_s \le F_s. \tag{7}$$
High-scoring departures receive lower search charges. Available modes share a cell score but use their own F_s, placing the charge in objective units. Sharing the reference own-mode score with outsourced options guides proposals; it does not predict outsourced activation. The logits need not be calibrated probabilities, and α is not an LP dual vector. Prespecified centers, temperatures or perturbations produce a finite charge collection $\mathcal{A} = \{\alpha^1, \ldots, \alpha^M\}$ from frozen neural or nonneural scores. The appendices specify the two implementations; the displayed upper bound applies before any multiplicative perturbation.

4.2 Initial candidate pool and original-cost master

Initial path proposals. For each shipment k and charge vector $\alpha \in \mathcal{A}$, an exact shipment-level path oracle solves
$$p_k(\alpha) \in \arg\min_{p \in \mathcal{P}_k}\left[c_{kp} + \sum_{s \in \mathcal{S}} a_{ps}\alpha_s\right]. \tag{8}$$
Each charge vector yields one minimizing path per shipment. The shortest-path reduction requires c_{kp} to be additive over event-graph arcs apart from a destination-arrival penalty, and all shipment-level restrictions to be represented by allowed arcs and, where needed, additional graph state. The oracle solves this problem on the complete shipment event graph without enumerating $\mathcal{P}_k$; the reference timetable gives a DAG, while the public implementation also handles zero-duration arcs. Shared service and terminal capacities remain in the restricted master. Under these conditions, exactness refers to Equation (8), not to the joint network-design problem.

Let V_k collect variants from option exclusion or feasible whole-trajectory shifts; Appendix A gives the rules. If a validated warm start is supplied, let W_k contain its path for shipment k; otherwise, $W_k = \varnothing$. The initial pool is
$$P_k^0 = \text{dedup}(\{p_k(\alpha^m) : m = 1, \ldots, M\} \cup V_k \cup W_k). \tag{9}$$

If no candidate has been stored for a shipment, a zero-charge call minimizes c_{kp}. Every stored path is checked against the original path rules; individual path coverage does not guarantee a jointly feasible assignment.

Original-cost joint selection. For any candidate pool $P = (P_k)_{k\in\mathcal{K}}$, let $\mathcal{F}(P)$ denote the feasible set defined by Constraints (2)–(6) after replacing $\mathcal{P}_k$ by P_k. The restricted master is the compact problem

$$\mathrm{RMP}(P): \quad \min_{(x,y)\in\mathcal{F}(P)} \left\{ \sum_{k\in\mathcal{K}} \sum_{p\in P_k} c_{kp} x_{kp} + \sum_{s\in\mathcal{S}} F_s y_s \right\}. \tag{10}$$

We solve $\mathrm{RMP}(P^0)$ within a time budget using the original costs and constraints, warm-starting from the supplied solution when available; all search charges are discarded. Let X^0 be the least-cost validated solution among the supplied warm start and any returned incumbents, retaining the warm start if the solve returns none. It provides an upper bound and the loads and users needed for expansion. Failure to obtain such an incumbent within budget does not establish infeasibility.

4.3 Incumbent-based pool expansion

Starting from (P^0, X^0), the algorithm repeats expansion and reoptimization subject to round and time limits. At round r, the service users and loads in incumbent X^r identify opportunities to reuse existing capacity or jointly leave a service and eliminate its vehicles.

Leave-one-out (LOO) expansion. Temporarily remove shipment k while holding the other paths fixed, and let W_s^{-k} be the residual load on service s. With vehicle counts reduced to those required by these loads, the additional vehicle cost of inserting k on s is

$$\alpha_{ks}^{\mathrm{LOO}} = F_s \left(\left\lceil \frac{W_s^{-k} + w_k}{Q_s} \right\rceil - \left\lceil \frac{W_s^{-k}}{Q_s} \right\rceil \right), \tag{11}$$

Options exceeding $\bar{y}_s$ receive a prohibitive charge. Re-solving Equation (8) with $\alpha_{ks}^{\mathrm{LOO}}$ proposes a path for k. This is the exact service-level increment for the fixed residual-load snapshot; the master handles terminal constraints and interactions among proposals. The appendices specify snapshot updates.

Service-level closure expansion. Moving one shipment may leave a vehicle in use, whereas rerouting several users can eliminate it. The decoder forbids or penalizes a used source service and proposes an alternative for each incumbent user. The master receives these paths together but may retain the source service; ranking and replacement rules are detailed in Appendices A and E.

Pool update and reoptimization. After deduplication, $P_k^{r+1} \supseteq P_k^r$. Warm-start $\mathrm{RMP}(P^{r+1})$ and retain the least-cost validated incumbent as X^{r+1}. Old paths preserve feasibility, so the reported best cost is nonincreasing. Stop at the round or time limit or when no path is added; no convergence to the unrestricted optimum is claimed.

Algorithm 1: One WaveCast search chain

Input: instance, frozen score sources, optional validated warm start, round limit, and time budgets.
Construct $\mathcal{A}$ and generate P^0, including the supplied warm-start paths. Solve $\mathrm{RMP}(P^0)$ and retain the least-cost validated solution from the warm start and the solve. If neither provides one, return no feasible solution.
For each round while time remains, generate LOO and closure paths from the retained incumbent, deduplicate them into $P^{r+1} \supseteq P^r$, and reoptimize; stop if no path is added.
Output: the least original-cost validated solution found within budget.

Independent chains may optionally use different charge collections; their final pools can be united and reoptimized once. The validator reconstructs paths, loads, vehicle counts and costs under the original rules. The original cost of any validated solution is an upper bound for the complete model; this primal pool heuristic supplies no global optimality certificate (Appendix C).

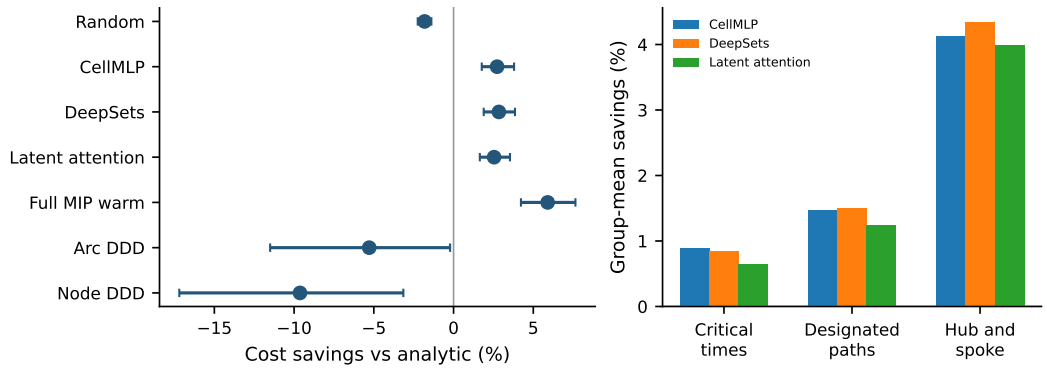


Figure 1: Native cost comparison. Left: paired group bootstrap intervals against the common analytic search. Right: group means by benchmark family; the families contain only 2/6/8 held-out networks, respectively. All three architectures and training seeds are retained.

5 Experimental evaluation

We evaluate whether learned fields improve matched nonlearned search, whether incumbent feedback improves solution quality, and how performance changes with demand scale. Held-out public tests provide the primary learning comparison; the real-data case is a development instance.

5.1 Evaluation protocol

The public SND-RR archive of Van Dyk & Koenemann (2023) contains 576 instances in 80 source-network groups. A metadata-only split assigns 48/16/16 groups and 348/120/108 instances to training/validation/test, keeping source variants together. All three encoders are trained on validated teachers with seeds 11/22/33; validation selects checkpoints, and test solutions supply no labels. The prespecified matrix contains 19 arms and 2,052 runs. Network groups are the units of statistical replication.

Each primary run receives 30 nominal online seconds and eight solver threads. Field methods share the decoder, enrichment rules, master and field count. Full MIP cold/warm starts and arc/node dynamic discretization discovery (DDD) provide optimization baselines. The clock includes input processing, model loading, inference, search and validation, but excludes interpreter startup and imports completed before benchmark entry; actual runtimes and overruns are retained.

Feasible costs are independently reconstructed. Savings are $100(U_{\text{control}} - U_{\text{method}})/U_{\text{control}}$. Seed costs are averaged within each instance before calculating its savings; instance savings are averaged within groups, which are weighted equally. Intervals use 20,000 paired group-bootstrap draws and are unadjusted 95% intervals conditional on frozen models and paired feasibility. Missing solutions receive no imputed cost. Gap is $100(U - L)/U$ using a valid global lower bound, not a restricted-master bound. Appendix J.1 gives the complete protocol; Appendix E specifies the benchmark and training.

5.2 Held-out native results

Table 1 reports the primary results. Learned fields improve group-mean cost by 2.54–2.84% over analytic search, while median online time falls from 23.6 to 7.5–8.9 seconds. Attention does not outperform the simpler encoders. Complete warm-started MIP is feasible on all 108 instances and obtains lower final costs: learned costs are 3.33–3.65% higher. Different actual runtimes under the nominal cap preclude an equal-time interpretation of this final-cost comparison.

With the same latent checkpoint, complete search improves on removing LOO or closure by 1.93% [1.48, 2.48] and 0.60% [0.23, 1.13], respectively. These are single-checkpoint diagnostics. In secondary follow-ups on the same test groups, CellMLP's additional saving over a thoroughly fitted 39-parameter affine scorer is only 0.20% $[-0.45, 0.91]$. A matched heterogeneous four-chain port-

Table 1: Native held-out test at 30 nominal seconds and eight solver threads. Positive savings means lower original cost than analytic search; intervals resample 16 source groups. Feasibility stacks all scheduled seed–instance runs (324 for three-seed methods, 108 otherwise). Costs are seed-averaged before group statistics. Gap uses the best valid global lower bound and averages only feasible runs; conditional comparisons must be read together with feasibility. Wall is the median measured online time.

Method	Feasible	Savings (%) [95% CI]	Gap (%)	Wall (s)
Analytic	108/108	0.00 [0.00, 0.00]	6.12	23.6
Random	324/324	-1.81 [-2.25, -1.39]	7.63	11.9
CellMLP	324/324	2.73 [1.77, 3.80]	3.72	7.5
DeepSets	324/324	2.84 [1.89, 3.85]	3.62	8.9
Latent attention	324/324	2.54 [1.65, 3.54]	3.90	8.1
Full MIP cold	108/108	5.91 [4.27, 7.67]	0.57	30.1
Full MIP warm	108/108	5.90 [4.23, 7.64]	0.58	30.1
Arc DDD	108/108	-5.29 [-11.50, -0.22]	8.95	30.0
Node DDD	102/108	-9.63 [-17.20, -3.15]	11.82	30.0
Latent without LOO	108/108	0.48 [-0.14, 1.20]	5.83	9.1
Latent without closure	108/108	1.81 [1.11, 2.64]	4.58	6.1

folio saves 1.85% [1.18, 2.55] against analytic chains. Figure 1 and Appendix J.2 give family results and further comparisons.

5.3 Derived 1,600-demand scale stress

The 16 held-out source networks also yield derived 1,600-demand cases, with no retraining and 120 nominal seconds per run; these are not independent new networks. Learned-field savings over analytic search range from 0.50% to 1.16% (Table 27). Learned costs remain 0.79–1.50% higher than warm MIP at 120 nominal seconds. In a separate fixed-time comparison, the three learned methods save 3.24–4.89% relative to warm MIP at 10 seconds, with all 16 cases feasible for every method; complete MIP improves more at later horizons. Appendix J.3 retains all stress outcomes, including two DDD exceptions, and Appendix H.7 reports the fixed-time comparisons.

5.4 Real-data development case

Reference-1602 is an anonymized development instance; its historical training scenarios perturb demand on the same network. The reported solution serves all 1,602 orders at cost $364{,}865.72469$, with a 3.118% gap to the separate historical global lower bound. Its recorded 807.08 seconds start from prepared features and frozen models. The direct full-model comparison uses different resources and time origins and does not establish a matched speedup. Appendix J.4 gives training and comparison details; Appendix B separates the reported run from the lower-cost six-chain union result.

A separate nine-case inverse-field study freezes one model trained on Medium scenarios 01–05 and selected on 06, then tests scenarios 07–09 at 800/1,600/3,200 orders on the same physical network. Runs use two threads and a 600-second accepted-incumbent deadline. The method has the lowest cost among the six reported main methods on all nine cases. Learned fields improve on analytic fields by 0.66–3.40%, but random fields are 0.40% better on Large/test_08. Small/Medium gaps use independently replayed numerical global bounds; Large has no verified global bound and reports best-observed relative deviation instead. These results support reuse across the observed scales on this network. Appendix J.5 provides the complete tables and ablations.

6 LIMITATIONS AND CONCLUSION

Three limits govern interpretation. The real-data case influenced development and is not evidence of out-of-distribution generalization. The public networks are held out but come from three related generators, and the scale stress reuses those network groups. Finally, the reference and public settings have different operational constraints and separately trained models.

The comparisons isolate departure-field choices within one search interface; they do not establish superiority over every learned column method, decomposition or local-search algorithm. Teacher generation requires offline computation, the master uses a commercial optimizer, and a time-limited restricted solve can fail to find or improve a feasible solution. WAVECAST is therefore a primal heuristic rather than an optimality-certified solver.

WAVECAST combines an optimization algorithm for coordinated path selection with learned recommendations over shared departures. Exact shipment-level decoding, an original-cost trajectory master and incumbent-driven pool expansion produce a validated 3.118% global-bound gap in 13.5 minutes on the 1,602-shipment development case. On held-out public networks, learning the recommendation field improves average cost by 2.54–2.84% over the matched analytic pipeline while reducing median online time from 23.6 to 7.5–8.9 seconds. Heterogeneous learned fields bring additional gains over repeated single-model chains. In the separate nine-case inverse-field study, one Medium-trained model attains the lowest feasible cost among the main methods at all three observed order volumes. Search-component ablations have larger effects than replacing the learned fields, and random fields perform slightly better on one Large case. These results support the combined field-and-search procedure on the evaluated instances while limiting claims of uniform learning gains or unrestricted scale transfer.

REPRODUCIBILITY STATEMENT

The supplement provides identifier-replaced reference inputs and validated solutions, frozen checkpoints and inference inputs, portable search modules, fixed public-benchmark splits and configurations, per-instance outputs, and table and figure scripts. A separate validator reconstructs reported feasible solutions. Historical global-bound records are audited for identity and consistency, but the accumulated cut-generation campaign and all historical training-corpus versions are not rerun; Appendix B states the evidence scope.

ETHICS STATEMENT

The study uses freight demand records and makes no claim of live deployment or realized commercial savings. The provider authorized anonymous use and release of the reference data. Order, terminal and service identifiers are replaced, although this is not a formal privacy guarantee. Aggregate cost improvements can coexist with worse service for individual shipments; any deployment evaluation should report both cost and timeliness.

AI USE STATEMENT

Generative AI tools assisted with implementation, experiment design, mathematical formulation and proof drafting, literature search, analysis and manuscript editing. The authors independently checked the resulting code, mathematics, citations, experiments and claims and take responsibility for the final manuscript.

REFERENCES

Abdo Abouelrous, Laurens Bliek, Adriana F. Gabor, Yaoxin Wu, and Yingqian Zhang. Reinforcement learning for solving the pricing problem in column generation for routing. *Operations Research Perspectives*, 15:100364, Dec 2025. ISSN 2214-7160.

Ahmad Baubaid. Learning-assisted optimization for the stochastic service network design problem. *Transportation Research Procedia*, 96:79–86, 2026. ISSN 2352-1465.

Natashia Boland, Mike Hewitt, Luke Marshall, and Martin Savelsbergh. The continuous-time service network design problem. *Operations Research*, 65(5):1303–1321, 2017.

Cheng Chi, Amine Aboussalah, Elias Khalil, Juyoung Wang, and Zoha Sherkat-Masoumi. A deep reinforcement learning framework for column generation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 9633–9644. Curran Associates, Inc., 2022.

Teodor Gabriel Crainic. Service network design in freight transportation. *European Journal of Operational Research*, 122(2):272–288, 2000.

Matteo Fischetti and Andrea Lodi. Local branching. *Mathematical Programming*, 98(1–3):23–47, 2003.

Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large tsp instances. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):7474–7482, May 2021. ISSN 2159-5399.

Bernard Gendron, Saïd Hanafi, and Raca Todosijević. Matheuristics based on iterative linear programming and slope scaling for multicommodity capacitated fixed charge network design. *European Journal of Operational Research*, 268(1):70–81, 2018.

Amira Hijazi, Osman Y. Özaltın, and Reha Uzsoy. An improving column is all you need: Enhancing column generation for parallel machine scheduling via transformers. *INFORMS Journal on Computing*, June 2026. ISSN 1526-5528.

André Hottung and Kevin Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. arXiv preprint arXiv:1911.09539, 2019.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664. PMLR, 18–24 Jul 2021.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 09–15 Jun 2019.

Jiawei Lu, Tinghan Ye, Wenbo Chen, and Pascal Van Hentenryck. Boosting column generation with graph neural networks for joint rider trip planning and crew shift scheduling. arXiv preprint arXiv:2401.03692, 2024.

Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, Ravichandra Addanki, Tharindi Hapuarachchi, Thomas Keck, James Keeling, Pushmeet Kohli, Ira Ktena, Yujia Li, Oriol Vinyals, and Yori Zwols. Solving mixed integer programs using neural networks. arXiv preprint arXiv:2012.13349, 2020.

Yuan Sun, Andreas T. Ernst, Xiaodong Li, and Jake Weiner. Learning to generate columns with application to vertex coloring. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Madison Van Dyk and Jochen Koenemann. Sparse dynamic discretization discovery via arc-dependent time discretizations. arXiv preprint arXiv:2305.19176, 2023.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander Smola. Deep sets. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

A Run recipe and implementation details

The reported run starts from four field sources: the training-corpus activation prior at target supports 160/320 and 125/260, one latent-attention checkpoint at 125/260, and one DeepSets checkpoint at 160/320. For each target K, let $z_{(K)}$ be the Kth-largest own-mode logit. With $\mu = 0$, $\gamma = 2$ and $\tau = \log(\sqrt{2} - 1)$, the builder uses $T = \mathrm{clip}(z_{(K)}/\tau, 10^{-3}, 10^{3})$ when $z_{(K)} < 0$, and $T = 10^{-3}$ otherwise. The smooth-field boundary equals $F_s/2$ when $z_{(K)} < 0$ and clipping is inactive. It emits a smooth field and a gated field that restores the full charge F_s outside the 256 highest-scoring cells, giving four fields per source. Fallback, clipping, tied logits and gating can change the achieved support from the target K. These rules use only frozen logits and prespecified support targets; teacher labels are not read online. The release includes sanitized tensor values, model and training configurations, selected epochs, numerical inference inputs and a standalone inference program. This recovers frozen-model reproduction, not a rerun of historical training.

