How Much Memory Does Temporal Graph Reasoning Need?

Information Limits for Deferred Temporal Path Queries

Anonymous authors
Paper under double-blind review

Abstract

A partial route that is not currently best may become optimal after a future connection appears. A system that summarizes a temporal graph before that future query is known must therefore preserve several possible routes. We formalize the resulting memory problem for deferred temporal path queries. On a three-vertex family encoding k independently queryable facts, any prefix-summary or one-pass streaming protocol requires $\Omega(k)$ bits, whereas one bit suffices when the query is known during encoding. We introduce **TempoFrontier**, a diagnostic benchmark with exact solvers, verified witnesses, query-only and full-instance references, and explicit inference-time bit bottlenecks. Learned event encoders exhibit an unavoidable low-capacity regime and a persistent gap from an achievable code after lossless storage becomes possible. Under repeated updates, the prefix information remains partly decodable, while task accuracy and selective use of the code decline. On procedurally generated graphs and real transit timetables, node-indexed memory with stable node identities outperforms the tested global summaries at matched bit budgets, especially when the completion requires prefix-derived labels. These results separate the roles of information capacity, learned utilization, retention, and addressability in deferred temporal graph reasoning.

1 Introduction

Consider two partial journeys that reach the same station. One departed at 8:00 and arrived at 8:30; the other departed at 8:20 and arrived at 8:40. Which should a planner remember? If the next usable connection leaves at 8:35, only the first can catch it. If it leaves at 8:45, both can—and the second, having started later, completes the faster door-to-door trip. Neither partial journey dominates the other; *a future event decides*. Exact algorithms for temporal path problems handle this by maintaining, at each vertex, the set of mutually nondominated partial paths—the *frontier* (Wu et al., 2014; Dibbelt et al., 2018). A learned or compressed system may represent the same information differently, but it must still retain enough of the prefix to answer the future query. We show that, for a family in which future queries can select among many such alternatives, any bounded reusable summary must retain information that grows linearly with the number of independently queryable facts.

Learning systems face an additional constraint. A recurrent model, an agent's memory, a log summarizer, or any system that answers questions about a stream it can no longer replay must compress the past *before* knowing which part of the past will matter. This paper asks the resulting question in a setting where it can be answered exactly:

> *How much information must a system preserve when it summarizes a temporal prefix before knowing which future query will be asked—and how should that information be organized for later use?*

Existing evaluations of graph reasoning vary graph size, path length, density, or label noise; none of these measures how much *past information a task inherently requires*. We make that quantity the controlled variable. Our contributions:

1. Deferred-query information limits. We construct temporal path families encoding k independently queryable facts, prove an $\Omega(k)$ prefix-memory lower bound (Theorem 1), derive a dis-

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.

tributional accuracy bound (Proposition 1), and show that revealing the query before compression reduces the exact requirement to one bit (Proposition 2).

2. TempoFrontier. We introduce a diagnostic benchmark that aligns hard inference-time bit bottlenecks with exact solvers, verified witnesses, query-only and full-instance references, and controlled gadget and non-gadget instances (Sections 4–6 and Appendix D).

3. Learned memory findings. We separate information insufficiency from code learning, identify retention-induced corruption and entanglement, and show that stable node-indexed memory improves retrieval when future completions require prefix-derived labels (Section 6).

We make no speed claims: exact one-pass algorithms solve these problems and serve here as references. The lower bound applies to models that summarize before the query arrives, not to models that see prefix and query jointly (Table 1).

2 Temporal Frontiers and Deferred Queries

Temporal graphs and walks. A temporal graph is a vertex set V with a finite list of *events* $e = (u, v, t, \lambda)$: a directed connection departing u at integer time $t \geq 0$ and reaching v at $t + \lambda$, with integer duration $\lambda \geq 1$; parallel events are allowed. A *temporal walk* is a sequence of events with matching endpoints and $t(e_{i+1}) \geq t(e_i) + \lambda(e_i)$. Waiting is free and vertices may repeat: requiring simple paths introduces NP-hardness that is orthogonal to our question (Casteigts et al., 2021). For a walk P, $\mathrm{start}(P)$ and $\mathrm{arrival}(P)$ are the first departure and last arrival; $\mathrm{duration}(P) = \mathrm{arrival}(P) - \mathrm{start}(P)$. The *fastest-path* problem asks for the minimum-duration walk from a source to a target.

Why one best-so-far is not enough. A partial walk to v is summarized by the label $\ell = (s, a)$: its start and current arrival. A label may be discarded only if another *dominates* it—starts no earlier and arrives no later ($s' \geq s$, $a' \leq a$, one strict). Walks with equal label values are interchangeable for every future completion, so the frontier $F(v)$ keeps one representative per nondominated *value*. This pruning is *extension-safe*: the chronological scan that starts a label $(t,\ t{+}\lambda)$ at every event leaving the source, extends every feasible frontier label, merges equal values, and deletes dominated ones is exact (Theorem A, Appendix A). In the station example above, $(8{:}00, 8{:}30)$ and $(8{:}20, 8{:}40)$ are incomparable, and each is uniquely optimal under a different future connection. The frontier is an exact label-based summary of the prefix. The next section shows that, over a family with many independently queryable alternatives, any bounded reusable representation must carry information linear in the family dimension.

Prefix–completion protocol. We split an instance chronologically into a prefix P and a completion S and study systems of the form

$$m = E_\theta(P) \in \mathcal{M},\ \ |\mathcal{M}| \leq 2^b, \qquad \hat{y} = D_\theta(m, S), \tag{1}$$

where the encoder is *arbitrary* (recurrent scan, GNN, Transformer, or an exact algorithm) but its output must cross a b-bit boundary before S is revealed, and the decoder never revisits P. We distinguish three evaluation regimes according to when the query becomes available:

Regime	Sees whole prefix	Query visible during encoding	Lower bound applies
Prefix summary	yes	no	**yes**
Streaming	incrementally	no	**yes** (Cor. 1)
Full instance	yes	yes	no (control)

Table 1: Evaluation regimes. The lower bound applies when the query is unavailable during prefix encoding; full-instance models serve as controls because they observe the query before constructing their representation.

