LMTE: Putting the "Reasoning" into WAN Traffic Engineering with Language Models

Xinyu Yuan†, Zonghui Wang†, Yan Qiao‡,✉, Meng Li‡ and Wenzhi Chen†,✉
†College of Computer Science and Technology, Zhejiang University
‡School of Computer Science and Information Engineering, Hefei University of Technology

***Abstract*—The rapid expansion of modern wide-area networks (WANs) has made traffic engineering (TE) increasingly challenging, as traditional solvers struggle to keep pace. Although existing offline ML-driven approaches accelerate TE optimization with deep neural networks (DNNs), they often lack sufficient expressiveness and generalization on unseen traffic patterns or topologies, limiting their practicality. Inspired by the success of large language models (LMs), for the first time, this paper investigates their potential as general-purpose traffic planners. Our contributions are two-fold: (i) Theoretically, we show that pre-trained LMs can simulate the sequential decision processes underlying TE and, crucially, exhibit parallel reasoning capabilities, making them well-suited for the task; (ii) Practically, we present LMTE, a novel LM-driven TE framework that embraces these insights through efficient multimodal alignment and lightweight configuration generation, all while preserving the model's original abilities. Extensive experiments demonstrate that LMTE matches top-tier performance on five datasets, achieving up to 15% better MLU and consistently lower performance degradation across diverse scenarios, e.g., less than 5% with high traffic dynamics and link failures. Moreover, it achieves 10 to 100 times speedups over traditional TE solvers. To aid future research, our code is publicly available at at `https://github.com/Y-debug-sys/LMTE`.**

***Index Terms*—traffic engineering, language models, machine learning, wide-area networks**

I. Introduction

Modern wide-area networks (WANs) interconnect geographically distributed data centers using high-capacity optical links, forming a critical yet expensive physical infrastructure [1]–[3]. To ensure high availability and low latency, traffic engineering (TE) plays a vital role in managing network performance. Typically planned by a centralized Software-Defined Networking (SDN) controller, TE periodically solves mathematical optimization problems to route traffic efficiently in the face of dynamic topology constraints and fluctuating service demands. This topic has been extensively studied across a variety of network environments [3]–[11].

After decades of research, there are still two major issues for conventional WAN TE relying on optimization tools like linear programming (LP): performance instability and computational complexity [3], [5]. Recently, the growing presence of machine learning (ML) provides a second option: leveraging historical demands to quickly output a good routing scheme for future conditions [12]. These ML-based algorithms train deep neural networks (DNNs) using reinforcement learning (RL) [9], [13] or end-to-end supervised learning [8], [14] to automatically infer WAN TE configurations, thereby eliminating the need for iterative optimization processes. Although promising, their imperfections can not be overlooked. As network environments evolve (e.g., due to topology changes or bursty traffic), the learned black-box mappings often fail to remain valid. This lack of generalization hinders the trustworthiness of their practical deployments. Consequently, offline-trained DNNs soon become outdated under new conditions, necessitating frequent retraining. Besides, it is not uncommon to encounter caveats such as the following statement, which reflects a persistent yet unresolved concern over their models' limited expressiveness:

> "Our realization of DOTE uses a relatively simple NN. · · · More expressive NN architectures could potentially lead to faster training and better quality solutions." - Perry [8].

To address these shortcomings, we now turn to a brand-new paradigm that capitalizes on the latest breakthroughs in the ML community—namely, large language models (LMs) such as ChatGPT [15] and LLaMA [16].

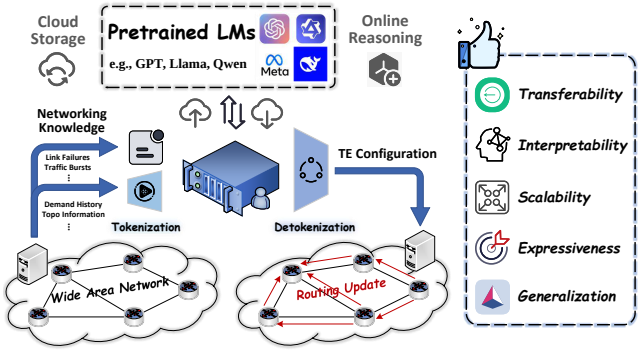


Fig. 1. Our ideal LM-driven TE system design for software-defined WANs.