In the historical reference pipeline, 1,602 shipments are aggregated into 919 demand contexts and scored against 61,341 mode-free departure cells. Each cell has timetable and service features, a learned lane embedding, and analytic demand-pressure channels summarizing route relevance, early departures, and tight-deadline demand. The pooling weight is $v_q = \sum_{k \in \mathcal{K}_q} w_k$ in the recorded weight units. The reference hard limit satisfies $\bar{d}_k = d_k + 2{,}880$, and equal recorded weights in the released data have equal lateness penalties; hence the context key and empirical weight distribution retain these attributes without separate grouping coordinates. The latent-attention configuration uses $R = 64$, hidden width 256, eight heads, and 2/4/2 context-cross, latent-self, and departure-query blocks. Frozen checkpoint configurations and inference inputs are included in the reproduction artifacts.

The historical training corpus consists of resampled and subsampled demand scenarios derived from the single reference setting, rather than additional real operating days. Heuristic solves label the larger derived scenarios, while exactly solved small scenarios provide higher-quality labels where available. A corpus record contains the features of one derived instance I_i and one or more independently checked feasible teacher solutions D_{ij}. For every departure cell, it stores

$$r_{ij\ell} = \mathbf{1}\{D_{ij} \text{ uses an option } s \text{ with } \ell(s) = \ell \text{ and } m_s = \text{own}\} \tag{12}$$

and the teacher load assigned to owned options in that cell, normalized by the reference vehicle capacity. Outsourced vehicle counts and loads are stored separately. Multiple solver checkpoints for the same instance are retained because they may activate different useful service sets. Their nonnegative quality weights q_{ij} are normalized within the record, so a better checkpoint contributes more without making its service set the unique target.

For $p_{i\ell} = \sigma(z_{i\ell})$, the historical activation objective is an imbalance-aware focal loss. It downweights easily classified inactive cells and divides by the number of active cells; active-cell weights further emphasize teacher load and fixed vehicle cost. The auxiliary head predicts normalized load only on active cells and uses a Huber loss. The combined objective is

$$\mathcal{L}_{\text{hist}}(\theta) = \sum_i \sum_j \bar{q}_{ij}\, \mathcal{L}_{\text{focal}}\big(z_\theta(I_i), r_{ij}\big) + \lambda \sum_i \mathcal{L}_{\text{Huber}}\big(\widehat{L}_\theta(I_i), L_i\big), \tag{13}$$

where $\bar{q}_{ij} = q_{ij} / \sum_j q_{ij}$, and the load term is evaluated only at cells activated by the designated teacher checkpoint. The recovered trainer uses focal parameters 0.25 and 2, clips load–cost cell weights to $[0.25, 8]$ after mean normalization, and uses Huber threshold 0.1. The load head regularizes the shared representation but is discarded after training. A frozen encoder therefore outputs only the logits $(z_\ell)_{\ell \in \mathcal{L}}$ used by Equation 7; it does not output paths, vehicle counts, or a feasible network design. Path feasibility and joint resource constraints remain the responsibilities of the oracle and restricted master.

The seven historical checkpoints were trained under heterogeneous corpus generations and weighting variants. Their tensors, configurations and inference inputs are retained, but not every historical corpus generation is available for an end-to-end rerun. Accordingly, the reference experiment is an instance-specific development case. Appendix E gives the separately generated and fully recorded train/validation/test protocol used for held-out evaluation.

Reference candidate generation. Each field yields one exact path per shipment. For the first field, an additional oracle call excludes the least-used service–mode option on each path among those with near-zero charge, or among all its options if none has near-zero charge; ties prefer the latest option on the path. Whole trajectories are also shifted forward by 60 and 120 minutes when the corresponding services exist and release, connection and deadline constraints remain satisfied. Paths are deduplicated by shipment and service sequence. A zero-field call supplies any missing shipment candidate. The reference oracle uses a cached event network and label processing; no complete path family is enumerated.

Reference feedback neighborhoods. LOO calls share a fixed incumbent snapshot, temporarily removing and restoring one shipment at a time. Choices exceeding a vehicle limit receive search charge 10^{12}; terminal constraints are enforced in the master. Closure sources are ordered by a weighted sum of standardized avoidable vehicle cost, cost per user, same-lane departure surplus and reachable alternatives, with weights $1, 0.8, 0.5, 0.5$. Shared closure charges approximate the state after removing the affected group: vacated services regain their charge, while residual loads and the group's weight range adjust charges on remaining services. These are proposal charges, not exact group marginal costs. The source is forbidden; free rerouting is supplemented by target hypotheses on the same lane and mode. Targets are ranked by reachable affected users with a 50% bonus for incumbent services. A commitment hypothesis makes the target free and requires it for eligible shipments; an optional cost-sharing variant divides its vehicle cost among affected users without requiring it.

The common chain schedule is

```
loo,n1:30:2,rmp:60,loo,n1:none:6,rmp:35,
loo,n1:none:6,rmp:35,loo,n1:none:6,rmp:68
```

The first N1 stage skips source services with more than 30 users and uses two targets; subsequent N1 stages remove this cap and use six targets. The initial RMP budget is 90 seconds and the final per-chain budget is 45 seconds. The archived reference driver assumes that initialization yields a validated incumbent; it does not implement general feasibility restoration if initialization fails. The original union run uses 24 generation workers per chain, six decode workers per source, up to four source jobs concurrently, and eight MIP threads per chain under the driver's worker allocation. Final solves use 32 threads and successive budgets of 300 and 600 seconds, warm-starting from the best available validated solution.

Historical feature preprocessing, corpus generation and training precede the 807.08-second clock; it is not a raw-CSV-to-solution measurement. The global lower bound comes from a separate historically accumulated computation, not from this online run.

B COST RECONSTRUCTION AND RESTRICTED BOUNDS

A separate six-chain field-diverse union run has an independently validated cost of 363,703.2746899997 and a recorded historical wall time of 3,470.27 seconds. Its cost components reconstruct as

$$93{,}463 + 264{,}066.2746899997 + 6{,}174 = 363{,}703.2746899997. \tag{14}$$

This union result is distinct from the selected 807.08-second run, whose validated cost is $364{,}865.72469$. We retain the latter as the main reported operating point. Against the same global lower bound, the union result has a gap of 2.808%; its lower cost is not assigned the 807.08-second runtime. Both times are historical records, not fresh measurements; the retained evidence includes exported solutions, independent validation records and receipt hashes, but not the original receipt bodies.

The supplied global-bound record gives $L = 353{,}490.6928953997$. The reported feasible solution has $U = 364{,}865.72468999965$. Thus $U - L = 11{,}375.03179459995$ and $100(U - L)/U = 3.1175939599\%$. If the denominator is the unknown optimum z^*, then $L \le z^* \le U$ implies only

$$\frac{U - z^*}{z^*} \le \frac{U - L}{L} = 3.2179154991\%. \tag{15}$$

We use the first convention consistently. For the separate six-chain union run, the recorded restricted-solve gap corresponds to a restricted-master bound of approximately 345,463.12520. This is a historical pool-specific diagnostic and is not used as a global certificate.

The lower-bound computation used for the reported 1,602-shipment gap is a separate historical pipeline, distinct from the general bucketed relaxation in Appendix B.1. It constructs the complete model over the shipment path/arc representation, loads a historically accumulated pool of globally valid cuts, replays the seed cuts against the saved model fingerprint, checks the per-shipment floor records, and then solves the resulting root-bound model to obtain a valid bound for the original instance. This procedure never uses the restricted candidate pool produced by WAVECAST; it is an offline certificate source used only to contextualize the independently validated feasible incumbent. The released audit records therefore bind the lower bound to the model identity, cut replay, shipment-floor checks and final bound value, while excluding the cost of accumulating those cuts from the online runtime of the proposed algorithm.

The complete arc audit has peak process RSS 105.36 GiB and stops at the solver's memory limit. Its raw incumbent overcommits 15,976 services; removing excess vehicles accounts for the discrepancy with independently reconstructed cost. The original objective-equality check and the feasible corrected baseline are both retained. A finite candidate domain and configured runtime need not yield an exact restricted optimum.

Evidence has distinct scopes. All fourteen retained reference control solutions and the best solution are independently replayed against identifier-replaced data. Their runtimes remain historical. All seven checkpoint tensors match their original recorded MD5 identities before sanitization; inference inputs and production search modules are recovered. Historical training-corpus generations are only partially recovered and are not presented as fully reproduced. The native models, training corpus and held-out experiments are newly generated with complete manifests.

The global-bound records identify a 1,952,437-column, 2,004,475-row model with 19,967 cuts and an SHA-256 fingerprint of the saved model. They record replay of 18,953 seed cuts and checks for all 1,602 shipment floors. A historical seeded campaign takes 14,721.91 seconds, excluding earlier cut accumulation; a confirmatory root solve records bound $353{,}490.6928953997$. This paper freshly checks record identities and consistency, but does not rerun the full accumulated cut pipeline or that bound solve. The exported records preserve this distinction.

For the six-chain union run, the stored-cost audit checks that each pool record's internal cost equals its saved handling plus lateness cost; it does not independently recalculate the full pool. Independent cost and feasibility reconstruction is performed for the complete selected solution. Every selected path occurs in the pool, all 1,602 orders have candidates, and the trajectory identifiers are unique. The pool's minimum/median/mean/maximum paths per order are 10/54/64.47/309.

B.1 Global lower bound by exact-path Lagrangian relaxation

The restricted integer masters used by WaveCast construct feasible schedules, but their relaxation objectives need not be global lower bounds. This section gives an independent bound from the LP relaxation of a bucketed capacity model; the historical bound used for Reference-1602 comes from the separate procedure described in Appendix B. The construction has two parts: a restricted master provides candidate LP dual variables, and exact full-domain path pricing separates the missing path-dual constraints.

Bucketed capacity resources. The lower-bound model relaxes exact-departure capacity coupling by pooling services on the same physical lane $\ell_s = (i_s, j_s)$ into 12-hour resources according to departure time t_s^{dep}. Let $B = 720$ minutes and define the resource of service s by

$$R(s) = \left(\ell_s, \left\lfloor \frac{t_s^{\mathrm{dep}}}{B} \right\rfloor\right), \qquad s \in \mathcal{S}. \tag{16}$$

The ownership mode m_s is not part of r, so owned and outsourced vehicles share the same resource row but retain separate vehicle variables. For mode $\nu \in \{\mathrm{own}, \mathrm{out}\}$, let

$$\mathcal{S}_r^\nu = \{s \in \mathcal{S} : R(s) = r,\ m_s = \nu\}. \tag{17}$$

Let $\mathcal{R} = \{R(s) : s \in \mathcal{S}\}$ be the set of resources.

Aggregation requires services in each nonempty $\mathcal{S}_r^\nu$ to share the same per-vehicle cost and capacity, which the implementation checks. We then write $F_r^\nu = F_s$ and $Q_r^\nu = Q_s$ for $s \in \mathcal{S}_r^\nu$ and aggregate their vehicle limits:

$$\bar{y}_r^\nu = \sum_{s \in \mathcal{S}_r^\nu} \bar{y}_s. \tag{18}$$

Empty resource–mode pairs are omitted from sums over ν. The pooling relaxes capacity coupling; release times, exact departures, connections, destination arrival times and path costs c_{kp} retain their original definitions.

Define the number of uses of resource r by path p using the service-incidence coefficient a_{ps} from Section 3:

$$m_{kpr} = \sum_{s \in \mathcal{S}} a_{ps}\, \mathbf{1}\{R(s) = r\}. \tag{19}$$

With x_{kp} relaxed to be continuous and y_r^ν denoting the vehicle mass assigned to resource r and mode ν, the lower-bound relaxation of the original formulation (1)–(6) is

$$v_{\mathrm{LB}} = \min_{x,y} \quad \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k} c_{kp} x_{kp} + \sum_{r \in \mathcal{R}} \sum_{\nu} F_r^\nu y_r^\nu \tag{20a}$$

$$\text{s.t.} \quad \sum_{p \in \mathcal{P}_k} x_{kp} = 1 \qquad \forall k \in \mathcal{K}, \tag{20b}$$

$$\sum_{\nu} Q_r^\nu y_r^\nu - \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}_k} w_k m_{kpr} x_{kp} \geq 0 \qquad \forall r \in \mathcal{R}, \tag{20c}$$

$$\sum_{\nu} y_r^\nu - \sum_{p \in \mathcal{P}_k} m_{kpr} x_{kp} \geq 0 \qquad \forall k \in \mathcal{K},\ \forall r \in \mathcal{R}, \tag{20d}$$

$$x_{kp} \geq 0, \qquad 0 \leq y_r^\nu \leq \bar{y}_r^\nu. \tag{20e}$$

Constraint (20c) pools the vehicle capacities of exact departures in the same resource. Constraint (20d) preserves the fact that if a positive-weight shipment uses a departure, at least one vehicle must be activated for that use; if a path uses the same resource multiple times, the corresponding multiplicity is required. Compared to the original model (1)–(6), this formulation additionally relaxes integrality and omits the daily hub-processing constraints (4). Therefore every feasible solution of (1)–(6) is also a feasible solution of (20) by aggregating its exact service variables, and hence

$$v_{\mathrm{LB}} \le z^*, \tag{21}$$

where z^* is the optimal value of the original formulation.

The full LP dual. Associate a free dual variable σ_k with assignment (20b), and nonnegative variables λ_r and π_{kr} with capacity (20c) and activation (20d), respectively. Write each vehicle upper bound as $-y_r^\nu \ge -\bar{y}_r^\nu$ and associate a nonnegative dual variable γ_r^ν with it. The full dual is

$$v_{\mathrm{D}} = \max_{\sigma,\lambda,\pi,\gamma} \quad \sum_{k\in\mathcal{K}} \sigma_k - \sum_{r\in\mathcal{R}} \sum_{\nu} \bar{y}_r^\nu \gamma_r^\nu \tag{22a}$$

$$\text{s.t.} \quad \sigma_k - \sum_{r\in\mathcal{R}} (w_k\lambda_r + \pi_{kr}) m_{kpr} \le c_{kp} \qquad \forall k\in\mathcal{K},\ p\in\mathcal{P}_k, \tag{22b}$$

$$Q_r^\nu \lambda_r + \sum_{k\in\mathcal{K}} \pi_{kr} - \gamma_r^\nu \le F_r^\nu \qquad \forall r\in\mathcal{R},\ \nu, \tag{22c}$$

$$\lambda_r, \pi_{kr}, \gamma_r^\nu \ge 0, \qquad \sigma_k \in \mathbb{R}. \tag{22d}$$

Each primal path variable gives one constraint (22b); its column has coefficient 1 in its assignment row, $-w_k m_{kpr}$ in capacity row r, and $-m_{kpr}$ in activation row (k, r). Each vehicle variable gives constraint (22c). Thus the signs and the upper-bound correction follow directly from the primal columns. The variables λ_r, π_{kr} are also the Lagrange multipliers of the same two coupling constraints; they are not a separate set of heuristic service charges.

For every feasible point of (22), LP weak duality gives

$$\sum_{k\in\mathcal{K}} \sigma_k - \sum_{r\in\mathcal{R}} \sum_{\nu} \bar{y}_r^\nu \gamma_r^\nu \ \le\ v_{\mathrm{LB}} \ \le\ z^\star. \tag{23}$$

The first inequality is duality for the bucket LP; the second is the original-to-bucket relaxation. When the bucket LP is feasible with a finite optimum, strong LP duality additionally gives $v_{\mathrm{D}} = v_{\mathrm{LB}}$. Equality with $z^\star$ is neither assumed nor required.

A full-dual feasible bound at every pricing pass. For nonnegative (λ, π), define the full-domain adjusted path cost and minimum reduced cost by

$$\phi_k(\lambda,\pi) = \min_{p\in\mathcal{P}_k} \left[c_{kp} + \sum_{r\in\mathcal{R}} (w_k\lambda_r + \pi_{kr}) m_{kpr} \right], \tag{24}$$

$$\mathrm{rc}_k(\sigma,\lambda,\pi) = \phi_k(\lambda,\pi) - \sigma_k. \tag{25}$$

Here $\mathcal{P}_k$ is the complete feasible path family from Section 3, not the current candidate pool. Let G_k be a shipment-specific event graph, augmented with any state needed for route restrictions, whose source–sink paths represent exactly this family. Under the additive-cost conditions in Section 4, assigning each service use the additional charge $w_k\lambda_{R(s)} + \pi_{kR(s)}$ reduces (24) to shortest-path pricing on G_k. Exact pricing requires a complete representation and an exact minimum over it. A topological pass suffices for an acyclic timetable graph; zero-duration transitions are not assumed to preserve acyclicity and require an appropriate exact graph algorithm.

Column generation need not finish before a global bound is available. Given any nonnegative (λ, π), complete the dual variables by setting

$$\widehat{\sigma}_k = \phi_k(\lambda,\pi), \qquad k\in\mathcal{K}, \tag{26a}$$

$$\widehat{\gamma}_r^\nu = \max\left\{0,\ Q_r^\nu\lambda_r + \sum_{k\in\mathcal{K}} \pi_{kr} - F_r^\nu\right\}, \qquad r\in\mathcal{R},\ \nu. \tag{26b}$$

Algorithm 2 Bucket-LP column generation and dual-feasible bounds: unsmoothed core

Require: Complete event graphs G_k admitting exact path pricing; bucket parameters; initial pools $P_k \subseteq \mathcal{P}_k$ with a feasible RMP
Ensure: Best full-dual lower bound L_{best} among completed passes
$L_{\text{best}} \leftarrow L(0,0)$ by complete zero-price shortest paths
for each iteration within the computational budget **do**
  Solve the RMP LP to optimality; read row duals (σ, λ, π)
  $\mathcal{C} \leftarrow \varnothing$
  for each shipment $k \in \mathcal{K}$ **do**
    Compute $(\phi_k, p_k^\star)$ by shortest-path pricing on G_k
    if $\phi_k - \sigma_k < 0$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{(k, p_k^\star)\}$
    end if
  end for
  Complete $(\widehat{\sigma}, \widehat{\gamma})$ by (26)
  $L_{\text{best}} \leftarrow \max\{L_{\text{best}}, L(\lambda, \pi)\}$
  if $\mathcal{C} = \varnothing$ **then**
    break ▷ Full-dual feasibility at the optimal RMP dual
  end if
  Add paths in $\mathcal{C}$ and all their capacity and activation rows to the RMP
end for
return L_{best}

By the definition of a minimum, $\widehat{\sigma}_k$ is no greater than the adjusted cost of any path, so every constraint (22b) holds. The choice of $\widehat{\gamma}$ satisfies every constraint (22c). Consequently, $(\widehat{\sigma}, \lambda, \pi, \widehat{\gamma})$ is feasible for the full dual, regardless of whether the original RMP assignment duals were feasible there. For fixed (λ, π), these choices also maximize the objective over σ and γ.