Two notions of required state. We use two measures of required state. *Information complexity* measures the number of bits needed by a reusable summary over a family or distribution of prefixes and future queries; it applies to any bounded representation (Theorem 1, Prop. 1). *Exact frontier*

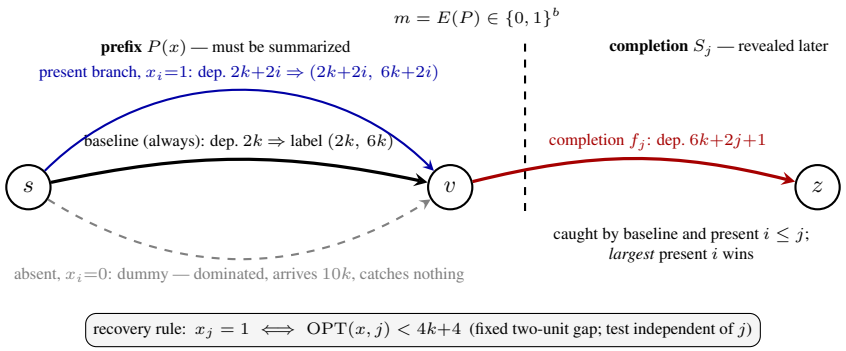

Figure 1: **The three-vertex lower-bound family.** The prefix encodes k independent bits as the presence or absence of mutually incomparable branch labels at v (later start trades against later arrival, so present labels form an antichain); dummies keep the event count at $k{+}2$ for every x without touching the frontier. The completion selects an index j, and the optimal *duration* of the resulting fastest walk reveals x_j through a fixed, query-independent two-unit gap. Any b-bit summary computed left of the boundary is therefore a one-way INDEX message (Theorem 1).

complexity measures the number of nondominated labels needed by an executor that stores labels explicitly, for one prefix and one declared completion family (W_{front}, W_{suf}; Definition 2). The two quantities need not coincide: an arbitrary code may compress several labels, while a fixed known prefix may require no communication (Table 2, appendix). We later test whether exact frontier complexity predicts the difficulty of learned compression and retrieval.

3 What Must Be Remembered: Lower Bounds

3.1 Worst-case memory

A three-vertex family with a k-bit secret. Figure 1 shows the construction. Vertices $\{s, v, z\}$; for $x \in \{0,1\}^k$ the prefix $P(x)$ has $k{+}1$ events from s to v: a *baseline* departing at $2k$ with duration $4k$ (label $(2k, 6k)$ at v); for each i with $x_i{=}1$ a *branch* departing at $2k{+}2i$ with duration $4k$ (label $(2k{+}2i,\ 6k{+}2i)$); and for each $x_i{=}0$ a *dummy* that is dominated by the baseline and arrives too late to catch any completion—so absent bits never inflate the frontier, and every instance has exactly $k{+}2$ events regardless of x. The present labels form an antichain of size $\|x\|_1{+}1$. The completion S_j is the single event $(v, z,\ 6k{+}2j{+}1,\ 2)$: it can be caught by the baseline and exactly the present branches $i \le j$, among which the *largest* present index gives the unique fastest walk, so $\mathrm{OPT}(x, j) = 4k{+}2j{+}3 - 2i^*$ with $i^* = \max(\{0\} \cup \{i \le j : x_i{=}1\})$. If $x_j{=}1$ then $\mathrm{OPT} = 4k{+}3$; if $x_j{=}0$ then $\mathrm{OPT} \ge 4k{+}5$—a fixed two-unit gap, so the recovery rule $x_j{=}1 \iff \mathrm{OPT} < 4k{+}4$ is a single threshold test independent of j: the optimal duration of a tiny routing instance reveals an arbitrary stored bit.

Definition 1 (Prefix-summary protocol)**.** An instance splits chronologically into a prefix P and completion S. A protocol with budget b is a pair (E, D) with shared public randomness: $m = E(P) \in \mathcal{M}$ with $|\mathcal{M}| \le 2^b$, and output $D(m, S)$. The encoder never sees S; the decoder never revisits P; the parameters of E and D are fixed before the instance is drawn and do not count toward b. Correctness is required with probability $\ge 2/3$ on *every* instance for Theorem 1, and on *average* over the stated distribution for Proposition 1. In our neural instantiation, E is an arbitrary trained encoder, m is the hard b-bit message at evaluation time, and D is the trained decoder. *Every* value made available to the decoder that depends on the prefix—tensors, shapes, masks, normalization constants—is part of m and counts toward b; the side-channel audit (Appendix C) supports this accounting for our implementation.

Theorem 1 (Memory lower bound)**.** *For every k, on the family above (3 vertices, $k{+}2$ events), any prefix-summary protocol* (1) *that decides* $\mathrm{OPT} < 4k{+}4$ *with error at most* $1/3$ *on every input requires* $b = \Omega(k)$.

Proof sketch. The recovery rule turns the protocol into a public-coin one-way protocol for INDEX(x, j) with the same message and error, and the randomized one-way communication complexity of INDEX is $\Omega(k)$ (Kremer et al., 1995; 2001). Full construction and lemmas in Appendix A. □

Independently of the proof, the construction is machine-checked exhaustively for $k \leq 6$ against two exact solvers and the closed-form optimum (Appendix A).

The bound applies to the *total* input-dependent summary. Because the vertex set has constant size, total node-indexed plus global memory cannot remain constant as k grows; we do *not* claim any individual node needs $\Omega(k)$ bits.

Corollary 1 (Streaming)**.** *A one-pass chronological executor's state at the prefix boundary is the message of a prefix-summary protocol; it therefore requires $\Omega(k)$ bits on the same family.*

3.2 AVERAGE-CASE ACCURACY AND QUERY TIMING

Proposition 1 (Distributional envelope)**.** *Let x be uniform on $\{0,1\}^k$, j uniform on $[k]$, independent, with public randomness R independent of both, and let the protocol's average error at deciding x_j be $\bar{e} = \Pr[\hat{X}_J \neq X_J]$ (all entropies conditioned on R; w.l.o.g. $\bar{e} \leq \frac{1}{2}$). Then $b \geq k\,(1 - H_2(\bar{e}))$, i.e., $\bar{e} \geq H_2^{-1}(\max(0, 1 - b/k))$; with accuracy $\bar{a} = 1 - \bar{e}$, $\bar{a} \leq 1 - H_2^{-1}(\max(0, 1 - b/k))$. Naming the decisive branch determines x_j, so branch accuracy obeys the same (possibly loose) envelope.*

Under the full-uniform distribution, Proposition 1 gives an accuracy ceiling for each message budget. At b=0, the best possible accuracy is $\frac{1}{2}$. For $b < k$, perfect recovery is impossible, although learned models may still fall below the best attainable b-bit code. At $b \geq k$, a lossless code exists (our truncation reference achieves it), so remaining error reflects code learning or generalization rather than insufficient capacity. Query timing changes the requirement itself:

Proposition 2 (Query-known contrast)**.** *On the same family, if the encoder observes the query index j before compressing, the one-bit message $m = x_j$ together with the fixed threshold rule decides $\mathrm{OPT} < 4k{+}4$ exactly. Hence future uncertainty alone moves the requirement from 1 bit to $\Omega(k)$ bits.*

Corollary 2 (Constant-weight family)**.** *Restrict to x uniform on $\{x : \|x\|_1 = k/2\}$. Every prefix then has junction frontier width exactly $k/2 + 1$, while $b \geq \log_2 \binom{k}{k/2} - k\,H_2(\bar{e}) = k - O(\log k) - k\,H_2(\bar{e})$ (Appendix A). The restriction fixes the number of present branches within each value of k: the lower bound is not caused by variation in Hamming weight or prefix-specific frontier size. (The bound is loose at b=0; the query-only baseline stays $1/2$ since every marginal is $1/2$.)*

3.3 EXACT-LABEL DEMAND ON NON-GADGET INSTANCES

To study non-gadget instances, we next define a completion-relative measure of exact-label demand; note the unit changes from bits to labels.