LMs for WAN TE, Why? These powerful models, with billions of parameters pre-trained on massive text corpora, have achieved remarkable *generalization* capabilities in natural language processing (NLP). Encouragingly, recent work [17] has shown that LMs are *transferable* to networking tasks, including viewport prediction, video streaming, and cluster scheduling. These successes suggest that LMs may be a good fit for WAN TE as well. To move beyond the intuition, we give a formal theoretical justification for their applicability to TE. Our core insight is that LMs solve complex problems in a manner akin to human reasoning [18], [19], while TE falls into the category that benefits from such ability: it involves making sequential decisions, i.e., a solvable automaton (§III), despite its NP-hardness. This perspective lays the foundation

✉ Yan Qiao and Wenzhi Chen are co-corresponding authors.

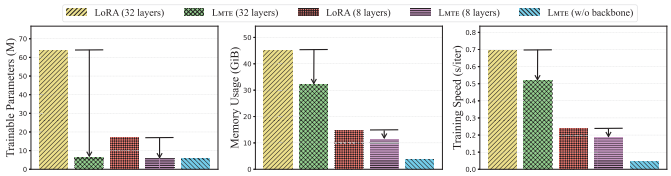


Fig. 2. Comparison of adaptation costs on a real-world topology: full-parameter fine-tuning of LLaMA-7B incurs high overhead, while LMTE is even more lightweight than LoRA by keeping the LM's backbone intact.

for delving into the usage of LMs (§IV). Specifically, we begin by investigating their *expressiveness* when applied to state-to-state transitions within TE automata. As it turns out, an off-the-shelf LM, when exposed to sufficiently diverse data, can approximate any mapping from an arbitrary initial state to the near-optimal TE configuration. Moreover, we provably show that LMs can simulate such mappings with only logarithmic depth. This capacity to abstract complex optimization procedures into conceptual leaps exemplifies *reasoning*, further underscoring the suitability of LMs for WAN TE.

LMs for WAN TE, How? At first glance, our theoretical setup appears tractable for a vanilla LM. Unfortunately, the modalities of TE system inputs, i.e., historical traffic and topology information, differ significantly from the NL supported by LMs [17]. In addition, simply fine-tuning LMs is prohibitively expensive for TE applications—not only requiring substantial computational resources, but also gradually impairing their ability to generalize. To overcome these challenges, we introduce LMTE (§V), the *first* framework designed to efficiently adapt LMs for TE optimization. LMTE enables LMs to understand both topology and traffic demands while retaining their pretrained knowledge. The key idea is to directly align inputs from multimodal encoders with prototype textual representations based on cross attention mechanism, progressively guiding the LM to interpret the task during adaptation (as visualized in §VI-E5). Further, we construct a domain-aware prompt template by preserving the LM's linguistic capabilities, to enhance *interpretability* and *robustness* under unseen cases. To support *scalability*, LMTE allocates router-wise demands individually through a shared head network. As illustrated in Fig. 1, our envisioned LM-driven TE system takes as input high-level instructions and real-time data to support on-the-fly reasoning. Note that most computation is offloaded to the cloud, with the backbone model kept fixed. Meanwhile, Fig. 2 shows LMTE achieves highly lightweight adaptation, e.g., only around 1% local trainable parameters in the 32-layer LLaMA-7B model, even compared to methods like LoRA [17].

We conduct a comprehensive evaluation of LMTE across diverse settings (§VI), including three real-world networks with publicly available traffic matrices and two large-scale topologies with synthetic data. Experimental results demonstrate that LMTE consistently outperforms state-of-the-art (SOTA) TE algorithms, particularly ML-based methods, achieving on average a 10% reduction in maximum link utilization (MLU). By leveraging the LM's pretrained knowledge and reasoning ability, LMTE generalizes well to a range of challenging and previously unseen scenarios. For instance, in comparison with FIGRET [14], a recently proposed robustness-enhanced method, LMTE surpasses it by up to 19%, 14%, and 21% in performance under network failures, abrupt traffic, and distributional drift, respectively, on GÉANT dataset. Furthermore, LMTE also offers efficiency gains—achieving over $10 \sim 100\times$ faster runtime than LP solvers on two large topologies.