Define the adjusted vehicle coefficient

$$\eta_r^\nu(\lambda, \pi) = F_r^\nu - Q_r^\nu \lambda_r - \sum_{k \in \mathcal{K}} \pi_{kr}. \tag{27}$$

Substituting (26) into the full dual objective gives

$$\begin{aligned} L(\lambda, \pi) &:= \sum_{k \in \mathcal{K}} \widehat{\sigma}_k - \sum_{r \in \mathcal{R}} \sum_{\nu} \bar{y}_r^\nu \widehat{\gamma}_r^\nu \\ &= \sum_{k \in \mathcal{K}} \phi_k(\lambda, \pi) + \sum_{r \in \mathcal{R}} \sum_{\nu} \bar{y}_r^\nu \min\{0, \eta_r^\nu(\lambda, \pi)\}. \end{aligned} \tag{28}$$

Thus, by (23),

$$\boxed{L(\lambda, \pi) \le v_{\text{LB}} \le z^\star, \qquad \lambda, \pi \ge 0.} \tag{29}$$

Column generation and termination. Algorithm 2 gives the column-generation procedure. At each RMP solve, rows required by hosted columns must be present; rows with no hosted path use are redundant in the restricted model and their absent duals are interpreted as zero. At an optimal RMP primal–dual pair, if full pricing finds $\text{rc}_k \ge 0$ for every shipment, then the RMP dual is also full-dual feasible. The RMP primal extends to the full bucket LP by setting unhosted variables to zero, giving

$$\text{RMP dual objective} \;\le\; v_{\text{LB}} \;\le\; v_{\text{RMP}} \;=\; \text{RMP dual objective}. \tag{30}$$

Hence column generation has solved the bucket LP relaxation. Even without this termination condition, (23) remains valid for every completed full-domain evaluation.

B.2 Historical Reference-1602 training splits, checkpoints and recorded metrics

The checkpoint manifest and supplementary Reference-1602 training materials provide the evidence for Tables 2 and 3. The former retains model tensors, standardizers, model/training configurations, selected epochs and file identities; the latter supplies the corresponding `receipt.json`,

`metrics.jsonl`, sampling plans and training commands. We transcribe the saved settings and the validation row at each recorded selected epoch, without retraining or adding new model evaluations.

Corpus and split definitions. Corpus A contains 355 silver-teacher and 11 exact-solver records; Corpus B contains 413 silver-teacher, 51 exact-solver and 390 self-training records. A record can include multiple validated designs, so record counts differ from design-checkpoint counts. Both corpora use the same 61,341-cell lattice. A family comprises one sampling mechanism and scale parameter, such as `sub0.50` or `boot1.00`; all records in a family are assigned to the same side. The original C1/C2 pair holds out these two families. For C3, the held-out families are `boot1.00`, `boot1.50` and `sub0.25`; for C4, they are `boot0.26`–`boot0.30`, `boot1.00`, `boot1.50` and `sub0.65`. These two runs select approximately one quarter of the families using split seed 20260814, not one quarter of individual orders. The three Corpus B runs C5–C7 return to the original two-family holdout and discard 51 training-side teacher twins in favor of their exact labels, leaving 724/79 records. These family-separated validation sets share the reference demand source and network; they are not independent external test sets.

Architecture and optimization. All seven checkpoints predict one own-mode activation logit and an auxiliary normalized owned load per mode-free departure cell; the load output is not used for decoding. DeepSets uses hidden, context and pooled widths of 128. Latent attention uses width 256, 64 latent tokens, eight attention heads, feed-forward width 1,024, two context cross-attention blocks, four latent self-attention blocks and two query cross-attention blocks. Both architectures use 16-dimensional embeddings for the 69 reference lanes and dropout 0.05. AdamW uses weight decay 0.01, accumulation over eight instances, a 10% linear warm-up and cosine decay to 5% of the initial learning rate. The activation objective is quality-weighted focal loss with $\alpha = 0.25$ and $\gamma = 2$, plus 0.1 times an auxiliary Huber load loss with $\delta = 0.1$. The first four checkpoints use load-times-cost cell weighting and gradient clipping at 1; checkpoints C5–C7 trained on Corpus B use avoidable-charge weighting and clipping at 5. The latter allocate epoch weight 0.35/0.20/0.10/0.35 to exact/warm/silver/self-training tiers. Training seeds are 20260818 for the original pair and 20260820 for the other five runs. Standardizers are fitted only to each run's retained training records.

Table 2: Released Reference-1602 checkpoint provenance. DS denotes DeepSets and LA latent attention. Selected epochs retain the zero-based indices in the saved files.

Checkpoint	Net	Corpus	Train/val	Epoch cap	Epochs run	Selected	Initial LR	H100 seconds
C1	DS	A	287/79	40	37	30	10^{-3}	254.04
C2	LA	A	287/79	150	42	26	10^{-3}	860.23
C3	DS	A	275/91	40	34	27	10^{-3}	572.83
C4	DS	B	641/213	40	37	30	10^{-3}	1,238.72
C5	DS	B	724/79	150	150	144	5×10^{-4}	7,945.98
C6	DS	B	724/79	40	40	29	5×10^{-4}	2,817.83
C7	LA	B	724/79	150	150	104	5×10^{-4}	12,647.66

Notes. A and B denote the two training corpora; C1–C7 identify model checkpoints. Checkpoints maximize validation load-weighted Recall@200. Early-stopping patience is 6 for C1/C3/C4, 15 for C2, and disabled for C5–C7. Times are the original recorded training times, excluding label generation. The supplementary naming key maps C1–C7 to the original checkpoint aliases; full exported SHA-256 and historical source MD5 identities are retained in the checkpoint manifest.

Recorded validation statistics. For each validation record, the trainer scores the design with the largest stored quality weight, using the first maximum in a tie, and averages the resulting metrics equally across records. Recall@K is the fraction of activated cells recovered among the K largest logits; load-weighted recall measures the fraction of activated-cell load recovered there. Brier score is the mean squared error between sigmoid scores and binary activation labels, and ECE uses ten equal-width probability bins. Table 3 contains only metrics present in the original histories. ROC-AUC and PR-AUC/average precision were not recorded in these histories or the coauthor checkpoint manifest and are not recomputed here. The extremely sparse activation targets make Brier and

ECE insufficient on their own; ranking recall is also reported. These are validation diagnostics of selected checkpoints, not transportation costs or independent test performance. Different held-out families and training variants prevent an unconditional ranking of all seven rows as one controlled experiment.

Table 3: Saved validation metrics at the selected Reference-1602 checkpoints. Higher recall is better; lower Brier and ECE is better. No AUC values are imputed.

Checkpoint	Recall@200	Recall@400	LW-R@200	LW-R@400	Brier	ECE
C1	0.6607	0.7569	0.8744	0.9221	0.001829	0.013736
C2	0.6982	0.7817	0.8909	0.9272	0.001493	0.007748
C3	0.6226	0.7143	0.8561	0.9016	0.001887	0.013730
C4	0.5970	0.7003	0.7962	0.8690	0.001838	0.012795
C5	0.6531	0.7552	0.8571	0.9101	0.002249	0.016250
C6	0.6089	0.7180	0.8301	0.8935	0.002534	0.018057
C7	0.7121	0.8012	0.8842	0.9279	0.001742	0.010465

C Interface property and limits of restriction

Proposition 1 (Interface property)**.** *Assume every stored path satisfies its shipment-level constraints, its stored cost and resource coefficients equal those of the original model, and the restricted master returns a solution satisfying Equations 2–4 and the stated integrality constraints. Then its reconstructed assignment is feasible for the original path formulation, and its original cost is an upper bound on the global optimum, independently of the field values.*

Proof. Each selected candidate belongs to its original path family. Extending the restricted assignment by setting all absent path variables to zero preserves all constraints and its objective. Thus it is an original feasible solution, whose cost bounds the minimum from above. □

This standard restriction argument explains the interface; it is not a new optimality theorem. Numerical tolerances still require independent reconstruction. A finite field portfolio need not recover any given optimal solution. Even a field that perfectly identifies useful services can generate insufficiently diverse paths when capacity competition matters. The empirical teacher-field reconstruction check in Appendix D is therefore a regression test, not a general sufficiency guarantee. Retaining incumbents makes the *reported* best value monotone; adding columns need not improve a time-limited solver's newly found incumbent.

D Why service labels and fields require care

A historical diagnostic gives zero search charge to services in an earlier validated design and full charge elsewhere. Exact decoding followed by master optimization reproduces that earlier objective, 366,485.73554. It supports using service design as a useful learning target on that example. It does not show that every feasible or optimal design can be recovered by rank-one decoding. For instance, two shipments can choose the same cheapest path under a perfectly identified service set and exceed its capacity, while one needs its second-best path. Multiple fields, exclusions, shifts, and incumbent-driven enrichment help address this loss of combinations.

The output scores also need not be calibrated activation probabilities. Focal loss, temperature changes, and support targeting make the logits operational ranking signals. Interpreting individual attention weights as causal explanations of shipment coordination would require additional analysis. Source-removal ablations test the usefulness of a model's generated candidates, not the causal role of every hidden feature.

For a fixed field, a path oracle is exact for the stated single-shipment objective and path domain. This exactness should not be confused with joint optimality over shipments. The master is responsible for capacity coupling, and the independent validator is responsible for confirming that its output corresponds to the original operational rules. Group-closure charges are also search devices: the master restores the true fixed charges after any suggested collective move.

E NATIVE EXTERNAL BENCHMARK AND TRANSFERABLE INPUTS

Native problem. The public SND-RR instances differ from the reference application. Each arc has its own vehicle capacity, fixed departure charge, and commodity-specific variable transport cost. Integer numbers of vehicles can depart at any time; deadlines are hard. Designated-path and regional restrictions limit the arc set of each commodity. There are no reference-specific terminal handling limits, owned/outsourced modes, weekly timetable, or lateness penalties. We preserve these distinctions. A solution records each commodity's original arc IDs and departure times; a separate validator checks permissions, continuity, release and arrival times, and recomputes variable cost plus the integer number of vehicles needed at every shared departure. Thirty of the 576 instances contain zero-transit arcs; these are retained.

Exact time grid. All input release times, deadlines and arc durations in this benchmark are integers. The implementation rejects noninteger values. Consider any feasible continuous-time solution. Replace each departure t by $\lfloor t \rfloor$. For an integer duration τ, precedence $t' \geq t + \tau$ implies $\lfloor t' \rfloor \geq \lfloor t \rfloor + \tau$. Integer releases and deadlines remain satisfied. Shipments that shared an arc departure before rounding still share it afterwards. Distinct departure groups may merge; because $\lceil (a+b)/Q \rceil \leq \lceil a/Q \rceil + \lceil b/Q \rceil$, merging cannot increase the number of vehicles. Variable costs are time-independent and nonnegative; removable cycles do not improve cost. Therefore an optimal solution exists on the integer grid, and the complete integer-time arc model supplies a global bound for the native problem. This argument relies on the benchmark's unlimited departure availability and does not replace the reference timetable.

Graphs and splits. The frozen archive contains 576 generated instances, 192 per family: critical-times, designated-paths, and hub-and-spoke. Variants of each source base stay in one partition, as do identical labelled directed topologies. The resulting 80 groups are split into 48 training, 16 validation and 16 test groups, containing 348, 120 and 108 instances. An additional exact directed-graph isomorphism audit of the 80 base representatives finds no isomorphic pairs even when node names and edge attributes are ignored. These are independent generated networks under shared generators, not new real operating days. The split uses input metadata only.

Scale stress protocol. Before generating stress outcomes, we fix variant zero from each of the 16 test source bases. Each source commodity is retained once, then commodities are sampled with replacement to reach 1,600. Every resulting weight is multiplied independently by a uniform factor in $[0.8, 1.2]$. Release and deadline move together by an integer in $\{-2, -1, 0, 1, 2\}$, clipped to keep release nonnegative, preserving time-window length. Topology, capacities, fixed and variable unit costs, and route restrictions are unchanged. The generator seed is a deterministic hash of the parent instance. These descendants retain their parent split/group. A separate training-network case is used only for resource calibration. All three frozen architectures use prespecified training seed 11 without retraining, compared with analytic search, full MIP cold/warm and arc/node DDD: 128 runs at 120 nominal seconds and eight threads. Four processes are interleaved on one second CPU host. Stress and primary wall times are not compared across hosts as a speedup. This is a secondary demand-scale shift, not additional independent networks or observed demand periods. Matching the reference shipment count does not match its long timetable, event-graph size, fleet limits or handling constraints.

Structural adaptation. We retrain the existing CellMLP, DeepSets and latent-attention encoder implementations. A service cell is an arc and feasible integer departure. Its 29 features describe duration, capacity, fixed cost, endpoint degrees and service statistics, hub status, departure time and global network statistics. Nine input-derived pressure features summarize reachable shipment mass, count, temporal flexibility, cost and capacity pressure. Each commodity contributes a 28-dimensional context describing its endpoints, release/deadline, weight and transport-cost statistics. Log transforms and instance-relative time scales are fixed in code. Original node, arc and network IDs are not input features. To reuse the original architecture interface, the lane embedding is a single frozen zero scalar shared by all cells. CellMLP still receives the demand-pressure features; it is an ablation of learned context aggregation, not a demand-blind model.

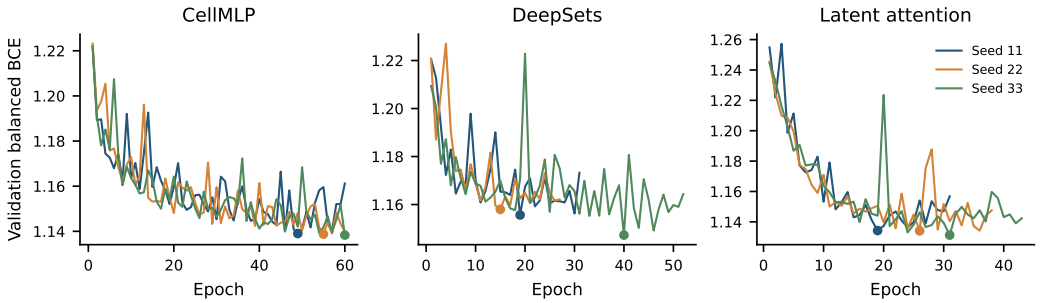


Figure 2: All nine native-model validation-loss histories. Dots mark the selected checkpoint; no test cost enters early stopping. Different stopping epochs are retained, rather than selecting the best-looking training seed.

Table 4: New native-model training. Epochs and wall seconds are ordered by seeds 11/22/33. Times include model setup and training on one H100; they are process wall times, not measured CUDA kernel durations. All model tensors are provided.

Architecture	Parameters	Selected epochs	Training wall (s)
CellMLP	22,147	49 / 55 / 60	67.5 / 65.7 / 64.8
DeepSets	108,419	19 / 15 / 40	74.4 / 64.0 / 115.0
Latent attention	1,039,107	19 / 26 / 31	290.7 / 355.8 / 398.3

Training. A full original-cost arc MIP, warm-started with full-charge individually shortest paths, produces one independently validated teacher per training and validation instance under a 30-second online limit. Labels mark cells used by that feasible teacher; they are not claimed to be optimal labels. The validation teachers are used only for checkpoint selection. We use balanced binary cross-entropy on activation, with positive weight $\min(100, \max(1, (n - n_+)/n_+))$, AdamW (learning rate 3×10^{-4}, weight decay 10^{-4}), gradient clipping at norm one, and at most 60 epochs. Early stopping selects minimum mean validation loss with patience 12. Seeds are 11, 22 and 33. Hidden/global/context widths are 128; the attention model uses 32 latents, four heads, a feed-forward width of 512 and 2/2/1 context-cross/self/query blocks, with dropout 0.05. The reference checkpoints and their heterogeneous training losses are separate artifacts.

All 468 teacher runs return independently validated labels. With six processes and eight solver threads each, teacher generation takes 1,704.70 seconds wall time, 10,024.37 summed online seconds and 50,757.34 summed process CPU seconds, with maximum per-unit RSS 670.44 MiB. Nine training programs total 1,496.10 seconds; suite overhead brings wall time to 1,514.90 seconds. Peak PyTorch-allocated GPU memory is 156.92 MiB, excluding CUDA context and other process allocations. The supplied raw histories and configurations retain the exact model/label hashes. These offline costs are not included in the new-instance online budgets.

Matched search. Each native control uses one search chain; each learned arm uses one encoder. The heterogeneous historical six-chain union is evaluated separately on the reference case. All native field controls use the same decoder and incumbent expansion code. Common full-charge and variable-cost-only paths guarantee a feasible starting pool in the native unlimited-fleet model. Six further fields use support targets $0.5|K|$, $|K|$, $2|K|$ with and without fixed-seed jitter. Score ranks determine a support-centered nonnegative charge $F(1-\sigma(\cdot))^2$; no arc is removed solely because of a score. After an initial restricted master, LOO rerouting and closure of under-utilized services enrich the pool. Feasible integer shifts of $\pm 1, \pm 2$ time units provide additional columns. Up to three further enrichment rounds consume remaining time, stopping when no new columns are found. Restricted solves use true costs, and the best independently validated incumbent is retained. Ablations disable LOO or closure in this same implementation.

Table 5: Primary native resource accounting. CPU is median cumulative process CPU time; load/inference are median seconds; memory is the maximum process RSS in MiB. Zero load/inference means no neural model is used, while a dash denotes an unavailable field.

Method	CPU (s)	Load (s)	Infer. (s)	Max RSS	Max wall (s)
Analytic	119.0	0.000	0.000	292.9	32.0
Random	73.2	0.000	0.000	329.7	30.3
CellMLP	36.7	0.950	0.005	541.1	30.8
DeepSets	41.9	0.954	0.008	495.8	30.5
Latent attention	35.4	0.972	0.024	520.1	30.5
Full MIP cold	130.0	–	–	591.0	36.5
Full MIP warm	137.0	–	–	776.9	35.2
Arc DDD	92.4	–	–	799.3	31.0
Node DDD	103.4	–	–	775.7	33.7
Latent without LOO	48.0	0.972	0.024	508.1	30.2
Latent without closure	26.5	0.969	0.024	505.5	30.3

Table 6: Source-group mean savings against analytic search for every native training seed. These values expose retraining variability; the main paired intervals are conditional on the three frozen models.

Architecture	Seed 11	Seed 22	Seed 33
CellMLP	2.52	2.99	2.68
DeepSets	2.83	2.86	2.83
Latent attention	2.40	2.58	2.64

Baselines and numerical accounting. Full arc MIPs use the complete feasible integer grid, with and without the common full-charge warm start. Arc- and node-based DDD use the pinned authors' source with instrumentation for a common wall deadline, solver threads/seed, solution export and validation; the original one-percent stopping rule remains unchanged. Online timers begin at unit entry before input loading; Python interpreter startup and module imports executed before entry are excluded. Learned and nonlearned runs include input processing, graph construction, model loading and CPU inference, candidate generation, optimization and final validation. The learned model loader imports PyTorch inside that timer. Process CPU and peak RSS use cumulative operating-system resource counters. Solver termination and validation can cause small deadline overruns, which are retained in receipts. Near-integer solver variables multiplied by large fixed charges can give a floating objective below the cost of its integral interpretation. We record this difference and its coefficient-weighted rounding envelope; only independently reconstructed integer costs are reported as feasible upper bounds. Restricted-master bounds are never pooled as global bounds.