Definition 2 (Minimal sufficient width)**.** Fix a prefix P with boundary frontier $F(P)$ and a declared finite completion family $\mathcal{S}$. A retained set $L \subseteq F(P)$ is *sufficient* if, for every $S \in \mathcal{S}$, solving the completion from L alone (together with walks originating inside S) yields the exact optimum of $P \circ S$. Then $W_{\mathrm{suf}}(P, \mathcal{S})$ is the minimum size of a sufficient set. It is a completion-relative requirement, dependent on the declared family $\mathcal{S}$ rather than an inherent property of the graph alone. We compute it exactly on our instances: one solve per (label, completion) pair reduces sufficiency to a minimum hitting set, which we solve by exhaustive subset enumeration—practical only because boundary frontiers are small in our families, not by a general efficient algorithm. Deletion testing—removing one label and checking whether any optimum changes—lower-bounds W_{suf} but is blind to interchangeable labels; the two differ on 0.5% of our procedural instances, and we bin by W_{suf} throughout. The estimate is stable across variations in the sampled completion family; Appendix B reports the sensitivity analysis.

Appendix A proves that an exact-label memory retaining fewer than W_{suf} labels must fail on at least one declared completion.

4 TempoFrontier: Turning the Theorem into an Instrument

TempoFrontier pairs each learned model with exact and non-learned references that isolate information capacity, code quality, and path correctness.

Controlled queries. The learning task is a graph-structured version of the theorem family: branches pass through intermediate vertices, branch *presence* is the hidden bit vector, instances carry random time shifts and vertex permutations, and absent branches are dominated distractors, so shape, event count, and degrees are identical across all 2^k patterns. The target is the decisive branch i^*. We call a prediction *verified* when its reconstructed witness walk is feasible and attains the exact optimum under the independent solver; *verified accuracy* is the fraction of verified predictions. Presence patterns are split train/test disjointly for $K_{\text{fam}} \geq 8$; smaller k are memorization-permitted controls (Appendix B).

Exact and non-learned references. (i) the *query-only baseline*, obtained by setting $b{=}0$ (≈ 0.5 on these families); (ii) an *achievable* hand-designed truncation code in closed form; (iii) the information bound (Prop. 1); (iv) *exact K-label slots* (8-bit fields, verified lossless)—inefficient but perfectly *addressable*; (v) the *full-instance* control (Table 1); (vi) two exact solvers cross-validated on 15,000 instances (zero discrepancies) and a standalone witness verifier.

Beyond the construction. The lower-bound family is additionally *embedded* in live carrier traffic (12 extra vertices, 3:1 distractors; inertness machine-checked per instance) and complemented by *procedurally generated* non-gadget temporal graphs (10 vertices, 30 events, fixed shape) split chronologically, with W_{suf} computed under a declared completion family and estimation perturbations disjoint from evaluation; we call these *procedural graphs*. An executable side-channel audit supports the conclusion that, for every message organization, the only prefix information reaching the decoder is the declared message; the budget binds at evaluation, while training uses an annealed continuous surrogate (Appendix C).

5 Memory Organizations

We group models by the organization of the boundary state. **Global compressed memory**: an encoder (a chronological event scan with a node-indexed state table—the learned analogue of the exact one-pass algorithms—or an attention encoder) pooled to one vector and quantized to exactly b hard bits. We train the hard sign message with LayerNorm and soft-to-hard annealing; Appendix C compares this procedure with a plain straight-through estimator. **Node-indexed memory**: each vertex's boundary state passes through its own shared-weight channel; the entire node table is the message, with $b_{\text{total}} = |V| \cdot b_{\text{node}}$ reported as such and never mixed with global curves. **Explicit label slots**: the exact reference (iv) above. **Full instance**: no bottleneck (the control regime of Table 1). Decoders are matched within each comparison; for procedural tasks the decoder is itself a scan that executes the completion from the decompressed memory. Five seeds per headline configuration; paired evaluation on identical instances; all hyperparameters were fixed before the replication campaign. The matched-battery interfaces, including parameter counts, are tabulated in Table 4 (Appendix C); the identity-aligned versus hidden-permutation pair is parameter- and bit-identical.

6 Results

Throughout, $\pm$ denotes standard deviation across seeds; per-bin sample sizes are 500 (procedural, confirmation evaluation), 750 (battery), and 4,096 instances per gadget collection.

6.1 Capacity and the information bound

Figure 2a compares verified accuracy with the message budget under the theorem-aligned and held-out evaluations. On the theorem-aligned track, accuracy increases with the budget and remains below the information-theoretic ceiling: the $K_{\text{fam}}{=}8$ scan stays at the query-only baseline for $b \leq 2$, reaches 0.904 at $b{=}k$, and approaches one by $2k$. Held-out patterns follow the same curve at slightly lower accuracy (0.836 at $b{=}k$, confirmation evaluation), indicating a modest generalization cost. Carrier embedding yields a similar budget–accuracy curve, with lower held-out accuracy at the