II. Related Works & Preliminaries

Classical TE: (WAN) traffic engineering is a well-known and established topic whose fundamental goal is to control and manage networks efficiently, playing a critical role in service and cloud provider infrastructures [1], [3]–[11], [20]–[22]. Traditional TE solutions make routing decisions relying on linear programming techniques to achieve the optimal or sub-optimal performance [3]–[5], [7], [20]–[22]. Most existing algorithms fall into a category we refer to as prediction-based TE [1], [3], [4]. These approaches collect a set of sampled traffic matrices and compute routing plans based on them. As a result, when actual traffic deviates significantly from the samples, the computed routing may perform poorly. In contrast to adaptive TE, oblivious routing seeks to optimize worst-case performance guarantees across all possible demand matrices [7], [21], [22], thereby offering robust but highly sub-optimal performance for normal cases. Although several prior efforts have sought a middle ground between robustness and optimality [5], [6], solving the associated mathematical programs is still computationally expensive.

ML-Driven TE: Recent advances in deep learning have spurred extensive research into ML–driven solutions for traffic engineering problems [8], [9], [12]–[14], [23]–[25]. Not only is inference using neural models much faster than traditional models but also they offer the potential for autonomously managing complex networked systems. The first class is RL-based methods [9], [12], [13], [23]–[25], typically using demand-prediction and RL approaches. While widely adopted in present-day literature, this family of solutions typically incurs high training overhead and exhibits sensitivity to parameter choices. The second class employs end-to-end optimization [8], [11], [14], constituting a training strategy closely aligned with ours. Following the approach of DOTE [8], this class of schemes trains a simple decision model using historical traffic demands to directly produce SOTA configurations. Most recently, FIGRET [14] further enhanced the robustness by incorporating a regularization term. While they achieve runtimes several orders of magnitude faster than solving a linear program, (unlike LMTE) whether their DNNs can serve as generalizable decision-makers needs further investigation.

Automata Theory: A (deterministic) *automaton* is formally defined as a triple $\mathcal{A} := (\Sigma, \mathcal{Q}, \delta)$, where Σ represents a finite input alphabet (the set of allowed symbols), $\mathcal{Q}$ denotes a finite set of states, and $\delta : \mathcal{Q} \times \Sigma \to \mathcal{Q}$ is the total transition function that uniquely determines the next state for each current state and input symbol combination. $\mathcal{A}$ becomes an acceptor when equipped with an initial state $q_0 \in \mathcal{Q}$ and a set of accepting states. For any positive integer T and q_0, the automaton induces a mapping from input sequences $(\sigma_1, \ldots, \sigma_T) \in \Sigma^T$ to

state sequences $(q_1, \ldots, q_T) \in \mathcal{Q}^T$ defined by the recurrence relation $q_t := \delta(q_{t-1}, \sigma_t), \forall t \in [1, \ldots, T]$.

LMs Architecture: In this paper, we adopt the prevailing Transformer-based architecture for LMs, using a decoder-only structure. An l-layer Transformer [26] is a sequence-to-sequence network, consisting of alternating self-attention blocks and feedforward MLP blocks. Concretely, we consider:

$$\text{LM} = f_c \circ f_{mlp}^{(l)} \circ f_{attn}^{(l)} \circ \cdots \circ f_{mlp}^{(1)} \circ f_{attn}^{(1)} \circ \text{PE},$$

where $f_c : \mathbb{R}^{T\times D} \times \Theta_c \to \mathbb{R}^{T\times D}$ denotes classifier output layer and PE is positional embedding. The self-attention layer first projects an input $\boldsymbol{H} \in \mathbb{R}^{T\times D}$ into three subspaces, namely $\boldsymbol{Q} \in \mathbb{R}^{T\times D_q}$, $\boldsymbol{K} \in \mathbb{R}^{T\times D_k}$ and $\boldsymbol{V} \in \mathbb{R}^{T\times D_v}$. Then the output is calculated as $\boldsymbol{H}' = \text{SOFTMAX}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{D_k}}\right)\boldsymbol{V}$. Specifically, it calculates the dot product between each token pair after projection, and the softmax is applied row-wise.