F Complete accounting and cost profiles

The result CSVs contain every scheduled arm–instance row, its validated original cost, feasible status, actual wall time, cumulative CPU time, peak RSS, model-loading/inference times when available, column counts, checkpoint hashes and valid global bounds. Missing objectives are retained. The pooled best observed feasible cost is used only for an empirical cost profile, never as a known optimum. Figure 3 averages the within-network fractions equally across source networks and includes all training seeds and failed attempts. A method that fails remains outside every finite cost threshold.

G Historical development operating points

G.1 Reference operating point

The reported reference run yields $U = 364{,}865.72468999965$. It uses 126 owned services and no outsourced services, with vehicle cost 94,185, handling cost 264,298.72469, and lateness cost 6,382.

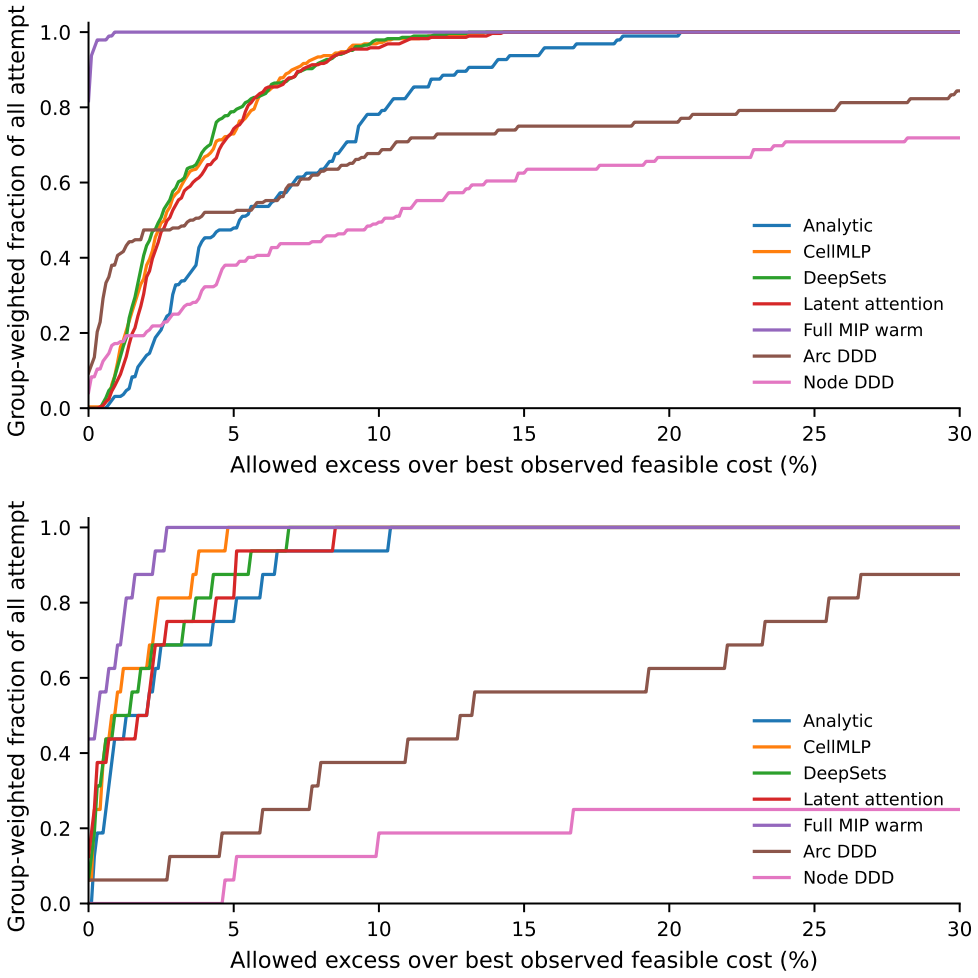


Figure 3: Group-weighted empirical cost profiles for primary native (top) and derived scale stress (bottom). The horizontal axis is excess over the best observed feasible cost across the fixed methods. This is not an optimality-gap or time-to-target plot. All scheduled attempts remain in the denominator.

Table 7: Development operating point reported in the main text. Lower objective is better.

Configuration	Objective	Online wall (s)	Services
Reported run	**364,865.72**	807.1	126

All 1,602 shipments pass the independent validator. Its recorded wall time is 807.08 seconds from prepared features and frozen field sources.

The global 3.118% bracket does *not* come from proving the restricted problem optimal. It combines a validated incumbent with an independently certified stronger lower bound.

G.2 Does the neural field add value?

An unconditional frequency prior is a useful control because it receives information from the same teacher corpus. Historical matched single-chain measurements give the ablation in Table 8. Removing either encoder worsens the objective. Removing a source also removes its four initial fields, so these controls confound information with proposal volume. They describe useful additions to

Table 8: Historical source-removal measurements on the development instance. Search plans match within this comparison. Each row is a recorded run, not a seed average. All retained solutions have been independently revalidated against identifier-replaced inputs; online timings remain those of the original solves.

Sources	Objective	Wall (s)	Increase vs full
Prior + latent + DeepSets	**364,865.72**	826	—
Prior + DeepSets	366,307.06	851	1,441.34
Prior + latent	366,184.55	876	1,318.83
Prior only	367,492.53	909	2,626.81

this development portfolio, not an isolated neural-information effect or a statistically established architecture advantage. The external comparison holds field counts and search code fixed.

The external fixed-field-budget experiment estimates the learning effect under a cleaner held-out protocol.

G.3 Diversity, search budgets, and failure modes

A matched four-chain comparison yields 364,163.62 when chains share the same DeepSets checkpoint but use different seeds, and 363,708.27 with different field sources. The 455.35 cost difference supports field diversity as a useful source of columns. Wall time changes from 2,796 to 2,694 seconds; the observed speed change is modest and does not justify attributing a large speedup to field diversity.

On a separate single-chain anchor of 366,315.94, two additional pricing rounds reduce cost by 1,450.22 while adding approximately 278 seconds. Allocating roughly thirteen times the final-MIP budget reduces cost by 95.20 while adding approximately 832 seconds. This comparison motivates spending budget on new candidates in that configuration. It is not a claim that pricing dominates all MIP work: the final cascade alone is 26.4% of the best original union's wall time, and master solves also occur inside every chain.

More fields and larger pools are not automatically better under fixed solve budgets. Development attempts with stronger small-instance validation scores failed to improve the reference batch, and retraining changed solution quality substantially. An experiment with three seeds per encoder variant shows considerable variability; these samples do not support a general variance-reduction theorem. We therefore report operating points and mechanism controls separately from generalization claims. The native external evaluation addresses held-out source networks with fixed architecture and search settings.

H Secondary experiments addressing alternative explanations

The following experiments were designed after inspection of the primary and initial stress results. They are secondary studies on the existing held-out source networks, not a new untouched test set. Their executable protocols were frozen before their own test outcomes. Resource calibration uses training instances; search-configuration selection uses validation networks only. We retain all scheduled outcomes, including failures and runs without a feasible incumbent. Repeated seeds and demand perturbations remain nested within the original source network for inference. Confidence intervals are unadjusted descriptive intervals across these multiple comparisons.

All seven secondary matrices are complete: 756 original affine comparisons, 540 affine-fitting follow-ups, 648 portfolio runs, 1,188 common-reservoir comparisons, 112 short-budget anytime runs, 32 fixed-pool master comparisons and 480 perturbation runs. All 3,756 final solutions pass independent reconstruction. The main anytime comparison uses all 16 original stress networks and seven methods at 120 seconds. The interrupted hour-budget campaign is retained separately; no incomplete matrix contributes a final estimate.

Linear scoring control. A logistic scorer uses the same 29 cell features and nine pressure features as CellMLP. It omits only the constant zero embedding and has 39 trainable parameters, including its intercept. Training uses the original 468 training/validation teachers, the same balanced activation loss, optimizer and validation-epoch rule, and seeds 11, 22 and 33. We rerun the three linear models, three CellMLP models and analytic control in an interleaved 756-unit campaign on the same host, retaining the original 30-second search budget and eight solver threads. This comparison tests whether nonlinear scoring improves on a learned linear combination of the informative features; it does not treat analytic search as the only non-neural alternative. Its timing starts at the original benchmark entry point, as in the primary experiment.

More thorough affine fitting. All three original affine controls selected their final allowed training epoch. A follow-up therefore fits the same 39 parameters using full-batch float 64 L-BFGS-B on the original 348 training teachers, with training-only feature standardization folded back into the original coordinates. Four original-coordinate L 2 penalties, $10^{-2}, 10^{-4}, 10^{-6}, 0$, are selected using only unregularized validation loss on 120 teachers. A bounded training-only exact-Hessian refinement addresses numerical stopping while retaining the declared 10^{-8} gradient targets in both coordinate systems. The initial unpolished fits are preserved. The selected checkpoint and all four fitted objectives/gradients are independently checked against the teacher data. The fresh 540-unit test interleaves this single selected affine scorer, analytic search and the three original CellMLP checkpoints on all 108 test instances, with the original 30-second search allowance. This remains a post-primary follow-up; no new independent test networks are claimed.

Heterogeneous fields versus repeated chains. The heterogeneous portfolio contains latent seeds 11 and 22, DeepSets seed 11 and CellMLP seed 11. Four controls each repeat one of these individual checkpoints across all four chains; a sixth arm uses four analytic chains. Every arm uses chain seeds 0–3 and the same support scales $0.5, 0.8, 1.25, 2.0$. Each of the four concurrent chains receives a 15-second limit and eight solver threads. The parent has a 60-second limit including startup, imports, input preparation, checkpoint loading, child execution and a final union master using eight threads. The same-checkpoint controls are essential because comparing only with repeated latent seed 11 would mix heterogeneity with the quality of the constituent models. Equal configured budgets and thread limits do not imply equal realized CPU time. The full matrix contains six arms on all 108 test instances.

Validated anytime comparison and analytic tuning. Four prespecified analytic search schedules are compared on one input-selected representative from each of the 16 validation source networks, using 30 seconds and eight threads. The selected schedule minimizes the mean cost ratio to the default analytic schedule across validation groups; ties follow the protocol's fixed order, and all validation solutions must be feasible. The test compares complete cold/warm MIP, default/tuned analytic search, and the three seed-11 neural models on the same 16 previously specified 1,600-demand stress instances, with a revised 120-second budget and eight solver threads per unit. The original 3,600-second full matrix was stopped for resource prioritization after 16 completed outcomes; all existing data are preserved separately.

The clock starts immediately before launching the worker process. An upper bound enters the trajectory only after its shipment paths have been reconstructed, independently validated against the raw input, repriced using exact decimal vehicle counts, and persisted. Full-MIP callbacks apply this rule to incumbent assignments rather than using the raw solver objective as the reported feasible cost. The main matrix evaluates availability at 10,15,30,60 and 120 seconds. The previously proposed long diagnostics are outside the current delivery scope. A method that terminates early carries its last valid solution forward, explicitly without claiming additional computation. Time-to-target uses fixed fractions $1.00, 0.95, 0.90, 0.80$ of the common input-only full-charge solution, not a hindsight best cost. Missing solutions and unattained targets retain their denominators. Because search allocates its phases using the total budget, a 120-second run's 30-second snapshot is not the same algorithmic schedule as a separately budgeted 30-second run.

Service versus path scoring on a common reservoir. We generate an input-only reservoir using analytic and random fields and retain the same full-charge anchor for every method. Training and validation receive a new restricted-master teacher on this reservoir; test reservoirs have no teacher solutions or targets. A path scorer and a service scorer use the same feasible teachers, train-only

feature normalization, three training seeds and candidate-retention rule. Each order retains up to eight candidates, with the common anchor consuming one slot. Analytic scoring, three random selections and the full reservoir provide controls; the 11 arms produce 1,188 test units. Every final master uses original costs, the common anchor and a 30-second construction-and-solve budget.

This is a comparison of representation systems: the path scorer has 112 input features and 31,105 parameters, whereas the service scorer has 66 features and 25,217 parameters. Context aggregation, loss normalization and the mapping to path ranks also differ. It does not isolate supervision granularity or reproduce a published learned-column method. We report reservoir generation and teacher/training costs separately. Adding measured reservoir-generation time to cached evaluation time gives an attributed uncached cost, not an observed single end-to-end execution.

One master versus restarts. On one input-selected representative from each of the 16 native test networks, we reuse the identical input-only reservoir and common anchor from the preceding experiment. One 60-second restricted master is compared with three successive 20-second masters using seeds 0, 1 and 2 and carrying the best validated incumbent. Model construction is charged to each phase. Common input/reservoir preparation is reported separately. Both methods may stop after reaching the declared restricted gap tolerance; the experiment does not force otherwise unnecessary optimization. These bounds apply to the common restricted pool and are never used as global certificates.

Demand, capacity and deadline perturbations. Six prespecified regimes are generated for one representative from each held-out source network: $0.5, 1, 2, 3$ times the original demand count, capacity multiplied by 0.75, and delivery-window width multiplied by 0.75. The half-count regime samples without replacement; the larger counts retain all original demands and add bootstrap copies. The $1\times$ regime is an exact input copy. Deadline reductions are clipped to the shortest permitted route's travel time, with the clipping count reported. The generator preserves route permissions and records demand lineage. Independently checked feasible witnesses validate the generated inputs but are never supplied to the algorithms. Analytic search, complete warm MIP and the three seed-11 neural models receive 60 seconds and eight threads, yielding 480 scheduled outcomes. These are controlled perturbations of existing data, not additional real operating dates or independent networks.

An independent regeneration reproduces all 102 input manifests and files, including the six training-only calibration cases, and independently checks the shortest-route feasibility thresholds. Of the 2,300 test demands in the deadline regime, 1,480 (64.35%) require clipping. Thus this regime is a feasibility-preserving attempt to tighten windows, not a uniform 25% reduction for every order. The family counts are 77/300 for critical-times, 869/1,200 for designated-paths and 534/800 for hub-and-spoke. This input audit does not replace replay of the optimization outputs.

H.1 Fixed-pool master restarts

We isolate solver restarts from candidate generation on one unscaled representative of each of the 16 held-out native network groups. Both arms read the same SHA-verified input-only reservoir and full-charge warm start. The budget is 60 seconds of master construction, optimization and validation: one solve with seed 0 versus up to three 20-second solves with seeds 0/1/2, carrying the best validated incumbent. Runs stop when the declared solver gap tolerance is reached. All bounds remain restricted-pool diagnostics.

On the same input-only candidate pools, three 20-second master restarts instead of one 60-second master change group-mean cost by 0.0083% savings [0.0000, 0.0228], with 14 ties and 2 improvements among 16 paired networks. This small effect does not establish a substantial benefit from restarts. The interval is an unadjusted 95% group bootstrap conditional on the fixed solver seeds; all 32 attempts are feasible. All 1,284 recorded improving incumbents, as well as the 32 final solutions, pass independent raw-CSV replay with exact-decimal costs. The primary comparison admits only incumbents available by 60 seconds on the master clock; late improvements are excluded. 5 single-solve and 7 restart runs exceed the nominal allowance in actual measured time. Only 7 restart runs execute all three master solves; other restart runs terminate at solver tolerance earlier.

Master schedule	Feasible at 60 s	Saving (%)	Median master s	Tolerance reached
One 60-s master	16/16	–	17.09	11/16
Three 20-s restarts	16/16	0.0083	16.08	9/16

Table 9: Fixed-reservoir E4 comparison. Savings are paired against the single master at exactly 60 master-clock seconds. Median master time is actual measured time, including permitted early termination and observed solver overruns. Tolerance reached reports final solver termination, not a global optimum.

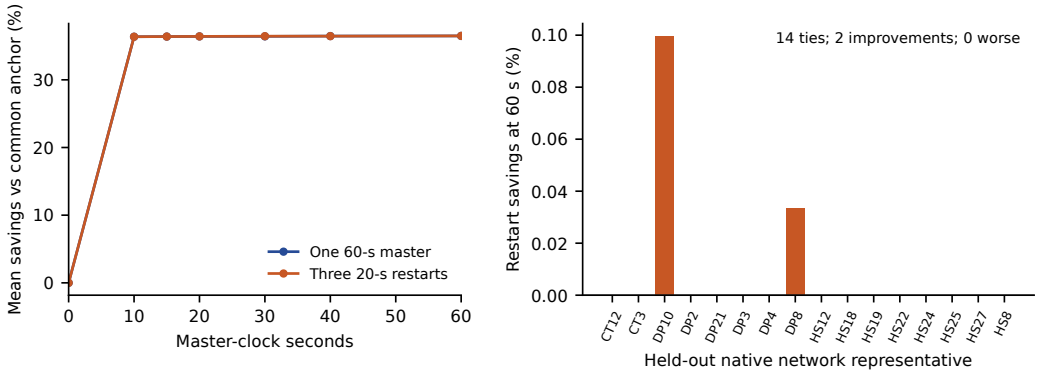


Figure 4: Left: equal-network mean savings from the input-only common anchor at fixed master-clock checkpoints. Right: paired restart savings at 60 seconds for every representative. CT/DP/HS denote the three native families; zero bars remain visible as labelled network positions.

H.2 A LINEAR SERVICE-SCORE CONTROL

We test whether a nonlinear encoder improves on a small learned scorer using the same structural and demand-pressure features. The control predicts an affine service logit,

$$s_c = w^\top [x_c; p_c] + b, \tag{31}$$

with 29 structural cell features, nine input-derived pressure features, and 39 parameters including the bias. It uses neither a demand-context encoder nor learned network identifiers. Its logits enter the same support normalization, field decoder, restricted masters, leave-one-out moves and service-closure search as the 22,147-parameter CellMLP. Thus the control makes the *score* affine; the shared feature construction and subsequent optimization pipeline are not linear algorithms.

This secondary experiment was designed after the primary results were observed. Linear models use only the original 348 training and 120 validation instances, with training seeds 11/22/33, balanced binary cross-entropy, AdamW, and validation-based checkpoint selection. The three original CellMLP checkpoints are retained. All seven arms—one analytic control and three checkpoints for each learned model—are freshly interleaved on the same CPU host, with a nominal 30-second allowance, eight solver threads and search seed 0. All 756 runs return feasible solutions and pass independent raw-CSV route, deadline and cost reconstruction; vehicle counts are additionally checked using exact decimal arithmetic.