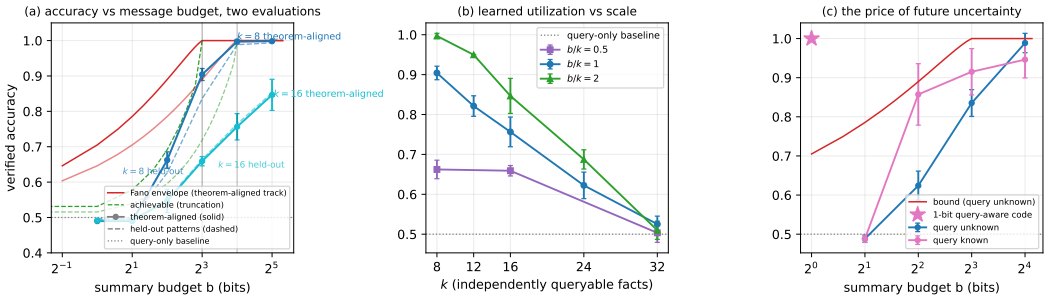

Figure 2: **Information limits and learned utilization.** (a) Verified accuracy versus message budget for the scan encoder under the theorem-aligned evaluation (solid; the red bound of Prop. 1 applies here) and the held-out-pattern evaluation (dashed); green dashes show the achievable truncation code. Error bars: standard deviation over seeds. Learned accuracy transitions through $b{=}k$ and approaches the achievable code only above it. (b) Accuracy at matched b/k for $k \in \{8, 12, 16, 24, 32\}$ (5 seeds for $k \in \{8, 16\}$, else 3): utilization decreases with k and reaches the query-only baseline by $k{=}32$. (c) Query timing at $K_{\text{fam}}{=}8$: with the query known during encoding, a one-bit code is exact (star, Prop. 2) and learned accuracy rises far earlier than in the query-unknown condition, to which alone the bound applies.

intermediate budget (0.768 vs 0.836 at $b{=}8$). The constant-weight family shows a similar transition when the number of present branches is fixed within each k (Cor. 2), and exact-label slots trace an addressable but information-inefficient staircase on their own raw-bit axis (Figure 5a,c).

6.2 Learned utilization and encoder scaling

Information sufficiency is not learnability. At $b{=}k{=}16$ the truncation code achieves 1.0, while the learned scan reaches $0.756 \pm .030$ (theorem-aligned; 0.764 held-out), and $b{=}32$ adds only about 0.09. Several diagnostics locate this gap. The small train-to-held-out difference (0.763 vs 0.741 at $k{=}16, b{=}16$) indicates that pattern memorization explains little of the error. The difference between the continuous-state and hard-message probes (0.775 vs 0.663 at $k{=}8, b{=}4$) shows an additional loss during quantization, and the 9.7-bit code entropy at $k{=}16, b{=}16$ indicates that the model does not use the full 16-bit alphabet. Finally, the presence-vector MLP reaches perfect accuracy with the same bottleneck and decoder when given the true presence bits (Table 3). This comparison identifies extraction of the relevant prefix facts from the event stream as a major remaining difficulty under the tested protocol. The gap also grows with k: accuracy at $b{=}k$ falls from 0.904 to 0.525 as k goes from 8 to 32 (Figure 2b). Among the event-based encoders tested under the shared protocol, only the node-keyed scan remains above the query-only baseline beyond $k{=}8$; a sequence GRU and Deep Sets fall to or below the baseline at $k{=}16$, as does one attention configuration whose representation retains the presence bits but whose end-to-end code is pattern-specific (Appendix C). The presence-vector MLP serves as a representation-free reference rather than a temporal encoder.

6.3 Future uncertainty

Query timing isolates the source of the memory burden. Revealing the query during encoding admits an exact one-bit code (Prop. 2); learned models at $b{=}4$ reach $0.857 \pm .079$ with the query known and $0.624 \pm .037$ with it unknown (Figure 2c). The learned curves reflect the theoretical contrast between the one-bit query-aware solution and the linear query-unknown requirement, while retaining a gap from the one-bit reference; the distributional bound applies only to the query-unknown condition.

6.4 Retention under repeated updates

What happens to a stored summary as irrelevant events arrive between frontier formation and use? For message bit r and predicted branch c, let B_{rc} be the probability that flipping r changes predictions on instances predicted as c (empty classes are skipped; the mean uses a small ϵ denominator).

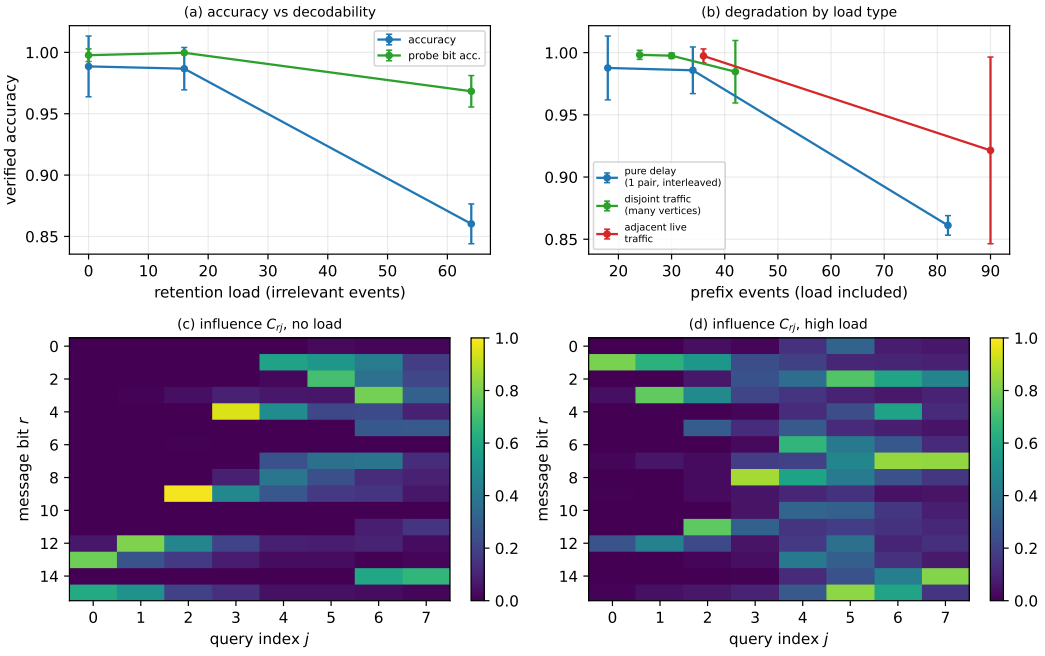

Figure 3: **Retention under repeated updates** (K_{fam}=8, b=2k, 5 seeds; error bars are standard deviations). (a) Task accuracy versus linear-probe decodability as load grows. (b) Degradation by load type, with the decision target machine-checked invariant under every load. (c–d) The causal influence matrix $C_{rj} = \Pr[\text{flipping bit } r \text{ changes the answer to query } j]$ at no load and high load: concentrated rows become diffuse.

Selectivity, $\text{mean}_r[\max_c B_{rc}/\text{mean}_c B_{rc}]$, equals 1 for a uniformly entangled code and k+1 for a factorized one.