III. TE Problem Setup: An Automata Perspective

To lay the groundwork for the discussions that follow, we begin with a standard formulation of the TE system, outlining its basic components and objectives. We then recast the iterative optimization problem through an automata-theoretic lens, offering a structured perspective on its sequential dynamics and decision-making processes.

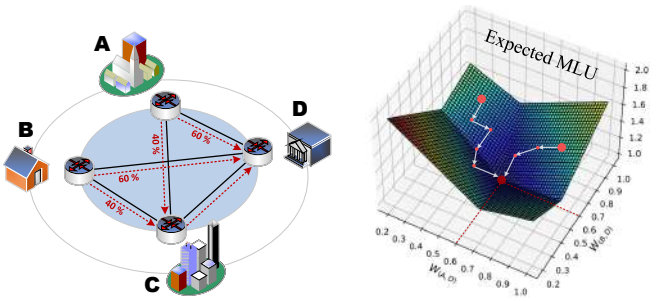
A. Problem Formulation

We model the network as a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, c)$ where vertices $\mathcal{V}$ represent network nodes (routers or switches), edges $\mathcal{E}$ correspond to directed communication links, and $c : \mathcal{E} \to \mathbb{R}^+$ assigns capacity to each link. Each origin-destination (OD) pair (s, t) communicates via *pre-selected* path sets $P_{s,t}$, called *tunnels*. The *traffic matrix* (TM) $\mathcal{D} = [\mathcal{D}_{i,j}] \in \mathbb{R}_{\geq 0}^{|\mathcal{V}|\times|\mathcal{V}|}$ provides a complete demand specification, where $\mathcal{D}_{i,j}$ represents the traffic demand of OD pair (i, j) per time interval. Now given measured TMs, a *traffic engineering configuration* $\mathcal{R}$ determines the mapping $\mathcal{D}_{s,t} \to \{r_p\}_{p\in P_{s,t}}$ where r_p is the split ratio on tunnel p for next time interval. Note that the split ratios should satisfy $\sum_{p\in P_{s,t}} r_p = 1.0$.

This paper focuses on maximum link utilization (MLU) [8], [22], [27], a classical TE objective, which refers to the maximum value of all link utilization ratios in the network. A corresponding TE configuration can be obtained by solving the following mathematical program:

$$\begin{aligned} \textbf{minimize} \quad & \alpha = \max_{e\in\mathcal{E}} \frac{f_e}{c_e} \\ \textbf{subject to} \quad & f_e = \sum_{s,t\in V} \sum_{p\in P_{s,t}} \sum_{e\ni p} \mathcal{D}_{s,t} \cdot r_p, \quad \forall e \in \mathcal{E} \\ & r_p \geq 0, \quad \forall s, t \in \mathcal{V},\ \forall p \in P_{s,t} \\ & \sum_{p\in P_{s,t}} r_p = 1.0, \quad \forall s, t \in \mathcal{V} \end{aligned} \tag{1}$$

We here illustrate the case using a example network [8], as depicted in Fig. 3(a), where all link capacities are set to 1. At fixed time intervals, the TE system determines traffic splitting ratios for origin nodes A and B, distributing their demands across two tunnels to destination D. The demands of A and B are independently drawn at each interval from a fixed distribution: with probability $1/2$, A's demand is $5/3$ and B's is $5/6$; otherwise, A demands $5/6$ and B demands $5/3$. Detailed in [8], the optimal split ratios for (A, D) and (B, D) are achieved at the red point (0.6, 0.6) marked in Fig. 3(b). Importantly, this subfigure further shows that the (expected) MLU is ***convex*** with respect to the split ratios—a desirable property that is not only *commonly assumed* in TE optimization problems [5], [8], but also forms the cornerstone of the problem transformation we introduce next.