The affine scorer saves 1.74% [0.79, 2.82] relative to analytic scoring, while CellMLP saves 2.64% [1.68, 3.68] (Table 10). The direct paired comparison gives CellMLP a further 0.89% [0.16, 1.81] saving relative to the affine scorer. For these comparisons, the three checkpoint objectives are first averaged within each instance; paired relative savings are then averaged within each source network and across the 16 networks with equal weight. Intervals use 20,000 paired network-bootstrap draws. They are descriptive, unadjusted 95% intervals conditional on the frozen checkpoints, and do not quantify uncertainty from retraining or correct for multiple comparisons. All seeds and all 108 instances are retained.

Median online time is 21.5 seconds for analytic scoring, 14.1 for the affine scorer and 7.0 for CellMLP; corresponding median process CPU times are 115.3, 83.8 and 34.2 seconds. The shared

Method	Feasible	Savings vs analytic (%) [95% CI]	Wall (s)
Analytic	108/108	0.00 [0.00, 0.00]	21.5
Linear	324/324	1.74 [0.79, 2.82]	14.1
CellMLP	324/324	2.64 [1.68, 3.68]	7.0

Table 10: Secondary affine-score comparison. Feasibility counts include every checkpoint–instance run; cost savings average checkpoint objectives within instances before weighting the 16 source networks equally. Wall time is the median across the individual runs, not a three-model inference ensemble.

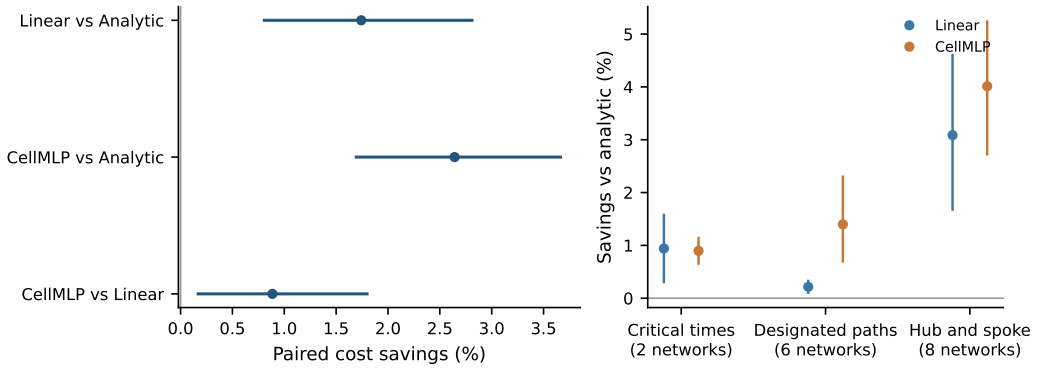


Figure 5: Left: direct paired cost savings with unadjusted 95% network-bootstrap intervals. Right: savings over analytic scoring by native family, with the number of source networks shown explicitly. Family-specific intervals, especially for critical times with only two networks, should not be interpreted as broad distributional evidence.

allowance permits early termination under the unchanged search stopping rules, so actual runtimes differ. The affine model has cheaper median inference (0.6 versus 3.9 milliseconds), but this does not give it the shorter overall solve: both learned models spend about 1.1 seconds loading, and the online algorithm also constructs candidates and solves restricted masters. Online timing starts after common module imports and includes input parsing, graph/features, model loading, inference, search and final validation; offline training is excluded. Actual maximum wall times are 30.5, 30.4 and 30.8 seconds respectively, retaining solver and validation overruns.

These results support the specified nonlinear cellwise scorer over this affine control, while showing that useful learned fields can already be obtained with 39 parameters. They do not establish a need for attention. The advantage is not uniform across families: CellMLP versus the affine scorer gives -0.05% $[-0.45, 0.35]$ on critical times, 1.18% $[0.49, 2.11]$ on designated paths and 0.90% $[-0.33, 2.58]$ on hub-and-spoke networks. All three affine models select epoch 60, the prescribed training limit; the more thoroughly optimized affine follow-up is reported in Appendix H.3. The conclusion concerns these frozen training and search recipes, not every possible affine scorer, and no global optimality gap is inferred from restricted-master bounds.

H.3 A THOROUGHLY FITTED AFFINE RECOMMENDATION CONTROL

The original affine models all selected the last permitted AdamW epoch. We therefore repeat the comparison after fitting the same 39-parameter scorer more thoroughly. The four-penalty, float64 L-BFGS-B procedure and training-only Newton refinement are specified in Appendix H. The validation-selected penalty is zero. Its standardized- and original-coordinate gradient infinity norms are 7.50×10^{-11} and 1.47×10^{-11}, below both declared 10^{-8} thresholds. All four candidates' losses, normalization and gradients are independently recomputed from the 348 training and 120 validation teachers; no test teachers enter fitting. The complete fitting procedure takes 47.88 seconds, separately from online search.

Method	Feasible	Savings vs analytic (%) [95% CI]	Wall (s)
Analytic	108/108	0.00 [0.00, 0.00]	25.9
Logistic (L-BFGS)	108/108	2.51 [1.47, 3.71]	16.5
CellMLP	324/324	2.74 [1.74, 3.83]	8.5

Table 11: Complete affine-fitting follow-up. The single deterministic affine checkpoint is selected by validation loss; CellMLP retains its three original checkpoints. Wall time is the median over individual runs, and cost comparisons average CellMLP checkpoint objectives within each instance.

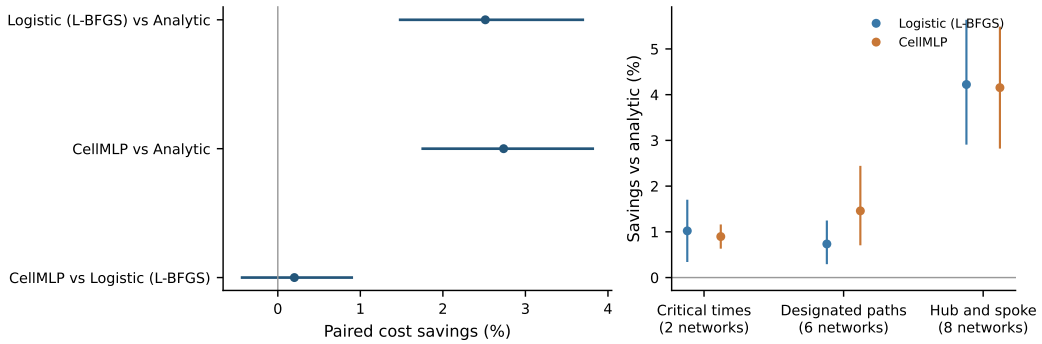


Figure 6: Thorough affine fitting strengthens the learned linear control. Direct cost intervals span zero for CellMLP versus this control; the online resource measurements remain distinct. All estimates retain the same 16 held-out source networks.

The fresh test interleaves the selected affine model, analytic search and the three original CellMLP checkpoints. All 540 scheduled runs are feasible and pass independent reconstruction. The affine scorer saves 2.51% [1.47, 3.71] over analytic recommendation, versus 2.74% [1.74, 3.83] for CellMLP. The direct CellMLP saving over the affine scorer is 0.20% [-0.45, 0.91]. More thorough fitting therefore substantially narrows the earlier affine–nonlinear cost difference; these data do not establish a clear remaining average-cost advantage for nonlinearity alone.

CellMLP nevertheless finishes sooner in median online time: 8.52 seconds versus 16.49 for the affine scorer and 25.86 for analytic search. Corresponding median process CPU times are 39.24, 78.27 and 119.15 seconds, and median pool sizes are 1,469.5, 1,786 and 2,017 paths. Affine inference itself is cheaper (0.78 versus 6.20 milliseconds), while both learned models incur approximately one second of loading. The shorter CellMLP runtime thus emerges from the full recommendation–optimization pipeline, rather than faster neural evaluation. All methods retain the original 30-second allowance and stopping rules.

The results support learning service preferences and the practical efficiency of the neural pipeline, while identifying a strong inexpensive linear representation. There is one deterministic affine fit, not replicated affine training seeds. Intervals are conditional on that selected model and the frozen CellMLP checkpoints. This follow-up was specified after observing the primary study and the original affine experiment; it adds no independent test networks.

H.4 Service-score and column selection on a common reservoir

This secondary experiment was specified after the primary results. It compares two supervision and selection representations on an identical input-only candidate reservoir; it does not compare their ability to generate new columns and is not a reproduction of a named published learned-column method. Every instance receives full-charge and variable-only paths plus analytic and fixed-random fields at three supports, with jitter and legal whole-path shifts. No learned or teacher path is injected into that reservoir.

Paired comparison	Savings (%) [95% CI]	Networks	Instances
Logistic (L-BFGS) vs Analytic	2.51 [1.47, 3.71]	16/16	108/108
CellMLP vs Analytic	2.74 [1.74, 3.83]	16/16	108/108
CellMLP vs Logistic (L-BFGS)	0.20 [-0.45, 0.91]	16/16	108/108

Table 12: Direct paired savings for the affine-fitting follow-up, with 20,000 equally weighted network-bootstrap draws and unadjusted 95% intervals. Each comparison contains all 108 instances and 16 networks.

One original-cost restricted-master teacher is solved for each of the 348 training and 120 validation instances, with a 30-second allowance and eight threads. Its selected column for each order supplies column labels; the union of selected services supplies service labels. Both models use two 128-wide ReLU layers and the same native input feature extractor. The column model receives 112 route/order features and has 31,105 parameters; the shared service model receives 66 cell/pooled-context features and has 25,217. Thus inputs and supervision granularity intentionally differ, and parameter counts are not exactly matched. Both use balanced BCE, seeds 11/22/33 and validation-based epoch selection. Alternative useful columns can be negative under the single-teacher column target, which is a limitation of this controlled classifier baseline.

Every selector keeps exactly $\min(8, |\mathcal{P}_k|)$ columns for order k, including a shared full-charge anchor that consumes one slot. The service model ranks paths by the sum of its support-normalized service charges plus original commodity-variable cost; the column model ranks its column logits. Analytic and three fixed-random selectors use the same slot count. A full-reservoir master is also run. All 11 arms use the same true-cost master, eight threads, solver seed 0, and a nominal 30-second master allowance, without LOO, closure or further pool expansion. All 1,188 runs over 108 instances and 16 original networks are feasible; no solver-incumbent fallback or preference for the anchor over a returned incumbent occurs. Every final solution is independently reconstructed from raw CSV, including exact decimal vehicle, fixed and commodity-variable costs. Cache identity, common anchors, exact slot counts, solution membership, six checkpoints and both teacher-label views also pass independent checks.

Column MLP saves 2.88 [1.67, 4.34]% relative to analytic selection; Service MLP saves 4.61 [3.61, 5.69]%. The prespecified column-versus-service saving is -1.83 [-2.86, -0.80]%, meaning the column selector costs 1.83 [0.80, 2.86]% more when the service selector is the denominator. Both compressed selectors sacrifice quality relative to the full reservoir: their cost premiums are 6.59 [5.26, 7.96]% and 4.71 [3.35, 6.57]%, respectively. Positive savings and positive cost premiums use different directions and are labeled explicitly in the tables.

The three seed costs are averaged within each instance before computing relative differences, then instances are averaged within each original network and networks are weighted equally. All confidence intervals use 20,000 paired network-bootstrap draws; they are unadjusted 95% intervals conditional on the frozen checkpoints and do not estimate retraining uncertainty. Every comparison retains all 108 instances and 16 networks. The family-specific column-versus-service savings are critical times 0.22 [0.10, 0.33]%; designated paths -4.14 [-4.78, -3.55]%; hub-and-spoke -0.62 [-1.48, 0.41]%. The families contain only 2, 6 and 8 networks, respectively; in particular, the critical-times interval is not broad distributional evidence.

The median retained column count is 1134.5 for every selector versus 2507.5 for the full reservoir. Cached-unit median times for analytic, random, column, service and full-reservoir arms are 1.18, 1.80, 2.30, 2.48, 23.93 seconds; corresponding attributed uncached medians are 1.57, 2.20, 2.61, 2.83, 24.29. Common input parsing, graph/features and reservoir generation take a median 0.367 seconds per instance. Scoring includes model loading, with medians 1.081 and 1.089 seconds for the learned selectors. The full-reservoir maximum observed wall time is 41.74 seconds; solver/persistence overruns are retained. The control is equal master allowance, not equal total wall time. Pre-entry common imports are outside the unit clock; per-process CPU/RSS include them.

The 468 teacher solves consume 9166.1 summed seconds, and 237/468 reach the solver's declared restricted-master gap tolerance. These are not global optimality certificates. Train/validation reser-

Method	Feasible	Cost premium vs full (%) [95% CI]	Cached (s)	Attributed (s)
Analytic	108/108	9.90 [7.62, 12.77]	1.18	1.57
Random	324/324	15.01 [11.33, 18.95]	1.80	2.20
Column MLP	324/324	6.59 [5.26, 7.96]	2.30	2.61
Service MLP	324/324	4.71 [3.35, 6.57]	2.48	2.83
Full reservoir	108/108	0.00 [0.00, 0.00]	23.93	24.29

Table 13: Common-reservoir selection. Positive cost premium means greater original cost than the full-reservoir solution, not an optimality gap. Feasibility counts include every scheduled seed–instance run. Times are medians across individual runs. "Attributed" adds measured common preparation and removes duplicate cache reconstruction; it is not a separately timed uncached execution.

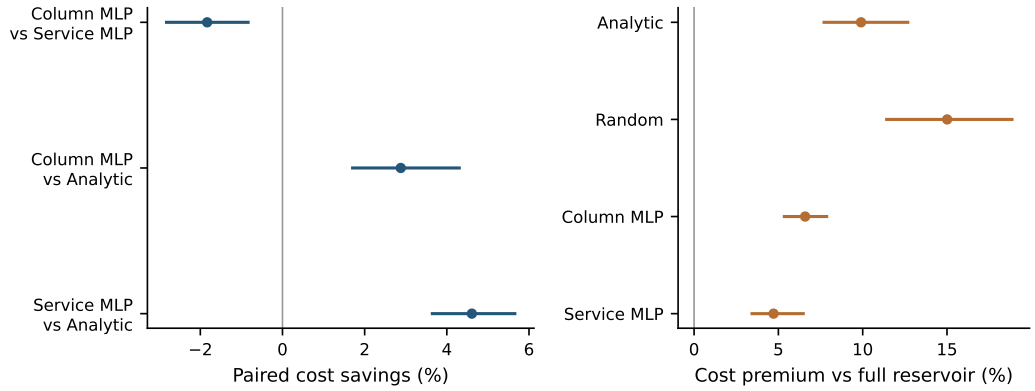


Figure 7: Paired common-reservoir cost comparisons with unadjusted 95% network-bootstrap intervals. Left: higher savings favor the first named selector. Right: higher premiums indicate worse cost than retaining the full reservoir. These results support this service-supervision recipe over the specified column classifier, while showing the quality lost by either compressed pool.

voir generation consumes another 218.9 summed seconds. The six model timers total 191.4 seconds; these internal timers start after PyTorch import and setup. Their parallel training campaign lasts 49.1 seconds. Test-only reservoir generation totals 43.8 seconds across 108 instances and is online preparation, not learning cost. These stage sums must not be presented as a cross-host or concurrent-stage makespan.

The comparison demonstrates a favorable service-level supervision/selection result within this fixed pool and training recipe. It does not establish attention necessity, superiority to every learned column model, a benefit in generating candidates, or dominance over the full-reservoir optimizer.

H.5 Frozen-model response to demand, capacity and deadline changes

The six input regimes and their construction are specified in Appendix H. All 480 scheduled runs return feasible solutions and pass a separate reconstruction of paths, deadlines, capacities and original costs from raw CSV, including exact decimal vehicle counts. There are 96 derived test instances but only 16 independent source networks. The three neural methods each use their frozen seed-11 checkpoint without adaptation. These post-primary comparisons do not measure retraining variability or performance on new operating dates.

All three learned methods improve on analytic search under the four count regimes and the capacity reduction, with positive unadjusted network-bootstrap intervals in these comparisons. For example, at three times the original commodity count, CellMLP, DeepSets and latent attention save 1.84 [0.99, 2.83]%, 1.79 [0.98, 2.70]% and 1.23 [0.57, 1.99]%, respectively. Their advantage is not uniform across perturbations: after deadline tightening, savings are only 0.21 [-0.09, 0.61]%, 0.15 [-0.13, 0.51]% and 0.17 [-0.12, 0.53]%, with all three intervals crossing zero.

Method relative to baseline	Saving (%) [95% CI]	Networks / instances
Column MLP vs Service MLP	-1.83 [-2.86, -0.80]	16 / 108
Column MLP vs Analytic	2.88 [1.67, 4.34]	16 / 108
Column MLP vs Random	6.99 [4.69, 9.37]	16 / 108
Column MLP vs Full reservoir	-6.59 [-7.96, -5.26]	16 / 108
Service MLP vs Analytic	4.61 [3.61, 5.69]	16 / 108
Service MLP vs Random	8.66 [6.77, 10.57]	16 / 108
Service MLP vs Full reservoir	-4.71 [-6.57, -3.35]	16 / 108
Analytic vs Full reservoir	-9.90 [-12.77, -7.62]	16 / 108
Random vs Full reservoir	-15.01 [-18.95, -11.33]	16 / 108

Table 14: All prespecified common-reservoir paired cost contrasts; positive values favor the first named method. Each includes 108 instances and 16 source networks.

Method	Columns med.	CPU med. (s)	RSS max. (MiB)	Scoring med. (s)	Wall max. (s)
Analytic	1134.5	6.52	250.0	0.008	30.23
Random	1134.5	10.52	286.3	0.004	30.25
Column MLP	1134.5	6.68	468.7	1.081	31.57
Service MLP	1134.5	7.75	481.1	1.089	31.44
Full reservoir	2507.5	124.54	333.9	0.003	41.74

Table 15: Common-reservoir resource accounting across individual runs. Scoring includes model loading. CPU/RSS are per-process measurements; the full observed wall overruns remain visible.

Complete warm-start MIP achieves lower mean cost than every neural method in every regime. In the three-times-count regime it saves 4.06 [2.37, 5.98]% against analytic search; CellMLP costs 2.40% more with full warm MIP as the denominator. The shared 60-second nominal budget does not imply equal stopping times. All methods have median times close to 60 seconds at two and three times demand, so those regimes do not demonstrate a substantial median runtime advantage for learning. Conversely, on the half-count instances full warm MIP has a 1.19-second median versus 2.45–2.56 seconds for the learned methods. In the clipped-deadline regime it has a 0.15-second median versus 1.51–1.58 seconds. Model loading and inference remain charged to the neural methods.

Deadline clipping affects 1,480/2,300 demands (64.35%); the result concerns this feasibility-preserving perturbation, not a uniform 25% reduction in every window. The small observed learned improvement in that regime does not establish a general explanation for tight-window behavior. All paired comparisons use equal source-network weights and 20,000 bootstrap draws, with unadjusted 95% intervals conditional on the fixed checkpoints. Regimes are repeated observations on the same networks, not additional independent replications.