Under increasing load (Figure 3), task accuracy falls from 0.989 to 0.860 while probe decodability barely moves (0.998 → 0.968); errors concentrate on asserting *absent* labels (from 0.007 to 0.103), and temporal feasibility violations remain at zero. The load-type comparison (Figure 3b) separates candidate causes: accuracy is essentially unchanged under disjoint traffic (0.985 ± .022 at the heaviest setting) but degrades under interleaved updates (0.861 ± .007 for pure delay; 0.921 ± .067 for adjacent live traffic). This suggests the cost lies in maintaining stored state across repeated updates rather than in sequence length or graph size. The intervention matrices (Figure 3c–d) show that each message bit affects a small set of queries at low load but a broader set at high load, and selectivity falls from 4.6 to 3.1. Because the decoder selects from a structured candidate set, the experiment does not test every aspect of path construction. It shows that suffix-feasibility decisions remain stable while the encoded branch-presence information becomes less accurate and less selectively usable. This pattern motivates memory mechanisms that protect or separate retained alternatives across repeated updates.

6.5 Memory Organization

Memory layout. Node-indexed state substantially outperforms the tested global summary on the bins that require prefix-derived labels (Figure 4a). On W_{suf}=1 instances the unquantized global summary reaches 0.05 exact match—approximately the query-conditional majority baseline of 0.056—while the per-node pathway with the same encoder class reaches 0.22 against a full-instance reference of 0.30. Increasing the node-indexed budget from 20 to 80 bits to unquantized improves accuracy monotonically (0.10 → 0.16 → 0.22 at W_{suf}=1), and even the 20-bit node-indexed message beats the unquantized global summary on $W_{\text{suf}}\leq$1 instances, so raw bit count alone does not explain the difference. The 20-bit message loses its advantage on the wider bins, suggesting that this budget is insufficient once several prefix-derived alternatives must remain recoverable; since W_{suf} counts exact labels rather than bits, we treat this boundary as empirical rather than information-theoretic.

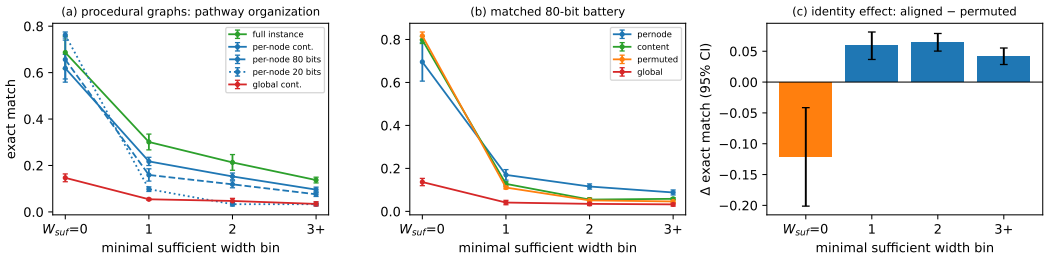

Figure 4: **Memory organization** on procedural graphs (5 seeds; error bars are standard deviations; panel (a) is the confirmation evaluation, 500 instances/bin; panels (b–c) use 750/bin). (a) Exact match by W_{suf} bin for each memory organization. (b) The matched 80-bit battery. (c) The identity effect (aligned − permuted) with 95% confidence intervals: negative at $W_{\text{suf}}{=}0$, positive on every bin with $W_{\text{suf}} \geq 1$.

Identity intervention. The aligned-versus-permuted comparison isolates stable node identity (Figure 4b–c; Table 4). The two variants use the same encoder, decoder, parameters, and bit budget. Permutation improves accuracy when $W_{\text{suf}}{=}0$ (0.817 vs 0.695), but aligned channels perform better in every bin with $W_{\text{suf}} \geq 1$ ($\Delta = +.059/+.065/+.042$ at $W_{\text{suf}}{=}1/2/{\geq}3$). A binomial interaction model places this change at the first bin that requires a prefix-derived label ($\beta = -2.22 \pm 0.06$ for the $\mathbf{1}[W_{\text{suf}}{\geq}1]$ interaction; the $\mathbf{1}[W_{\text{suf}}{\geq}2]$ term adds $+0.03 \pm 0.07$). The result is consistent with the need to retrieve prefix-derived state at the graph location where the completion uses it.

Width association. Across full-instance and node-indexed models, and after controlling for frontier mass, decisive-label delay, witness multiplicity, and target value, W_{suf} remains negatively associated with accuracy (coefficients -0.63 and -0.88; 95% cluster-bootstrap intervals exclude zero; estimation and evaluation perturbations disjoint).

6.6 Robustness and External Validity

The transition also appears under a minimum-cost dominance rule ($0.500 \to 0.630 \to 0.799 \to 0.996$) and on constant-weight instances (Figure 6, appendix). On procedural graphs, accuracy decreases with W_{suf} even without a bit bottleneck. Appendix D evaluates the same memory organizations on a real transit timetable—whose scheduled connections directly match our temporal-event representation—and finds the same ordering, while also showing that wide frontiers are uncommon in typical local transit slices.

7 Related Work

Communication and expressiveness limits. Loukas (2020a) and Loukas (2020b) derive capacity limits for message-passing GNNs via communication arguments, bounding what bounded-width and bounded-depth architectures can compute across a static graph's cuts. We bound a different object: an explicit one-way boundary between a temporal prefix and a later-revealed query, with an arbitrary prefix encoder, so the requirement is a property of the deferred task rather than of a message-passing architecture. Randomized one-way communication lower bounds (Kremer et al., 1995) provide the proof technique; our construction realizes a one-way communication problem within a verifier-backed temporal path task, and Fano arguments are standard (Cover & Thomas, 2006).

Neural algorithmic reasoning and diagnostics. This line studies which algorithms networks can execute and how supervision helps (Veličković et al., 2020; 2022; Ibarz et al., 2022; Dudzik & Veličković, 2022), varying the algorithm and instance size. Our benchmark adds a controlled axis for the reusable state an instance forces. Temporal graph benchmarks evaluate prediction (Huang et al., 2023; Yi et al., 2025) or diagnose temporal skills (Dizaji et al., 2025); we evaluate exact, verifier-backed computation with a derived difficulty measure.

Temporal graph memory. Temporal graph models maintain per-node memories updated by events (Kumar et al., 2019; Trivedi et al., 2019; Rossi et al., 2020); our node-indexed models are their minimal analogue. Our evaluation adds tasks with a controlled requirement, matched total bit budgets, an identity-destroying intervention, and stratification by W_{suf}.

Memory architectures, discrete bottlenecks, and exact algorithms. Memory-augmented and set networks introduce explicit storage, retrieval, and aggregation mechanisms (Graves et al., 2014; Weston et al., 2015; Graves et al., 2016; Santoro et al., 2018; Zaheer et al., 2017; Lee et al., 2019); TempoFrontier complements this line with tasks whose information and exact-label requirements can be calculated. Our annealed sign quantizer and the straight-through comparison sit in the discrete-representation line (Bengio et al., 2013; van den Oord et al., 2017; Huh et al., 2023). Exact temporal and multiobjective path algorithms (Wu et al., 2014; Dibbelt et al., 2018; Bazgan et al., 2025) provide the frontier concept and the exact states and solvers we use as references.