(a) Toy network topology (b) MLU function of the splitting

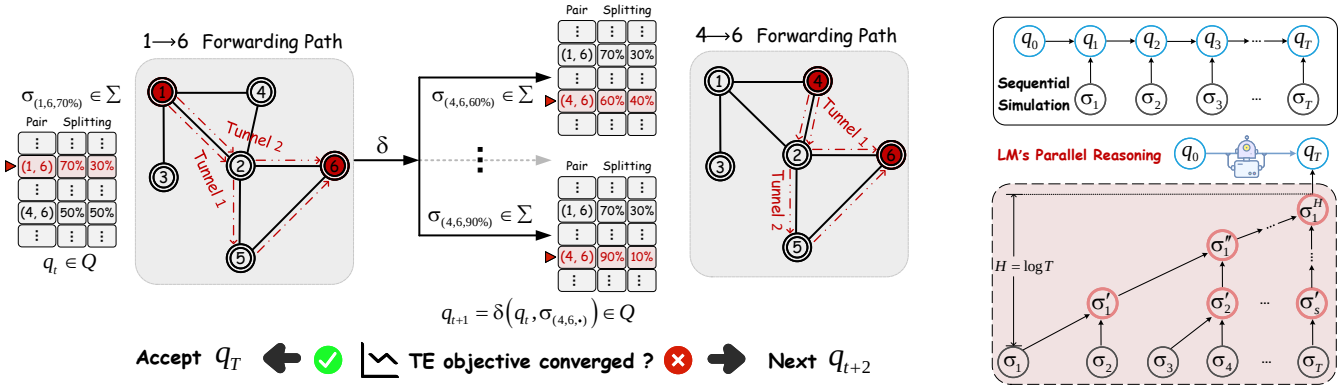
Fig. 3. Illustrative network traffic engineering (TE) example [8].

B. Tailored Problem

In particular, the TE function in Fig. 3 suggests a discrete optimization trajectory: starting from arbitrary split ratios, the configuration is iteratively adjusted along the curve of (expected) MLU until convergence to a global minimum. This step-by-step adjustment naturally aligns with the concept of a decision chain where each action incrementally improves the TE outcome. Motivated by the observations, we can model TE configuration selection as an automaton $\mathcal{A}$. The state space $\mathcal{Q}$ is defined as all possible solutions to the TE problem, i.e., the collection of split ratios for the communication tunnels. The input alphabet Σ comprises two components: candidate OD pairs and their updated tunnel weights. Let $q_0 \in \mathcal{Q}$ be the WAN condition-dependent initial state. As illustrated in Fig. 4(a), a simple $\mathcal{A}$ processes an input $\sigma_t \in \Sigma$ and transitions to state q_{t+1} by optimizing the selected pair(s)'s allocation at each step t, e.g., from (1, 6) to (4, 6). The automaton halts at state q_T once the improvement in MLU falls below a predefined threshold, thereby yielding the final TE configuration. The objective of TE is thus to compute q_T, the terminal state of $\mathcal{A}$ after T transitions. We then prove it can be modeled in *finite-length*:

Lemma 1. *(TE as Solvable Automata) WAN traffic engineering is NP-hard, but any instance admits a solution through finite-step automaton simulation.*

Proof Sketch. We start by reducing the Two-Commodity Integral Flow (D2CIF) problem: *determine whether two sources can simultaneously send a given amount of integral flow without exceeding edge capacities*, which is known to be NP-Complete [28], to the problem of interest. Set up the TE MLU problem with total demand and minimize $\alpha = \max_e \frac{f_e}{c_e}$; then



(a) An automata-style iterative optimization of the TE objective

(b) TE automata simulation

Fig. 4. Theoretical intuition and foundations for our LM-driven TE approach. **(a)** Simple TE example from a solvable automaton perspective (Lemma 1). **(b)** A pretrained LM can approximate any optimal state from an initial state (Theorem 2), and accelerate automaton simulation with logarithmic-depth (Theorem 3).

deciding whether $\alpha \leq 1$ is equivalent to solving D2CIF. Since MLU also optimizes congestion beyond feasibility, it is at least as hard as D2CIF. Now, our key concern is whether there exists a finite-length automaton state transition path that yields an optimal solution. Let state $\pi \in \mathcal{R}$ be a mapping result from the finite set of all possible history realizations to a TE configuration. By leveraging the convexity of the TE objectives, the proof implies a stochastic gradient descent (SGD [29])-style transition rule. Formally, the state transition is defined as: $\pi_t = \delta(\pi_{t-1}, \sigma_t) := \text{Proj}\{\pi_{t-1} - \eta v_t\}$, where $v_t \in \partial\mathcal{L}(\pi_t, \mathcal{D}_t)$ is a subgradient of the MLU function, $\text{Proj}\{\cdot\}$ denotes the projection onto the simplex for each OD pair, and η is the step size. Then according to Theorem 1 of [8], we have: For any target accuracy $\varepsilon > 0$, there exist a $\eta > 0$ and a finite iteration horizon K such that $\left|\tilde{\mathcal{L}}\left(\frac{1}{K}\sum_{k=1}^{K}\pi_k\right) - \tilde{\mathcal{L}}(\pi^*)\right| \leq \varepsilon$. Therefore, the automaton's states π_t converge ergodically around the optimal TE state π^* via finite simulations, concluding the proof. □