H.6 Complementary neural recommendations at matched portfolio size

The heterogeneous portfolio uses latent seeds 11/22, DeepSets 11 and CellMLP 11. Four controls repeat each constituent checkpoint across four chains; a fifth uses four analytic chains. Chain seeds, support scales, the four-chain count and union optimization are identical across arms. Every chain receives 15 seconds and eight solver threads, within a 60-second parent budget that includes startup and the final union. The complete 648-unit matrix covers 108 instances from the same 16 source networks. All 648 final solutions pass independent raw-input reconstruction; all 2,592 child receipts and exported pool identities pass the separate accounting audit.

Heterogeneous neural recommendation saves 1.85% [1.18,2.55] relative to four analytic chains. It also improves on repeating each of its constituents:0.49% [0.25,0.79] against latent 11,0.36% [0.12,0.66] against latent 22,0.34% [0.16,0.54] against DeepSets 11, and 0.52% [0.19,0.89] against CellMLP 11. All contrasts retain 108 paired instances and 16 networks, with 20,000 equal-network bootstrap draws and unadjusted 95% intervals. Median online runtime is 22.75 seconds for the heterogeneous neural portfolio versus 48.43 for analytic chains, and median total CPU time is 255.85

Half commodity count

Method	Feasible	Savings (%) [95% CI]	Wall (s)	Process (s)	CPU (s)	Max RSS (GiB)
Analytic	16/16	0.00 [0.00, 0.00]	1.9	2.2	8.8	0.21
CellMLP	16/16	1.77 [0.85, 2.79]	2.6	3.1	7.5	0.60
DeepSets	16/16	1.68 [0.83, 2.64]	2.6	3.1	7.3	0.60
Latent attention	16/16	1.52 [0.73, 2.42]	2.4	3.0	6.9	0.62
Full MIP warm	16/16	3.49 [2.30, 4.79]	1.2	1.5	6.4	0.24

Original commodity count

Method	Feasible	Savings (%) [95% CI]	Wall (s)	Process (s)	CPU (s)	Max RSS (GiB)
Analytic	16/16	0.00 [0.00, 0.00]	26.4	26.7	169.3	0.34
CellMLP	16/16	2.86 [1.62, 4.08]	5.2	5.8	19.7	0.73
DeepSets	16/16	2.97 [1.78, 4.19]	8.2	8.7	45.2	0.71
Latent attention	16/16	2.32 [1.43, 3.31]	12.5	13.0	75.8	0.75
Full MIP warm	16/16	5.62 [4.05, 7.25]	25.2	25.5	141.8	0.73

Twice commodity count

Method	Feasible	Savings (%) [95% CI]	Wall (s)	Process (s)	CPU (s)	Max RSS (GiB)
Analytic	16/16	0.00 [0.00, 0.00]	60.1	60.3	321.3	0.37
CellMLP	16/16	2.70 [1.40, 4.33]	60.1	60.6	341.9	0.80
DeepSets	16/16	2.62 [1.34, 4.28]	60.1	60.6	336.8	0.80
Latent attention	16/16	2.50 [1.27, 4.09]	60.1	60.6	314.8	0.83
Full MIP warm	16/16	5.26 [3.47, 7.21]	60.2	60.5	296.5	1.02

Table 16: Count perturbations: all scheduled outcomes and resource measurements. Savings use analytic search as denominator. Wall, process-wall and CPU entries are medians; RSS is the maximum. Process wall includes interpreter and common imports excluded from the inner unit clock.

Three times commodity count

Method	Feasible	Savings (%) [95% CI]	Wall (s)	Process (s)	CPU (s)	Max RSS (GiB)
Analytic	16/16	0.00 [0.00, 0.00]	60.1	60.4	321.4	0.46
CellMLP	16/16	1.84 [0.99, 2.83]	60.1	60.6	295.5	0.85
DeepSets	16/16	1.79 [0.98, 2.70]	60.1	60.6	301.2	0.85
Latent attention	16/16	1.23 [0.57, 1.99]	60.1	60.6	298.4	0.84
Full MIP warm	16/16	4.06 [2.37, 5.98]	60.4	60.7	292.7	1.31

Capacity multiplied by 0.75

Method	Feasible	Savings (%) [95% CI]	Wall (s)	Process (s)	CPU (s)	Max RSS (GiB)
Analytic	16/16	0.00 [0.00, 0.00]	40.5	40.8	256.3	0.33
CellMLP	16/16	1.96 [1.13, 2.87]	19.4	19.9	129.8	0.71
DeepSets	16/16	2.05 [1.14, 3.02]	26.3	26.9	182.5	0.76
Latent attention	16/16	1.62 [0.80, 2.56]	27.6	28.2	157.2	0.73
Full MIP warm	16/16	4.48 [3.32, 5.66]	48.2	48.5	257.6	0.65

Window multiplied by 0.75, with feasibility clipping

Method	Feasible	Savings (%) [95% CI]	Wall (s)	Process (s)	CPU (s)	Max RSS (GiB)
Analytic	16/16	0.00 [0.00, 0.00]	0.4	0.7	0.5	0.09
CellMLP	16/16	0.21 [-0.09, 0.61]	1.5	2.0	1.6	0.50
DeepSets	16/16	0.15 [-0.13, 0.51]	1.6	2.1	1.6	0.50
Latent attention	16/16	0.17 [-0.12, 0.53]	1.5	2.0	1.6	0.51
Full MIP warm	16/16	0.55 [0.13, 1.12]	0.1	0.4	0.3	0.10

Table 17: Higher demand, capacity and deadline perturbations, with the same accounting as Table 16. All observed budget overruns are retained in the per-instance records; the maximum inner wall time is 79.68 seconds for DeepSets on a three-times-count instance.

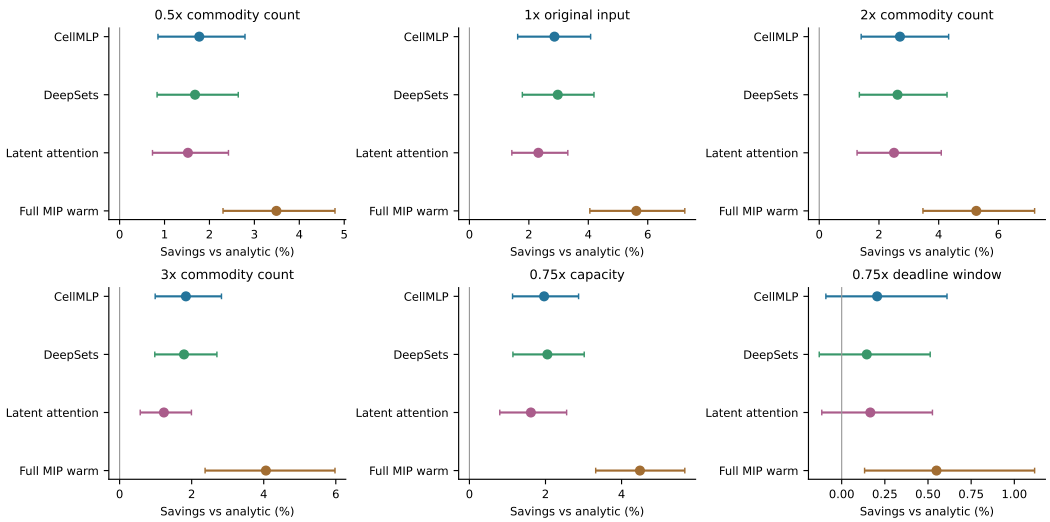


Figure 8: Frozen-model perturbation results. Positive savings favor the named method over analytic search. Error bars are unadjusted 95% paired network-bootstrap intervals. Full warm MIP remains the strongest mean-cost control; the deadline regime gives no clear learned gain over analytic search.

Control	Portfolio saving (%) [95% CI]	Networks	Instances
Latent11 repeated	0.49 [0.25, 0.79]	16	108
Latent22 repeated	0.36 [0.12, 0.66]	16	108
DeepSets11 repeated	0.34 [0.16, 0.54]	16	108
CellMLP11 repeated	0.52 [0.19, 0.89]	16	108
Analytic chains	1.85 [1.18, 2.55]	16	108

Table 18: Heterogeneous neural portfolio versus each matched four-chain control. Positive savings favor the heterogeneous portfolio; the named control supplies the denominator.

versus 420.28 seconds. This supports complementary recommendations from the frozen heterogeneous portfolio beyond simply allocating four restarts to one checkpoint. It does not identify attention as the unique source of that complementarity.

The execution was briefly interrupted after 560 aggregated outcomes and resumed on the same host with the same configuration, retaining completed units. The final matrix includes every scheduled arm and instance. Per-unit timers are preserved across the interruption; the resumed scheduler's duration is not presented as the total original campaign time. These remain secondary results specified after the primary study, rather than a new independent test set.

H.7 Quality at fixed elapsed time

The revised short-budget experiment compares seven methods on all 16 input-selected, 1,600-demand stress cases. Each run has a 120-second allowance and eight solver threads. All 112 runs return feasible solutions, and an independent reconstruction verifies all 9,566 recorded improving incumbents. Every method has a validated solution on all 16 cases at each reported checkpoint. The clock includes process launch, imports, input preparation, model loading, inference, search and solution validation. Only solutions already validated and saved by a checkpoint contribute to its value.

At 10 seconds, CellMLP, DeepSets and latent attention save 4.89% [2.97, 6.69], 4.23% [2.30, 6.07] and 3.24% [1.18, 5.18] relative to full warm MIP. Relative to analytic recommendation, their savings are 2.32% [0.22, 4.91], 1.66% [−0.13, 4.13] and 0.71% [−0.52, 2.43]. CellMLP retains positive sav-

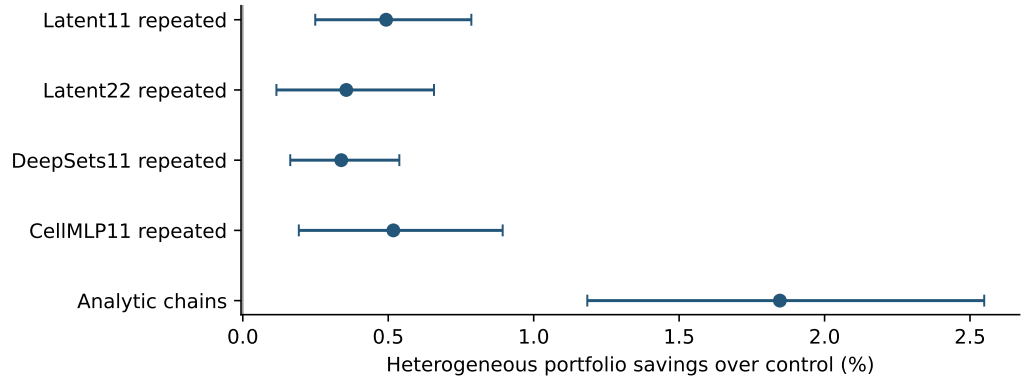


Figure 9: The complete portfolio comparison supports both learned recommendation over analytic chains and complementarity beyond repeated individual neural constituents. Intervals are conditional on the frozen models and resample the original 16 networks.

Portfolio	Feasible	Wall median (s)	CPU median (s)	Union saving (%)
Latent11 repeated	108/108	22.53	270.62	1.046
Latent22 repeated	108/108	18.87	221.99	1.022
DeepSets11 repeated	108/108	23.95	275.24	0.878
CellMLP11 repeated	108/108	20.16	233.87	1.011
Heterogeneous neural	108/108	22.75	255.85	1.258
Analytic chains	108/108	48.43	420.28	1.882

Table 19: All portfolio runs and their actual resource use. Union saving compares the final solution with the best child, averaged equally across source networks. CPU sums coordinator and child-process usage; identical nominal limits do not imply identical realized CPU time.

ings over analytic search at every reported checkpoint, reaching 1.13% [0.27, 2.12] at 120 seconds. These measurements directly support the usefulness of learned recommendations when high-quality solutions are needed early.

The longer observation times show the trade-off. At 120 seconds, full cold and warm MIP save 1.87% [0.50, 3.31] and 1.83% [0.46, 3.33] over analytic search. Relative to warm MIP, learned savings are -0.75% [-1.65, 0.12], -1.30% [-2.56, -0.14] and -1.56% [-2.95, -0.29]. Thus the early neural advantage does not imply lower final cost at every horizon. The validation-selected analytic schedule differs little from the default on these tests: its 120-second saving is 0.07% [-0.04, 0.21]. All tested schedules and the validation selection are retained.

The target measurements distinguish cheap initial feasibility from subsequent quality. Analytic recommendation reaches the common anchor and the 90%-of-anchor target sooner in median time than the neural methods; neural startup includes checkpoint loading. Every method reaches the first three target levels on all 16 cases. For the 80% target, hits are 13/16 for default analytic, the three neural methods and cold MIP, versus 12/16 for tuned analytic and warm MIP. These input-only targets avoid defining success from a hindsight best result.

No execution failures occur. Median measured termination times are 120.33–120.43 seconds; the largest overruns are 135.38 seconds for default analytic and 127.69 for tuned analytic. Other maxima are below 120.59 seconds. Late improvements are retained in raw trajectories but excluded from the 120-second comparison. All intervals use 20,000 equally weighted source-network bootstrap draws and the frozen seed-11 encoders. The same original 16 stress parents supply every comparison. The short-budget revision was made for resource prioritization; the interrupted hour-budget campaign is preserved separately and contributes no final estimate here. The 30-second observation of a 120-second schedule is distinct from a separately budgeted 30-second primary run.

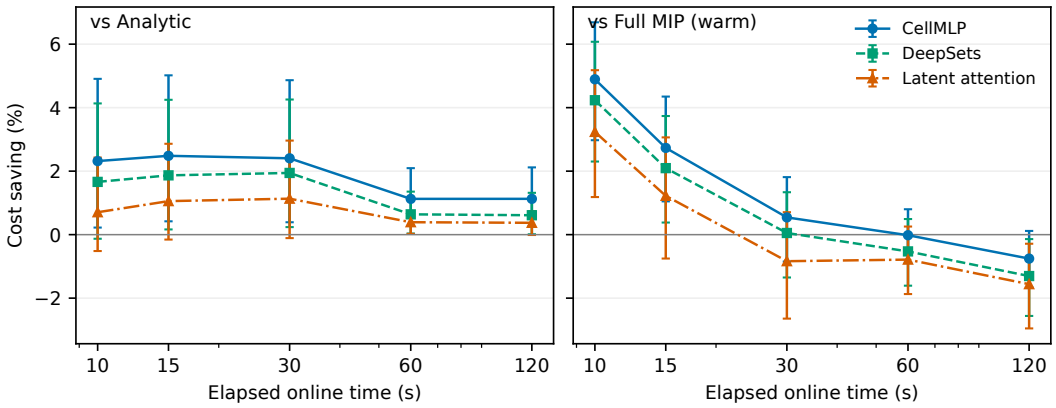


Figure 10: Cost at fixed observation times in the complete 112-run short-budget study. Positive savings favor the learned method over the indicated control. All points have 16 paired feasible networks; bars are unadjusted 95% network-bootstrap intervals. Lines join observed checkpoints and do not interpolate solver events.

Method	10 s	15 s	30 s
Analytic	—	—	—
Tuned analytic	-0.09 [-0.26, 0.00]	-0.02 [-0.06, 0.01]	-0.00 [-0.00, 0.00]
CellMLP	2.32 [0.22, 4.91]	2.48 [0.42, 5.02]	2.40 [0.39, 4.86]
DeepSets	1.66 [-0.13, 4.13]	1.87 [0.16, 4.25]	1.94 [0.24, 4.26]
Latent attention	0.71 [-0.52, 2.43]	1.06 [-0.16, 2.86]	1.13 [-0.11, 2.96]
Full MIP (cold)	-3.05 [-5.54, -0.36]	-0.24 [-2.61, 2.35]	2.03 [-0.08, 4.50]
Full MIP (warm)	-2.80 [-5.36, -0.06]	-0.33 [-2.73, 2.34]	1.84 [-0.23, 4.31]

Method	60 s	120 s
Analytic	—	—
Tuned analytic	0.10 [-0.01, 0.22]	0.07 [-0.04, 0.21]
CellMLP	1.13 [0.28, 2.09]	1.13 [0.27, 2.12]
DeepSets	0.64 [0.04, 1.35]	0.61 [0.01, 1.32]
Latent attention	0.39 [0.02, 0.74]	0.37 [-0.01, 0.74]
Full MIP (cold)	1.41 [0.12, 2.74]	1.87 [0.50, 3.31]
Full MIP (warm)	1.12 [-0.04, 2.33]	1.83 [0.46, 3.33]

Table 20: Group-mean cost savings (%) relative to analytic recommendation at the same elapsed time, with unadjusted 95% intervals. All seven methods have 16/16 feasible solutions at every displayed checkpoint; the analytic row is the common reference.

Method	$1.00A$	$0.95A$	$0.90A$	$0.80A$
Analytic	0.90 (16/16)	1.38 (16/16)	1.40 (16/16)	1.41 (13/16)
Tuned analytic	0.91 (16/16)	1.41 (16/16)	1.43 (16/16)	1.43 (12/16)
CellMLP	2.04 (16/16)	2.55 (16/16)	2.57 (16/16)	2.58 (13/16)
DeepSets	2.04 (16/16)	2.51 (16/16)	2.55 (16/16)	2.63 (13/16)
Latent attention	2.07 (16/16)	2.53 (16/16)	2.62 (16/16)	2.64 (13/16)
Full MIP (cold)	4.07 (16/16)	4.37 (16/16)	4.97 (16/16)	6.32 (13/16)
Full MIP (warm)	0.82 (16/16)	4.41 (16/16)	5.49 (16/16)	5.70 (12/16)

Table 21: Time to input-only target: median observed seconds among successful attempts, followed by hits/16 in parentheses. A is the common full-charge input-only solution cost for each instance. Missing targets remain in the denominator; conditional medians for $0.80A$ are not censored-population medians.

I MEDIUM-MODEL TRAINING AND FIXED-WEIGHT CASE DIAGNOSTICS

The selected test_07 case at each scale illustrates the nine-case protocol. Each FULL row, its anytime trajectory and its pool snapshots come from the same canonical execution; no additional fit, teacher solve or diagnostic optimization is performed. All finite final solutions and displayed saved-pool solutions are replayed under the original constraints and monetary objective.

I.1 INVERSE-SUPERVISED FIELDS AND MEDIUM TRAINING

The nine-case study uses a separately trained inverse-supervised field generator within the candidate-pool framework. Its four heads produce four field proposals for one search chain; they are distinct from the independent chains used by Portfolio. The generator is supervised by inverse-target vectors $\beta^{(h)}$, $h = 1, \ldots, 4$, and cached cuts, rather than the binary activation targets in Section 4.1. Its field charges enter the shipment oracle in Equation (8); paths are still selected by the original-cost integer master. Thus the common search interface does not imply identical supervision or field construction across the experiments.