8 Limitations and Outlook

The formal lower bound applies to prefix-summary and one-pass streaming systems because the query is unavailable when the prefix state is constructed. Full-instance models do not satisfy this condition, and W_{suf} is used only as an empirical difficulty measure for them. In addition, W_{suf} depends on a declared completion family and bounds exact-label memories rather than arbitrary bit encodings, and the distributional bound applies to the full-uniform distribution rather than to held-out pattern pools, for which only a weaker support-size bound holds (Appendix A).

The learned experiments use small models and mostly constructed workloads. Several alternative encoders remain near the query-only baseline at larger k, while the positive scaling results rely on the node-keyed scan, which also reaches the baseline by $k{=}32$ under the fixed training protocol; the bit budget binds at evaluation, with training using a continuous surrogate. The real-transit study supports the memory-organization findings but also shows that wide frontiers are uncommon in local timetable slices. Future directions include program-conditioned objectives and adaptive memory allocation.

Reproducibility statement

Anonymized code and data accompany the submission and are available at `https://anonymous.4open.science/r/tempofrontier-56CE`. All results regenerate from committed configurations, versioned manifests, fixed hyperparameters, two cross-validated exact solvers, machine-checked generator invariants, and figure scripts that read raw logs. Figure 2a's held-out curves and Figure 4a report one-time confirmation evaluations generated before analysis; Appendix D reports the preregistered transit experiment. Anonymized code and data accompany the submission.

AI use statement

An LLM assisted with implementation, experiment management, and manuscript polishing under continuous author supervision. The authors designed the research questions and protocols, reviewed the code and verified all reported results, and take full responsibility for the paper.

References

Cristina Bazgan, Johannes Kager, Clemens Thielen, and Daniel Vanderpooten. A general label setting algorithm and tractability analysis for the multiobjective temporal shortest path problem. *Networks*, 2025.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432*, 2013.

Arnaud Casteigts, Anne-Sophie Himmel, Hendrik Molter, and Philipp Zschoche. Finding temporal paths under waiting time constraints. *Algorithmica*, 83, 2021.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2nd edition, 2006.

Julian Dibbelt, Thomas Pajor, Ben Strasser, and Dorothea Wagner. Connection scan algorithm. *ACM Journal of Experimental Algorithmics*, 23, 2018.

Alireza Dizaji et al. T-GRAB: A synthetic diagnostic benchmark for learning on temporal graphs. *arXiv:2507.10183*, 2025.

Andrew Dudzik and Petar Veličković. Graph neural networks are dynamic programmers. In *NeurIPS*, 2022.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv:1410.5401*, 2014.

Alex Graves et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538, 2016.

Shenyang Huang et al. Temporal graph benchmark for machine learning on temporal graphs. In *NeurIPS Datasets and Benchmarks*, 2023.

Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. In *ICML*, 2023.

Borja Ibarz et al. A generalist neural algorithmic learner. In *LoG*, 2022.

Ilan Kremer, Noam Nisan, and Dana Ron. On randomized one-round communication complexity. In *STOC*, 1995.

Ilan Kremer, Noam Nisan, and Dana Ron. Errata for: On randomized one-round communication complexity. *Computational Complexity*, 10(4), 2001.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *KDD*, 2019.

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 *ICML*, 2019.

Andreas Loukas. How hard is to distinguish graphs with graph neural networks? In *NeurIPS*, 2020a.

Andreas Loukas. What graph neural networks cannot learn: Depth vs width. In *ICLR*, 2020b.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv:2006.10637*, 2020.

Adam Santoro et al. Relational recurrent neural networks. In *NeurIPS*, 2018.

Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. DyRep: Learning representations over dynamic graphs. In *ICLR*, 2019.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017.

Petar Veličković, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. Neural execution of graph algorithms. In *ICLR*, 2020.

Petar Veličković et al. The clrs algorithmic reasoning benchmark. In *ICML*, 2022.

Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *ICLR*, 2015.

Huanhuan Wu, James Cheng, Silu Huang, Yiping Ke, Yi Lu, and Yanyan Xu. Path problems in temporal graphs. *PVLDB*, 7(9), 2014.

Zihao Yi et al. TGB-Seq: Challenging temporal GNNs with complex sequential dynamics. In *ICLR*, 2025.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets. In *NeurIPS*, 2017.

A FORMAL DEFINITIONS AND PROOFS

Quantity	Unit	Object	Binds which systems?
k (source entropy)	bits	prefix family/distribution	arbitrary bounded summaries
$W_{\mathrm{front}}(P)$	exact labels	one prefix	exact frontier algorithms
$W_{\mathrm{suf}}(P, \mathcal{S})$	exact labels	prefix + completion family	exact-label memories
b	bits	implemented message	the evaluated models

Table 2: State quantities used in the analysis. Information complexity is measured in bits over a prefix family; frontier quantities count exact labels for one prefix.

A.1 ASSUMPTIONS AND EXACT FRONTIER SCAN

Events are $e = (u, v, t, \lambda)$ with $t \geq 0$ and $\lambda \geq 1$; parallel events are allowed. Temporal walks satisfy endpoint continuity and $t(e_{i+1}) \geq t(e_i) + \lambda(e_i)$; waiting and repeated vertices are allowed. Equal label values (s, a) are merged.[1] In a public-coin prefix-summary protocol, every prefix-dependent value available after the boundary is included in a message with at most 2^b possible values.

If (s_1, a_1) dominates (s_2, a_2), every continuation feasible from the second label is feasible from the first and finishes with duration no larger, because both completions share the final arrival and $s_1 \geq s_2$. Induction over the chronological event order (the invariant: before each event, every realizable nondominated label using earlier events is present, or is dominated by a present value) proves the exactness of the frontier scan described in Section 2.