The first result, Lemma 1, shows that our automaton-based formulation is generally solvable. However, due to the NP-hardness of the underlying mathematical program, solving it via sequential optimization becomes computationally prohibitive at scale, as illustrated at the top of Fig. 4(b). This underscores the need for decision models with sufficient expressive capacity to support practical deployment.

IV. Harnessing LMs for TE: Theoretical Insights

We next investigate pre-trained language models, known for their strong generalization and emergent reasoning capabilities [18], as a promising decision model to TE. This section provides theoretical insights into their applicability to our automaton-based formulation.

A. LMs Have Enough Expressiveness

We start by posing a fundamental question: can LMs discover and reach the optimal state in WAN TE? This hinges on whether LMs are expressive enough to capture and simulate the underlying state transitions. To explore this, we consider the TE state q_i of the automaton as compositional sequence vector like $\underbrace{u_1\ v_1\ c[u_1][v_1], \ldots, d[u_1][v_1], \ldots}_{\text{input conditions \& historical demands (prefix)}}, \underbrace{r_1[u_1][v_1], \ldots}_{\text{split ratios}}$, analogous to the natural language token. With this representation, we now state the following theorem:

Theorem 2. *(LMs as Powerful Simulators) For any $\epsilon > 0$ and state-to-state function f, we can always find an LM h satisfying $d(f(\mathbf{x}), h(\mathbf{x})) \leq \epsilon$, where d denotes a discrepancy over states, typically induced by a function norm $\|f - h\|_p$.*

Proof Sketch. Adapted from Yun et al. [30], the proof of Theorem 2 goes in three steps: (i) Approximate state-to-state mappings with step functions on compact domain. A step function is piecewise constant and can approximate continuous functions by leveraging Universal Approximation Theorem [31]. Particularly, for any f, there exists piece-wise constant function $\tilde{f}$ such that $d(f, \tilde{f}) \leq \epsilon/3$. (ii) Approximate piece-wise constant functions with hardmax-based Transformers. Using hardmax operator simplifies the construction of piece-wise constant functions. According to Proposition 4 of [30], for all $\tilde{f}$, there exists the modified Transformer g such that $d(\tilde{f}, g) \leq C_{\tilde{f}}$, where $C_{\tilde{f}}$ is a constant associated with $\tilde{f}$. (iii) Approximate hardmax-based Transformers with original Transformers. Due to softmax approximating hardmax (see Lemma 4 of [32]), we also have that the modified Transformer can be approximated with error $\leq \epsilon/3$. Finally, by choosing $C_{\tilde{f}} \leq \epsilon/3$ and applying the triangle inequality, we obtain $d(f, h) \leq d(f, \tilde{f}) + d(\tilde{f}, g) + d(g, h) \leq \epsilon$. □

Theorem 2 implies that, in terms of expressiveness, LMs can simulate general TE automata from q_0 to q_T, by implicitly encoding δ, $\sigma_{1:T}$ within their parameters, given convergence training on corpora of sufficient richness. Nevertheless, as indicated by [11], trivially deep architecture (e.g., recurrent adjustments with $\mathcal{O}(T)$-depth and appropriate nonlinearities) may still be required to obtain the near-optimal TE configuration. This raises the question of whether LMs can leverage their "reasoning" power to circumvent the inefficiency.

B. LMs Bring "Reasoning" to TE

To this end, we highlight the parallel computation capabilities of LMs, by extending results from prior works [32], [33].

Specifically, any parallelizable TE automaton can be simulated through a tree-structured process (Fig. 4(b), bottom), enabling all states to be computed within just $\mathcal{O}(\log T)$ layers. This resolves the earlier question and demonstrates how LMs can perform non-sequential, compositional inference—hallmarks of *reasoning*. Theorem 3 formalizes this key insight, showing that LMs can solve the TE problem with a high probability under logarithmic depth. Taken together with Theorem 2, these results establish the fundamental soundness of LMs for TE.