The model is trained on Medium scenarios 01–05 and validated on scenario 06, with normalization fitted on the training scenarios. One selected checkpoint is reused for Small, Medium and Large scenarios 07–09. The recorded configuration uses hidden width 64 and Adam with learning rate 10^{-3} for 160 epochs. The Reference-1602 activation checkpoints and the public-benchmark classifiers have separate training protocols, given in Appendices B.2 and E.

The training loss is balanced inverse-target MSE plus 0.1 times a cached-cut surrogate. Within each head, the MSE balances informative ($\beta < 0.99$) and neutral targets, retains cell weights within each stratum, and then averages over heads. The surrogate averages positive normalized violations of cached cuts. Each training point averages losses computed before successive optimizer updates over the five scenarios; validation is evaluated after that epoch. Thus the two curves are not evaluations of the same fixed parameters on two datasets at each step. Neither loss is a transportation cost.

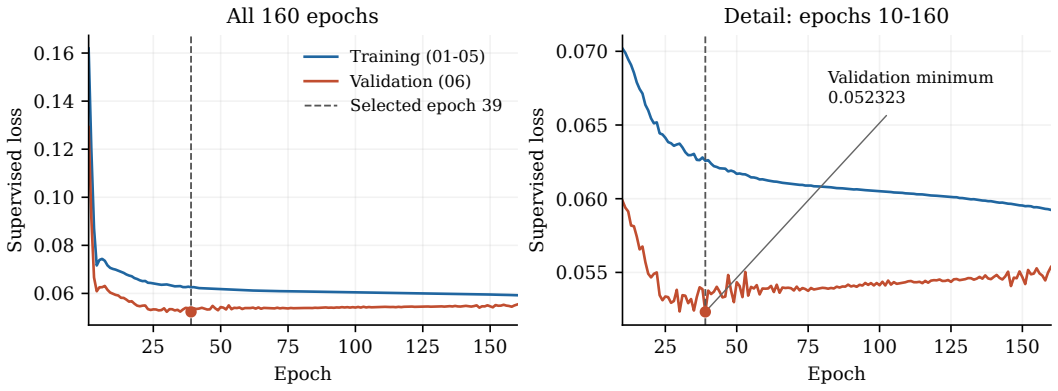


Figure 11: Recorded Medium learned-IFC losses over all 160 epochs. The right panel enlarges epochs 10–160 without smoothing or refitting. The dashed line marks epoch 39, selected by minimum validation loss on scenario 06. This one checkpoint is reused on Small, Medium, and Large; no separate scale-specific fits are implied.

The selected epoch is 39: validation balanced MSE is 0.050900, the cached-cut term is 0.014230, and total loss is 0.052323. At epoch 160 the total validation loss is 0.055368, so the final epoch is not the deployed checkpoint. The history records only aggregate training and validation losses, not per-epoch head quality or online routing cost. The unchanged checkpoint hash links this curve to the frozen nine-case campaign. The current fit excludes the displayed test cases from weights, normalization, labels, and selection, while the historical development exposure described in Section 5.4 still limits generalization claims.

I.2 Component controls and accepted-incumbent anytime behavior

Table 22 compares each control with the same scale-specific FULL run. Figure 12 shows the accepted-incumbent trajectory from that run, including initialization time. These diagnostics describe the saved executions; they do not add independent repetitions or reset the clock between stages.

Table 22: Matched test_07 ablations with frozen Medium weights. Cost is in thousands; saving is $100(U_{variant} - U_{FULL})/U_{variant}$.

Variant	Small cost	Saving	Medium cost	Saving	Large cost	Saving
FULL	1,607.50	0.00	2,878.84	0.00	5,073.07	0.00
Without LOO	1,697.58	5.31	2,994.82	3.87	5,192.29	2.30
Without N1	2,010.83	20.06	3,503.64	17.83	5,735.37	11.55
Without feedback	2,525.18	36.34	4,204.14	31.52	6,669.31	23.93
Without nested RMP	1,826.29	11.98	3,141.43	8.36	5,286.49	4.04
Analytic fields	1,640.46	2.01	2,917.05	1.31	5,150.28	1.50
Random fields	1,640.00	1.98	2,896.09	0.60	5,173.56	1.94

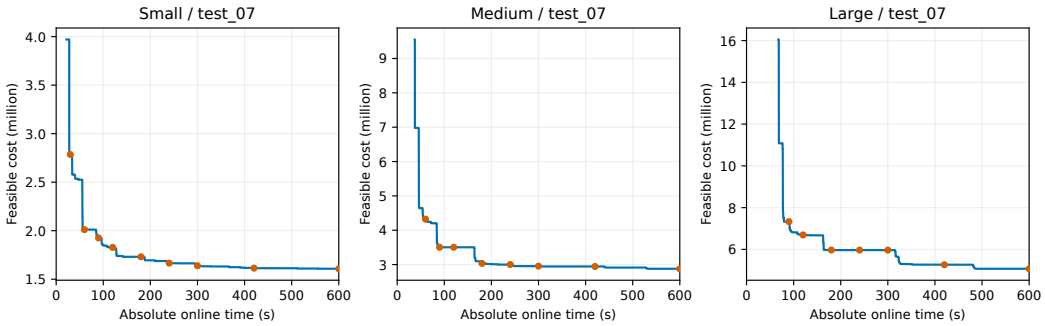


Figure 12: One continuous FULL run per scale. Lines show recorded accepted incumbents; dots mark the fixed absolute checkpoints. No feasible solution before initialization is plotted. These are different order sets, not replicated measurements or confidence intervals.

I.3 Pool growth, selection and identifiable error components

Let $z(P_j)$ be the integer optimum over the exact saved candidate pool, $z^\star$ the global integer optimum, and U_j its accepted feasible incumbent. With a verified global lower endpoint L,

$$U_j - L = [U_j - z(P_j)] + [z(P_j) - z^\star] + [z^\star - L]. \tag{32}$$

The terms are selection residual, pool restriction loss and reference slack. This identity alone cannot determine their individual values; pool restriction is not an isolated learning error. For each exact pool hash we collect the best saved feasible cost V_j, and a retained in-budget restricted-pool bound b_j when its receipt survives. No bound is transferred between different pools. The strongest such same-pool evidence may be later in the same run, so the intervals are retrospective diagnostics, not certificates available at the earlier time. Let H be the best verified feasible cost among the complete 13-setting registry for this instance, including the Portfolio outcomes in Table 32, and $\beta_j = \max(L, b_j)$, or L if no such b_j survives. Then

$$\max(0, U_j - V_j) \le U_j - z(P_j) \le \max(0, U_j - \beta_j), \tag{33}$$

$$\max(0, \beta_j - H) \le z(P_j) - z^\star \le V_j - L, \tag{34}$$

$$0 \le z^\star - L \le H - L. \tag{35}$$

The intervals are coupled, conditional on numerical receipts, and are not additive point estimates or statistical confidence intervals. Missing global references do not prevent reporting the selection lower bound $\max(0, U_j - V_j)$; unavailable upper endpoints are marked "–" below. No bound is

transferred from a smaller pool to an expanded pool. The following diagnostics use Small/test_07, Medium/test_07 and Large/test_07, using the same canonical FULL executions as the main comparison. They use only saved stage, solution and validation records; no additional fit, replay or optimization is performed for this extension.

Pool growth and selected solutions. Table 23 reports the initial selection, the fourth learned-field selection, both subsequent restricted-master boundaries, LOO, the first N1 selection and final polish. Intermediate learned-head selections and generation-only snapshots remain in the saved records. Generation-only snapshots retain the previous incumbent and are not separate solves. "Initial" denotes M0 for Small and mode recovery for Medium and Large. REUSE denotes a skipped solve on the same previously proved pool; OPT is an in-budget restricted-pool optimality receipt, TL is time limit and INT is callback interruption. None establishes the global integer optimum.

Table 23: Saved pool and selection boundaries for test_07 at each scale. Costs are in thousands of objective units.

Case	Boundary	Pool size	Status	Cost	Accepted (s)
Small / 07	Initial	800	OPT	3,970.88	21.76
	M4	2,303	OPT	2,525.18	47.47
	RMP A	2,303	REUSE	2,525.18	47.47
	LOO	2,843	OPT	2,010.83	57.54
	First N1	48,315	INT	1,924.82	88.33
	RMP B	48,315	INT	1,828.72	110.29
	Final polish	48,315	INT	1,607.50	573.56
Medium / 07	Initial	7,813	OPT	6,978.23	37.56
	M4	10,967	OPT	4,204.14	72.46
	RMP A	10,967	REUSE	4,204.14	72.46
	LOO	12,057	TL	3,503.65	89.18
	First N1	99,501	INT	3,096.94	167.18
	RMP B	99,501	INT	3,015.80	190.16
	Final polish	99,501	INT	2,878.84	530.99
Large / 07	Initial	15,631	OPT	11,080.88	68.85
	M4	21,768	INT	6,677.39	130.15
	RMP A	21,768	INT	6,673.05	140.75
	LOO	23,848	INT	5,967.95	168.54
	First N1	189,424	INT	5,966.37	270.11
	RMP B	189,424	INT	5,966.37	270.11
	Final polish	189,424	INT	5,073.07	486.23

Accepted time is the time when the retained incumbent was found, not the end of the stage. Repeated costs and times indicate an unchanged incumbent. RMP A is retained because it improves M4 in Large/test_07.

Retrospectively identifiable error. Table 24 focuses on the LOO pool, the first selection after N1 expansion and the final selection in that same expanded pool. The Small LOO receipt leaves only a numerical selection residual, while its pool restriction loss is at least 403.33 thousand units relative to the known feasible solution H. Medium LOO stopped at its time limit without a retained numeric pool bound, so the same conclusion cannot be drawn for that case.

At the first N1 selection, later improvements on the identical saved pool establish selection-residual lower bounds that round to 317.33, 218.10 and 893.30 thousand units for Small, Medium and Large/test_07, respectively. These are lower bounds at the first N1 boundary, not estimates of the final residual. No numeric bound survives for the final N1 pools. Consequently, the final Small and Medium differences from L cannot be uniquely divided among selection residual, pool restriction loss and reference slack. No verified global lower-bound reference is available for Large/test_07, and its displayed LOO/N1/final records also lack a numeric same-pool bound. Their selection lower bounds remain identifiable, but finite upper endpoints and reference slack do not. The best observed feasible objective used for RPD in Table 31 cannot replace a global lower bound in this decomposition.

Table 24: Retrospectively identifiable pool-error intervals (thousands of objective units).

Case	Boundary	Selection residual	Pool restriction loss	$U_j - L$
Small / 07	LOO selection	$[0,\ 5.1 \times 10^{-6}]$	[403.33, 723.62]	723.62
	First N1 selection	[317.33, 637.61]	[0.00, 320.29]	637.61
	Final polish	[0.00, 320.29]	[0.00, 320.29]	320.29
Medium / 07	LOO selection	[0.00, 1,249.93]	[0.00, 1,249.93]	1,249.93
	First N1 selection	[218.10, 843.22]	[0.00, 625.12]	843.22
	Final polish	[0.00, 625.12]	[0.00, 625.12]	625.12
Large / 07	LOO selection	[0.00, –]	[0.00, –]	–
	First N1 selection	[893.30, –]	[0.00, –]	–
	Final polish	[0.00, –]	[0.00, –]	–

Endpoints are rounded for display. The Small LOO selection upper endpoint is conservatively displayed as 5.1×10^{-6} thousand units (0.0051 original units). "–" means no reported finite endpoint, not a measured zero. Reference slack lies in $[0, 320.29]$ for Small and $[0, 625.12]$ for Medium, in thousands; no such interval is available for Large.

Immediate adoption of new routes. Table 25 counts new, distinct order–route candidates by comparing saved pools, then counts those selected at the corresponding phase boundary. Four fields compares stages 0 and 8, LOO compares stages 9 and 11, and N1 compares stages 11 and 13. The order share divides adopted candidates by 800, 1,600 or 3,200 orders; utilization divides adopted candidates by the number added.

Table 25: Immediate adoption of newly generated routes.

Case	Phase	Added	Adopted	Orders (%)	Utilization (%)
Small / 07	Four fields	1,503	648	81.00	43.11
	LOO	540	329	41.12	60.93
	N1	45,472	438	54.75	0.96
Medium / 07	Four fields	3,154	1,301	81.31	41.25
	LOO	1,090	734	45.88	67.34
	N1	87,444	655	40.94	0.75
Large / 07	Four fields	6,137	2,627	82.09	42.81
	LOO	2,080	1,421	44.41	68.32
	N1	165,576	3	0.09	< 0.01

Adoption is measured at the phase boundary, not over the rest of the run. An unused route may be useful later. Counts and rates have different denominators and do not attribute causal savings to a phase.

LOO uses a much larger fraction of its added candidates immediately than N1 does. In Large/test_07, the first N1 selection adopts only three newly added candidates, although later selection on the expanded pool reduces cost substantially. This separates immediate route adoption from subsequent same-pool selection progress; it does not establish the eventual value of each generated route.

True-cost components. Table 26 reports vehicle, handling and lateness costs from the saved validation records at the same boundaries as Table 23.

Vehicle cost falls from the initial selection to the final incumbent in all three test_07 cases, while handling cost changes more modestly. Final lateness cost exceeds its initial value in Small, Medium and Large. During the last fixed-pool polish, all three cases trade lower vehicle cost against higher lateness cost; handling cost rises in Small and Medium and falls in Large. These are cost decompositions of the selected schedules under the original objective, not causal contributions of algorithm components.

Scope. Saved-pool intervals do not isolate learning error or certify exhaustion of feasible routes. Stage improvements depend on earlier pools and selection effort. This same-network, single-seed

Table 26: True-cost components at saved boundaries for test_07 at each scale (thousands of objective units).

Boundary	Small / 07			Medium / 07			Large / 07		
	Vehicle	Handling	Lateness	Vehicle	Handling	Lateness	Vehicle	Handling	Lateness
Initial	3,303.40	596.30	71.18	5,517.66	1,264.06	196.51	8,254.79	2,547.94	278.15
M4	1,790.64	601.18	133.36	2,648.39	1,278.60	277.15	3,710.51	2,565.72	401.16
RMP A	1,790.64	601.18	133.36	2,648.39	1,278.60	277.15	3,706.56	2,565.63	400.86
LOO	1,254.42	608.08	148.33	1,935.91	1,278.00	289.75	3,050.10	2,558.48	359.36
First N1	1,174.77	608.70	141.36	1,518.78	1,274.55	303.61	3,048.52	2,558.48	359.36
RMP B	1,075.06	607.81	145.84	1,427.53	1,277.53	310.74	3,048.52	2,558.48	359.36
Final polish	791.17	616.67	199.66	1,264.46	1,281.90	332.48	2,127.30	2,554.69	391.08

Within each case, the three unrounded components sum to the corresponding objective in Table 23; displayed components are rounded independently, and repeated rows retain the same selected solution.

study uses fixed Medium weights and does not establish arbitrary-scale or unseen-network generalization.

J Detailed experimental protocols and results

J.1 Complete evaluation protocol

Data and independence. The public SND-RR archive of Van Dyk & Koenemann (2023) contains 576 generated instances in 80 source-network groups. A metadata-only split assigns 48/16/16 groups and 348/120/108 instances to training/validation/test. Source variants remain together, and a directed-isomorphism audit finds no duplicated base topologies. The test families contain 24/36/48 instances from 2/6/8 critical-times/designated-paths/hub-and-spoke groups. Network groups, rather than individual variants, are the units of statistical replication.

The native benchmark preserves commodity-specific transport costs, route permissions, hard deadlines, zero-duration arcs and unlimited integer vehicles; it has no reference-specific terminal caps or ownership modes (Appendix E). All three encoders are retrained on structural features without network-specific identifiers, using seeds 11/22/33. Independently validated feasible teachers supervise training; validation teachers select checkpoints, and test solutions never supply training labels. The primary matrix was frozen before its test outcomes and includes all 19 arms and 2,052 scheduled runs. Demand-scale tests reuse the same held-out source groups, and the real reference instance is a development case.

Matched controls and computation. Each primary run receives a 30-second nominal online budget and eight solver threads. Analytic, random and learned field methods use one chain, two common baseline fields and six additional fields, with the same decoder, enrichment rules and master. Full MIP cold/warm starts and the authors' arc/node dynamic discretization discovery (DDD) implementations supply optimization baselines. Warm MIP receives the common full-charge path assignment; DDD retains its one-percent stopping rule. Runs are interleaved on one CPU host. The online clock includes input processing, model loading, CPU inference, search and validation from benchmark entry; interpreter startup and imports completed before benchmark entry are excluded. Actual runtimes and overruns are retained.

Costs, gaps and uncertainty. All feasible costs U are independently reconstructed under the original model. For a paired comparison, savings are $100(U_{\text{control}} - U_{\text{method}})/U_{\text{control}}$, so positive values favor the named method. We first average frozen-seed costs within each instance, calculate its savings, average within each source group, then weight groups equally. Intervals use 20,000 paired group-bootstrap draws and are unadjusted 95% intervals conditional on the frozen models and paired feasibility. Missing solutions retain their feasibility denominators and receive no imputed cost.

Gap is $100(U - L)/U$, where L is the strongest available valid global bound for that instance from complete-model or DDD runs; restricted-master bounds are excluded. Table gaps average feasible seed–instance runs, whereas savings weight source groups equally. Wall time is the median over individual runs. Offline teacher generation and training are reported separately. Subsequent affine, portfolio and anytime studies reuse the original test groups and are identified as secondary follow-ups (Appendix H).

J.2 Additional held-out results

Table 1 reports all methods and Figure 1 separates aggregate and family behavior. Group-mean savings against analytic search are 2.73% [1.77, 3.80], 2.84% [1.89, 3.85], 2.54% [1.65, 3.54] for CellMLP, DeepSets and latent attention, respectively (unadjusted 95% intervals). These comparisons hold field count and search code fixed. They concern the complete held-out set rather than the repeatedly evaluated real case.

Learning improves both average cost and median runtime within the shared recommendation–optimization pipeline. Median online time falls from 23.6 seconds for analytic search to 7.5,8.9 and 8.1 seconds for CellMLP, DeepSets and latent attention. Median retained pool sizes are 1,475,1,554 and 1,526.5 paths, versus 2,017 for analytic search. Median inference takes 5.5,8.4 and 23.9 milliseconds; approximately one second of checkpoint loading is also included in the online clock. The speed result therefore concerns the complete algorithm, not inference alone.

Table 27: Secondary scale stress: eight methods, sixteen 1,600-demand cases, one prespecified training seed, 120 nominal seconds and eight threads. Savings, gap and timing conventions match Table 1; each method has sixteen attempts. Intervals concern the same sixteen parent networks, not additional independent data.