A.2 LOWER-BOUND CONSTRUCTION AND PROOF OF THEOREM 1

For $x \in \{0,1\}^k$, the prefix on $\{s, v, z\}$ contains a baseline label $(2k, 6k)$, label $(2k{+}2i,\ 6k{+}2i)$ when $x_i = 1$, and a dummy label $(2k{-}2i,\ 10k)$ when $x_i = 0$. The baseline dominates every dummy (at most one dummy is transiently nondominated before the baseline is processed), and no dummy catches completion $S_j = (v, z,\ 6k{+}2j{+}1,\ 2)$. The prefix frontier is therefore

$$F(v) = \{(2k, 6k)\} \cup \{(2k{+}2i,\ 6k{+}2i) : x_i = 1\},$$

an antichain of size $\|x\|_1 + 1$. Completion S_j is catchable by the baseline and the present branches $i \leq j$. With $i^\star = \max(\{0\} \cup \{i \leq j : x_i = 1\})$,

$$\mathrm{OPT}(x, j) = 4k + 2j + 3 - 2i^\star,$$

so $x_j = 1$ iff $\mathrm{OPT}(x, j) < 4k + 4$. Any protocol deciding this predicate gives a one-way INDEX protocol with the same message and error, and the randomized one-way communication complexity of INDEX is $\Omega(k)$ (Kremer et al., 1995; 2001), proving Theorem 1. The bound covers the total input-dependent state crossing the boundary; no single node is claimed to need $\Omega(k)$ bits, and full-instance models are outside its scope.

A.3 DISTRIBUTIONAL BOUNDS

Conditioning on public randomness,

$$I(X; M \mid R) \leq b, \qquad H(X \mid M, R) \geq k - b.$$

Subadditivity, per-coordinate binary Fano (the query index is independent of (X, M, R)), and concavity of H_2 give

$$k - b \leq \sum_i H_2(e_i) \leq k\, H_2(\bar{e}),$$

proving Proposition 1. If X is uniform on an arbitrary subset of size 2^{k-2} while J remains uniform on all k coordinates, only $k - 2 - b \leq k\, H_2(\bar{e})$ follows; replacing k by $k{-}2$ in the envelope is valid only when the query ranges over $k{-}2$ free coordinates. For the constant-weight class,

[1] Positive durations imply strictly increasing departure times along any feasible walk, so tie-processing order does not affect feasibility.

$H(X) = \log_2 \binom{k}{k/2} = k - O(\log k)$, yielding Corollary 2; coordinates are negatively correlated, so the bound is loose at small b (the query-only baseline remains $1/2$) and informative in the transition region.

A.4 Exact-label capacity

For each completion S, let O_S be the set of boundary labels through which its optimum is attainable. Then

$$W_{\text{suf}} = \min\{|L| : L \cap O_S \neq \varnothing \ \ \forall S \in \mathcal{S}\}$$

is a minimum hitting set, computed with one exact solve per (label, completion) pair. If $K{+}1$ labels are each uniquely optimal for some declared completion, any executor retaining at most K exact labels and no summary of discarded labels must fail on one of them. For the all-present lower-bound prefix, adding a completion catchable only by the baseline makes all $k{+}1$ labels uniquely necessary, so $W_{\text{front}} = W_{\text{suf}} = k + 1$.

B Data and evaluation protocols

Controlled families. The presence-fan replaces each lower-bound branch by a two-event path through a private intermediate vertex. Absent branches are dominated distractors, and random time shifts and vertex permutations preserve shape while removing fixed identifiers, so vertex count, event count, and degrees are identical across all 2^k presence patterns. Carrier instances add 12 vertices and 36 events under edge restrictions that prevent new paths into the branch, junction, or target states; exact per-instance checks verify unchanged frontiers and optima. The minimum-cost variant uses the same temporal skeleton with decreasing branch costs, producing a (d, a) antichain and the same decisive index. Presence patterns are disjoint across train and test for $k \geq 8$; smaller k values are used only as memorization-permitted controls because their pattern spaces are too small for stable held-out evaluation.

Procedural graphs. We generate fixed-shape temporal graphs with 10 vertices, 30 events, horizon 40, and durations 1–6, split after 60% of the event stream, retaining reachable instances whose optimum is not determined by the prefix alone. W_{suf} is computed exactly from the actual suffix and a finite family of time-jitter and event-drop perturbations; perturbations used to estimate width are disjoint from those used for evaluation, and deletion-tested width diverges from W_{suf} on 0.5% of instances. Outcome statistics by W_{suf} bin: 20–43 distinct duration classes; query-conditional majority shares $0.161/0.056/0.096$ for bins $0/1/2$.

W_{suf} sensitivity. Varying the declared completion family on $1{,}000$ fresh procedural instances:

Variation	Spearman vs. declared family	Bin migration
perturbation count $(2/8)$	0.92–0.95	4.4–6.9%
jitter magnitude $(\pm1/\pm6)$	0.98	2.8–3.8%
perturbation type / re-seed	0.92–0.98	2.8–6.5%

Mean W_{suf} increases monotonically as completions are added, as expected for a minimum over a growing family.

Machine validation. Two exact solvers agree on 15,000 random instances, the construction is exhaustively checked for $k \leq 6$, and approximately 30,000 generator assertions verify structural invariants.

Confirmation evaluation. After fixing all models, hyperparameters, analyses, and reported comparisons, we generated disjoint confirmation collections for the $k{=}8$ and $k{=}16$ presence-fan families, carrier embeddings, and procedural graphs. Figure 2a's held-out curves and Figure 4a report these one-time evaluations; all means fall within the corresponding development seed ranges.

C Models, training, and audit

Training. All learned models use $d = 64$, summary width 128, AdamW with learning rate 2×10^{-3}, OneCycle scheduling, weight decay 10^{-4}, gradient clipping at 1.0, and batch size 256. Gadget and

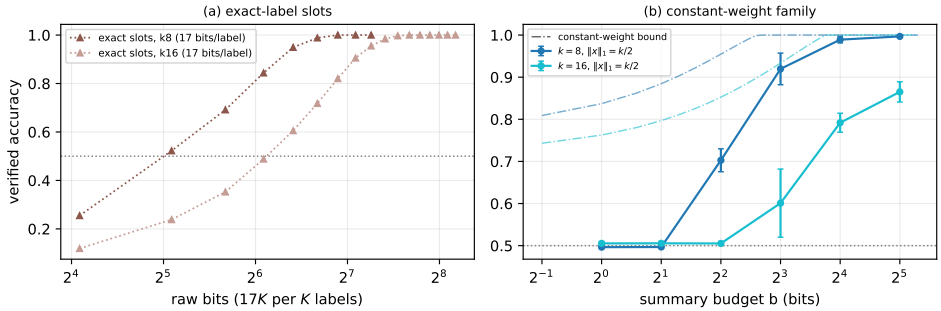

Figure 5: **Structured references.** Exact-label storage yields an addressable but bit-inefficient staircase (a); the constant-weight family retains the budget–accuracy transition when the number of present branches is fixed (b).

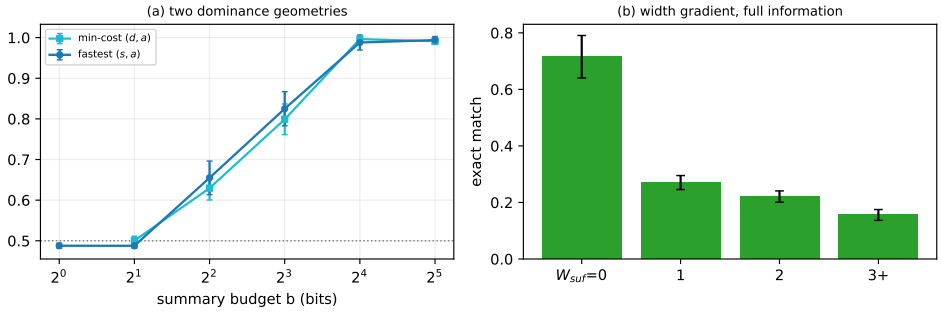

Figure 6: **Robustness beyond the lower-bound family.** The budget–accuracy transition persists under minimum-cost dominance (a), and full-instance accuracy decreases with W_{suf} on procedural graphs (b).

procedural models train for 4,000 and 8,000 steps, respectively, over five seeds. Hyperparameters were fixed before the replication and confirmation runs. Evaluation always uses a hard sign message; training uses a continuous annealed surrogate.

Quantizer optimization. A plain straight-through sign estimator remained near the query-only baseline even at $b = 2k$. Encoder probes and a continuous-bottleneck control showed that the prefix representation was informative, localizing the failure to the discrete optimization step. Layer normalization before the sign and soft-to-hard temperature annealing restored learning, reaching 0.997 at $b{=}2k$ and 0.764 at $b{=}k$. All reported inference uses hard bits.

Bottleneck audit. At evaluation, the decoder receives the hard message and suffix-public completion features. Prefix events, encoder states, masks, shapes, and instance-specific identifiers are unavailable after the boundary. A prefix-derived time normalizer discovered during auditing was removed before the replication campaign.

Interface	Stored-msg agreement	Constant msg	Permuted msg	Prefix destroyed
global, gadget ($b{=}16$)	1.000	0.422	0.341	1.000
node table, procedural (80 bits)	1.000	0.111	0.159	1.000
query-known ($b{=}16$)	1.000	0.257	0.356	1.000

The decoder receives only the declared message and suffix-public features: decoding is unchanged after deleting every prefix tensor, whereas replacing or permuting the message reduces accuracy to or below the query-only baseline.

Tested attention encoder. Under the shared protocol, the tested Set Transformer does not generalize to held-out presence patterns. A linear probe recovers the presence bits perfectly from its encoder

Encoder (k=16, theorem-aligned)	b=4	b=8	b=16	b=32
presence-vector MLP (reference)	0.57	0.70	**1.00**	1.00
event scan (node-keyed)	0.55	0.66	0.76	0.85
sequence GRU	0.40	0.35	0.36	0.34
Deep Sets	0.49	0.48	0.46	0.44

Table 3: Verified accuracy by encoder family (mean over seeds; query-only baseline 0.49). The reference MLP receives the true presence bits.

Variant (battery)	Total bits	Cells	Keys	Query at encode	Params
identity-aligned per-node	80	10	stable, identity	no	107k
hidden permutation	80	10	stable, hidden	no	107k
content-based read	80	10	learned retrieval	no	204k
global	80	1	—	no	200k
query-known (control)	80	1	—	yes	229k

Table 4: Matched 80-bit memory interfaces. The aligned and permuted node tables are parameter-identical; other read mechanisms use additional parameters.

representation, yet held-out accuracy remains below the query-only baseline after either a continuous or hard bottleneck. The failure therefore occurs in learning a reusable end-to-end code rather than in encoding the prefix facts. We treat this as a result for the tested configuration, not a general limitation of attention models.

Compute. Models contain 0.1–0.3 million parameters and train on 16 CPU cores in 1.5–7 minutes per configuration; no GPUs were used. The reported replication campaign used approximately 1,200 CPU-core-hours, and dataset generation and verification about two additional core-hours. Experiments use Python 3.12 and PyTorch 2.13, and all figures regenerate from recorded configurations and raw logs.

D REAL TRANSIT STUDY

To examine external validity, we apply the same memory organizations and width estimator to a real public-transit timetable, whose scheduled connections directly match the temporal-event representation of Section 2: the fastest-path frontier at a boundary is the Pareto profile maintained by profile-query algorithms in transit routing (Dibbelt et al., 2018). The protocol, model configurations, and evaluation criteria were fixed before the feed was downloaded (`experiments/PREREG_REALDATA.md`), with all outcomes reported.[2]

Item	Setting
feed	TriMet GTFS, busiest weekday (SHA-256 recorded)
scale	954,864 scheduled connections
instance	24 stops, 48 connections, 120-minute window
boundary	after 28 events
split	disjoint seeds and window–center pairs
models	full instance; global cont.; node cont.; node 20-bit; node 80-bit
evaluation	exact match by W_{suf} bin, 3 seeds

All evaluation bins contain 500 instances except $W_{\mathrm{suf}} \geq 3$, which contains 262 because wide-frontier slices are rare.

[2]Candidate-generation caps were introduced for feasibility before model evaluation; realized bin sizes are reported explicitly.

arm	$W_{\mathrm{suf}}{=}0$	1	2	≥ 3
full instance	$.880 \pm .026$	$.455 \pm .027$	$.475 \pm .008$	$.317 \pm .041$
global cont.	$.464 \pm .009$	$.132 \pm .007$	$.166 \pm .009$	$.135 \pm .005$
per-node cont.	$.757 \pm .025$	$.270 \pm .016$	$.316 \pm .014$	$.237 \pm .016$
per-node 20 bits	$.829 \pm .012$	$.171 \pm .013$	$.205 \pm .016$	$.177 \pm .013$
per-node 80 bits	$.827 \pm .025$	$.286 \pm .009$	$.323 \pm .012$	$.247 \pm .012$

Approximately 30% of accepted transit slices require at least one prefix-derived label, but only 4.3% have $W_{\mathrm{suf}} \geq 2$; wide local frontiers are therefore uncommon in this feed, and the preregistered criterion of at least 5% mass in every width bin was not met. The organization results nevertheless replicate. At matched capacity, node-indexed memory exceeds the global summary by 17.1 percentage points, and accuracy decreases with W_{suf} in the pooled regression ($\beta = -0.155$, $t = -9.4$). The full-instance model exceeds the 20-bit node memory by 14.0 points on $W_{\mathrm{suf}} \geq 3$ slices. Increasing node memory from 20 to 80 bits has little effect at $W_{\mathrm{suf}}{=}0$ but improves every bin requiring prefix-derived labels.