Method	Feasible	Savings (%) [95% CI]	Gap (%)	Wall (s)
Analytic	16/16	0.00 [0.00, 0.00]	5.62	120.3
CellMLP	16/16	1.16 [0.32, 2.13]	4.54	120.3
DeepSets	16/16	0.71 [0.13, 1.37]	4.97	120.3
Latent attention	16/16	0.50 [0.07, 0.95]	5.16	120.3
Full MIP cold	16/16	1.97 [0.59, 3.42]	3.74	121.3
Full MIP warm	16/16	1.90 [0.53, 3.40]	3.81	121.1
Arc DDD	16/16	-13.50 [-19.38, -8.08]	15.93	120.2
Node DDD	4/16	-8.21 [-12.47, -4.25]	9.14	120.3

The direct paired savings of latent attention against CellMLP and DeepSets are -0.20% [-0.40, -0.01] and -0.31% [-0.53, -0.05]. Simple encoders remain effective within the learned-service interface.

Complete warm-started MIP remains a useful quality reference on these public instances: all 108 runs are feasible, its median runtime is 30.1 seconds, and learned costs are 3.33–3.65% higher relative to its objective. This comparison complements the matched nonlearned-recommendation control. Early search termination gives different actual runtimes under the common nominal cap; the separate anytime experiment compares validated solutions at fixed observation times.

Removing LOO or service closure is evaluated using the same prespecified latent checkpoint (seed 11). The complete search's paired savings against those ablations are 1.93% [1.48, 2.48] and 0.60% [0.23, 1.13]. These are single-checkpoint search diagnostics, not uncertainty over retraining. Appendix F supplies cost profiles that retain unsuccessful attempts.

A thoroughly fitted 39-parameter affine scorer also improves on analytic recommendation. CellMLP's additional cost saving is 0.20% $[-0.45, 0.91]$, while its median runtime is 8.5 versus 16.5 seconds for that control (Appendix H.3).

A matched four-chain study further supports learned complementarity: the heterogeneous neural portfolio saves 1.85% [1.18,2.55] against analytic chains, with median runtime 22.8 versus 48.4 seconds, and improves on repeating each constituent model by 0.34–0.52% (Appendix H.6).

J.3 Derived 1,600-demand scale stress

We retain the 16 held-out source networks and expand their prespecified representative instances to 1,600 demands, using the fixed generator in Appendix E. No model is retrained; all three architectures use seed 11. Eight arms run at 120 nominal seconds on one separate CPU host. These are derived scale-shift cases, not 16 additional independent networks, new real records, or replicas of the reference timetable.

The respective learned-field savings against analytic search are 1.16% [0.32, 2.13], 0.71% [0.13, 1.37], 0.50% [0.07, 0.95]. Paired savings against full warm MIP are -0.79% [-1.68, 0.09], -1.28% [-2.53, -0.10], -1.50% [-2.90, -0.23]; negative values again favor full optimization. Exception and no-incumbent outcomes remain in Table 27, including any earlier valid incumbent retained before an implementation exception. The original DDD scheduling code attempts to read a variable value after an LP deadline without a solution in two stress runs. Arc DDD retains an earlier validated incumbent in its affected run; node DDD has no retained incumbent in its affected run. Both exceptions remain in the records.

The fully instrumented short-budget comparison confirms an early quality advantage at equal elapsed time: at 10 seconds, the three learned methods save 3.24–4.89% relative to complete warm MIP, with all 16 cases feasible for every method. CellMLP saves 2.32% [0.22, 4.91] over analytic

recommendation at that time and 1.13% [0.27, 2.12] at 120 seconds. Complete MIP improves more at later horizons. Appendix H.7 reports all checkpoints, target attainment and startup costs.

J.4 REFERENCE-1602: TRAINING AND SUPPLEMENTARY COMPARISONS

Reference training data and selection. Reference-1602 denotes the anonymized 1,602-shipment development instance. Its historical fields are trained on perturbed demand books on the same 69-lane network. Subsampling retains a fraction of the reference orders; block bootstrap resamples demand contexts and perturbs shipment weights while preserving the network and service templates. The unperturbed 1,602-shipment development instance is not itself a training record. The released training corpora, denoted Corpus A and Corpus B, contain 366 and 854 records, with 2,122 and 2,976 validated design checkpoints, respectively. Labels combine heuristic teacher solutions, exact-solver small-instance solutions, and, in Corpus B, 390 model-guided solver records; 39 exact records also carry 69 alternative designs. Each record stores demand-context features, service features and three pressure channels on the fixed $69 \times 889 = 61{,}341$ service-cell grid, together with activation and load targets.

Training and validation are separated by sampler family, rather than by individual service cells. The seven model checkpoints are denoted C1–C7. The DeepSets and latent-attention runs C1 and C2 hold out the `sub0.50` and `boot1.00` families, giving 287 training and 79 validation records. The C3 and C4 runs use their recorded family splits of 275/91 and 641/213. The three Corpus B runs C5–C7 use the same 79-record holdout as the original pair and retain 724 training records after removing 51 teacher records shadowed by exact labels. Feature normalization is fitted on the training side only. Models minimize quality-weighted activation focal loss with an auxiliary load loss using AdamW; the selected checkpoint maximizes validation load-weighted Recall@200. Appendix B.2 specifies the per-checkpoint settings and validation diagnostics. These are demand perturbations of one development network, not independent real-world test networks; offline label generation and training precede the reported online clock.

The reported operating point serves all 1,602 transport-requiring orders with 126 owned departures and no outsourcing. Its independently replayed cost is $364{,}865.72469$: vehicle cost $94{,}185$, handling $264{,}298.72469$, and lateness $6{,}382$. Against the separate historical global bound $353{,}490.6928953997$, its gap is 3.118%. Starting with frozen checkpoints, the corpus prior and precomputed input features, the run takes 807.08 seconds (13.5 minutes).

Restricting optimization to learned and prior-guided proposals makes a high-quality joint schedule accessible, while the complete arc formulation contains 33,703,622 variables and 33,754,661 rows. In our direct-solve audit, the complete model stops for memory after 2,526.52 seconds under a 3,600-second total budget, eight threads and a 160-GB solver memory limit. Its feasible incumbent costs $2{,}047{,}465.10$ after removing excess vehicles. The two runs use different parallelism and time origins, so their timings describe operating configurations rather than a matched speedup. The global bound is separate historical evidence; the restricted master's own bound is not used for the 3.118% gap (Appendix B).

We additionally run the recovered production search from the anonymous package, starting with frozen label-free feature tensors but no previous pool or solution. Both chains use four sources and sixteen fields, the same search plan, solver seed and worker limits on one CPU host. The neural chain combines two prior sources with DeepSets and latent attention; the control uses four different prior support pairs. It yields $365{,}478.87$ versus $367{,}596.13$, a 0.576% reduction. Internal chain times are 1,866.22 and 1,898.89 seconds; outer process times are retained separately. Both exported solutions are independently revalidated. This is a single development-case comparison, not a training-seed confidence interval, and does not isolate attention from the combined neural portfolio. Fresh time-limited runs also need not recover the best historical six-chain incumbent.

Historical source-removal controls improve with neural sources, but they also increase the number of fields. They cannot isolate neural information from proposal volume. Appendix G preserves all operating points and this confound. The fixed-field external experiments and the fresh four-source comparison are the stronger controls for the representation.

J.5 NINE-CASE FIXED-WEIGHT STUDY

The nine-case study uses demand sets of 800, 1,600 and 3,200 orders, denoted Small, Medium and Large, on the same physical network. The inverse-field model is trained on Medium scenarios 01–05 and selected on scenario 06. Its weights and training-set normalization are then frozen for scenarios 07–09 at all three scales; order sets differ across scales even when scenario identifiers agree. Tables 29–31 report six main methods: WAVECAST, CellMLP, DeepSets, latent attention, Slope Scaling and Local Branching. Appendix I.1 describes its supervision and training protocol.

All runs use two solver threads, online seed 11, and a 600-s accepted-incumbent deadline, including input loading, inference, decoding, and optimization. CellMLP and DeepSets (Zaheer et al., 2017) use width 128; Latent (Lee et al., 2019; Jaegle et al., 2021) uses width 256, 64 latent tokens, and eight attention heads. These baselines retain their 60-epoch AdamW fits. Training curves, accepted-incumbent trajectories and pool diagnostics for test_07 at each scale are provided in Appendix I. The Slope Scaling (Gendron et al., 2018) and Local Branching (Fischetti & Lodi, 2003) baselines are adapted to the present model; Local Branching operates with at most $\lfloor N/5 \rfloor$ changed routes for N orders. The component and field controls are defined in Table 28. LOO generates leave-one-out alternatives; N1 generates service-level replacement alternatives for the candidate pool. Feedback denotes these two incumbent-dependent expansion stages. The four learned heads are field sources within one chain, not a four-chain ablation. All retained restricted masters use the original mixed-integer model.

Table 28: Controls in the nine-case study. FULL denotes the main WAVECAST execution; the six controls below are distinct settings. Field controls retain four proposal fields.

Control	Change from FULL
Without LOO	Disable leave-one-out candidate expansion; retain N1 and pool selection.
Without N1	Disable N1 candidate expansion; retain LOO and pool selection.
Without feedback	Disable both LOO and N1 expansion; retain field-generated candidates and pool selection.
Without nested RMP	Replace restricted-pool mixed-integer selection with greedy insertion.
Analytic fields	Replace the learned fields with four analytic fields; retain the subsequent search.
Random fields	Replace the learned fields with four fields using independent Uniform$(0.9, 1.1)$ multipliers; retain the subsequent search.

Each case has a separate 1,800-s allowance for a P12 full-path numerical lower-bound reference. The six existing Small/Medium references are retained. Specifically, obtaining a verified full-path global lower bound for the Large instances proved difficult within the available time and memory budgets. For Small and Medium, gap is $100(U - L)/U$ using the independently replayed lower endpoint L, not an LP upper endpoint or a restricted-master bound. These are binary64 numerical checks rather than exact-arithmetic proofs. For Large, we instead report relative percentage deviation, $\mathrm{RPD} = 100(U - U_{\mathrm{best}})/U_{\mathrm{best}}$, where U_{best} is the lowest validated feasible objective for the same instance across the complete 13-setting registry, including Portfolio (Table 32). Its value agrees with the minimum across the two displayed panels.

Observed cost and component effects. WaveCast has the lowest feasible objective among the six reported main methods on all nine cases. Relative to the lowest-cost available competing main method in each case, its savings are 4.18–5.04% on Small, 0.12–3.14% on Medium, and 5.28–7.37% on Large. The Small numerical gaps are 16.64–19.92%, and Medium gaps are 18.37–22.44%. On Large, WaveCast has RPD 0.00%, 0.40%, and 0.00% on test_07, test_08, and test_09, respectively, against the best observed feasible objectives across both panels. These final-cost comparisons do not establish an equal-time speedup, since several baselines stop before the common cap.

Removing LOO, N1, feedback, or nested RMP gives a higher cost on every case. With savings defined as $100(U_{\mathrm{variant}} - U_{\mathrm{FULL}})/U_{\mathrm{variant}}$, FULL improves on these four controls by 2.30–5.90%, 10.59–20.34%, 19.17–36.35%, and 2.60–12.60%, respectively. The field comparison is more modest: learned fields reduce cost relative to analytic fields by 0.66–3.40% on all nine cases, and outperform random fields on eight. On Large/test_08, random fields cost $4{,}927{,}453.29$, versus $4{,}947{,}212.16$ for FULL, an advantage of 0.40% relative to FULL. The field-level gains are smaller

Table 29: Small (800 orders): objective values and numerical lower-bound gaps with fixed Medium weights under a 600-s accepted-incumbent limit.

(a) Main comparison

Method / variant	test_07		test_08		test_09	
	Objective	Gap (%)	Objective	Gap (%)	Objective	Gap (%)
WaveCast	**1,607,498.76**	**19.92**	**1,640,826.41**	**16.64**	**1,660,616.46**	**18.40**
CellMLP	1,760,266.86	26.87	1,789,227.38	23.56	1,791,214.52	24.35
DeepSets	1,750,364.91	26.46	1,753,531.60	22.00	1,796,317.64	24.56
Latent	1,741,791.34	26.10	1,758,957.53	22.24	1,792,504.91	24.40
Slope Scaling	1,692,814.35	23.96	1,721,408.31	20.54	1,733,107.01	21.81
Local Branching	1,766,051.63	27.11	1,755,080.49	22.07	1,790,335.65	24.31
Lower bound L	1,287,210.94		1,367,763.88		1,355,071.26	

(b) Component and field ablations

Method / variant	test_07		test_08		test_09	
	Objective	Gap (%)	Objective	Gap (%)	Objective	Gap (%)
FULL	**1,607,498.76**	**19.92**	**1,640,826.41**	**16.64**	**1,660,616.46**	**18.40**
Without LOO	1,697,583.37	24.17	1,687,435.78	18.94	1,717,945.98	21.12
Without N1	2,010,830.78	35.99	2,016,264.33	32.16	2,084,590.22	35.00
Without feedback	2,525,182.82	49.03	2,480,699.85	44.86	2,608,929.81	48.06
Without nested RMP	1,826,287.63	29.52	1,837,360.22	25.56	1,900,112.39	28.68
Analytic fields	1,640,461.21	21.53	1,651,711.70	17.19	1,710,994.35	20.80
Random fields	1,640,003.20	21.51	1,651,954.37	17.20	1,687,076.98	19.68
Lower bound L	1,287,210.94		1,367,763.88		1,355,071.26	

Table 30: Medium (1,600 orders): objective values and numerical lower-bound gaps with fixed Medium weights under a 600-s accepted-incumbent limit.

(a) Main comparison

Method / variant	test_07		test_08		test_09	
	Objective	Gap (%)	Objective	Gap (%)	Objective	Gap (%)
WaveCast	**2,878,838.52**	**21.71**	**3,045,667.82**	**18.37**	**2,890,139.58**	**22.44**
CellMLP	3,123,592.80	27.85	3,312,748.21	24.95	3,140,711.04	28.63
DeepSets	3,109,009.11	27.51	3,297,429.67	24.60	3,135,291.25	28.50
Latent	3,100,348.52	27.31	3,292,896.47	24.49	3,122,194.54	28.20
Slope Scaling	2,931,723.86	23.13	3,144,337.17	20.93	2,893,725.36	22.53
Local Branching	3,106,030.26	27.44	3,317,013.65	25.04	3,112,252.31	27.97
Lower bound L	2,253,721.55		2,486,302.66		2,241,636.71	

(b) Component and field ablations

Method / variant	test_07		test_08		test_09	
	Objective	Gap (%)	Objective	Gap (%)	Objective	Gap (%)
FULL	**2,878,838.52**	**21.71**	**3,045,667.82**	**18.37**	**2,890,139.58**	**22.44**
Without LOO	2,994,818.53	24.75	3,166,029.65	21.47	3,014,955.55	25.65
Without N1	3,503,639.65	35.67	3,659,847.53	32.07	3,551,684.29	36.89
Without feedback	4,204,139.99	46.39	4,259,945.00	41.64	4,315,409.18	48.06
Without nested RMP	3,141,429.26	28.26	3,373,400.56	26.30	3,161,007.03	29.08
Analytic fields	2,917,050.67	22.74	3,152,761.60	21.14	2,972,007.01	24.57
Random fields	2,896,087.57	22.18	3,098,151.62	19.75	2,926,694.01	23.41
Lower bound L	2,253,721.55		2,486,302.66		2,241,636.71	

Table 31: Large (3,200 orders): objective values and relative percentage deviations from the best observed feasible objective with fixed Medium weights under a 600-s accepted-incumbent limit.

(a) Main comparison

Method / variant	test_07		test_08		test_09	
	Objective	RPD (%)	Objective	RPD (%)	Objective	RPD (%)
WaveCast	**5,073,066.30**	**0.00**	**4,947,212.16**	**0.40**	**5,086,249.09**	**0.00**
CellMLP	5,505,394.10	8.52	5,223,037.05	6.00	5,576,443.40	9.64
DeepSets	5,574,334.65	9.88	5,335,554.87	8.28	5,615,210.07	10.40
Latent	5,580,268.24	10.00	5,314,202.64	7.85	5,701,287.58	12.09
Slope Scaling	5,470,809.09	7.84	5,304,489.43	7.65	5,517,770.36	8.48
Local Branching	5,648,248.32	11.34	5,981,350.11	21.39	5,639,531.24	10.88
Best observed U_{best}	5,073,066.30		4,927,453.29		5,086,249.09	

(b) Component and field ablations

Method / variant	test_07		test_08		test_09	
	Objective	RPD (%)	Objective	RPD (%)	Objective	RPD (%)
FULL	**5,073,066.30**	**0.00**	4,947,212.16	0.40	**5,086,249.09**	**0.00**
Without LOO	5,192,293.87	2.35	5,102,063.16	3.54	5,404,916.36	6.27
Without N1	5,735,365.11	13.06	5,533,458.01	12.30	5,876,637.27	15.54
Without feedback	6,669,309.50	31.47	6,120,857.02	24.22	6,807,941.19	33.85
Without nested RMP	5,286,487.09	4.21	5,150,964.40	4.54	5,221,876.10	2.67
Analytic fields	5,150,276.95	1.52	4,992,672.82	1.32	5,207,534.54	2.38
Random fields	5,173,559.20	1.98	**4,927,453.29**	**0.00**	5,129,306.29	0.85
Best observed U_{best}	5,073,066.30		4,927,453.29		5,086,249.09	

than those of the search-component ablations, and learned fields do not uniformly dominate random proposals.

These results support reuse of the frozen weights across the observed demand scales on the same physical network. They do not establish generalization to unseen networks or arbitrary scales.

Complete evaluation registry. The complete registry contains 13 distinct settings per case: the six displayed main methods, Portfolio, and six non-FULL ablations. FULL repeats the same WAVECAST run in the main and ablation panels. Of the 117 scheduled entries, 116 yield validated feasible incumbents; Portfolio has no validated incumbent on Large/test_08. The 117 entries comprise 53 previously recorded executions and 64 additional executions; no entry is selected as the best of repeated attempts. Portfolio comprises two latent-attention chains, one DeepSets chain and one CellMLP chain, each with one feedback round. Table 32 retains its outcomes separately from the six-method tables. The diagnostic reference H in Appendix I and the Large-instance RPD denominator use the lowest validated cost across this complete registry, excluding unavailable incumbents. Including Portfolio does not change these minima.

Table 32: Portfolio outcomes from the complete nine-case registry. Small/Medium report numerical global-bound gaps; Large reports best-observed RPD. A dash denotes no validated feasible incumbent, not a zero cost.

Scale	Case	Objective	Gap/RPD (%)	Outcome
Small	07	1,855,471.40	30.63	Feasible
	08	1,798,121.41	23.93	Feasible
	09	1,911,591.36	29.11	Feasible
Medium	07	3,262,138.12	30.91	Feasible
	08	3,418,940.54	27.28	Feasible
	09	3,288,368.17	31.83	Feasible
Large	07	5,457,586.85	7.58	Feasible
	08	–	–	No incumbent
	09	5,490,929.35	7.96	Feasible