Theorem 3. *(LMs Find TE Shortcuts) An LM with feedforward MLP dimension $D = \mathcal{O}\left(|\mathcal{Q}|^2\right)$, attention dimension $D_{(\cdot)} = \mathcal{O}\left(|\mathcal{Q}|\right)$ and depth logarithmic in T can perform continuous simulation of any TE automaton $\mathcal{A} = (\Sigma, \mathcal{Q}, \delta)$ at length T with probability at least $1 - 2/T^2$.*

Proof. At a high level, the proof of Theorem 3 consists of two parts: the composed transition chain $\delta(\cdot, \sigma_T) \circ \cdots \circ \delta(\cdot, \sigma_1)$ can be evaluated in $\mathcal{O}\left(\log T\right)$-depth via a binary tree, and the shortcuts can be constructed with a high probability. The first part is a direct application of Theorem 1 of [32]. We here focus on proving the second part. Without loss of generality, we assume the TE state is a 1-dimensional vector for clarity.

Define $S = \mathcal{O}\left(\log T\right) \geq 1$, sequential state vector $u \in \mathbb{R}^T$, random matrix $R \in \mathbb{R}^{S \times T}$ in which $R_{i,j} \sim \mathcal{N}\left(0, 1\right)$, and shortcut state vector $v \in \mathbb{R}^S$ in which $v_i = \frac{1}{\sqrt{S}} \sum_j R_{i,j} u_j$. Then let $x = \frac{\sqrt{S}}{\|u\|_2} v$ i.e. $x_i = \frac{R_i^T u}{\|u\|_2}$, we have $\|x\|_2^2 = \sum_{i=1}^S x_i^2 = \frac{S\|v\|_2^2}{\|u\|_2^2}$ and $x \sim \mathcal{N}\left(0, I_S\right)$. By Markov's inequality i.e. $\Pr\left[x \geq a\right] \leq \frac{\mathbb{E}(x)}{a}$, for any $0 < \epsilon < 1$ and $0 < \lambda < 0.5$

$$\Pr\left[\|v\|_2^2 \geq (1+\epsilon)\|u\|_2^2\right] = \Pr\left[\frac{\|u\|_2^2}{S} \cdot \|x\|_2^2 \geq (1+\epsilon)\|u\|_2^2\right]$$

$$= \Pr\left[e^{\lambda\|x\|_2^2} \geq e^{(1+\epsilon)\lambda S}\right] \leq \frac{\mathbb{E}\left[e^{\lambda\|x\|_2^2}\right]}{e^{(1+\epsilon)\lambda S}} = \left[\frac{\mathbb{E}\left(e^{\lambda x_i^2}\right)}{e^{(1+\epsilon)\lambda S}}\right]^S$$

Using MGF of χ^2 distribution i.e. $\mathbb{E}\left(e^{\lambda\chi_1^2}\right) = \frac{1}{\sqrt{1-2\lambda}}$, and setting $\lambda = \frac{\epsilon}{2(1+\epsilon)}$, we can write

$$\left[\frac{\mathbb{E}\left(e^{\lambda x_i^2}\right)}{e^{(1+\epsilon)\lambda S}}\right]^S \leq \left[\frac{1}{\sqrt{1-2\lambda} \cdot e^{(1+\epsilon)\lambda}}\right]^S = e^{S(\log(1+\epsilon)-\epsilon)/2}$$

Now using the inequality $\log(1+a) < a - a^2/2 + a^3/3$, and selecting an appropriate $S = \mathcal{O}\left(\log T\right)$,

$$\Pr\left[\|v\|_2^2 \geq (1+\epsilon)\|u\|_2^2\right] \leq e^{-2\log T} = \frac{1}{T^2}$$

Similarly, by setting $\lambda = \frac{\epsilon}{2(1-\epsilon)}$, we get

$$\Pr\left[\|v\|_2^2 \leq (1-\epsilon)\|u\|_2^2\right] = \Pr\left[e^{-\lambda\|x\|_2^2} \geq e^{-(1-\epsilon)\lambda S}\right]$$

$$\leq \frac{\mathbb{E}\left(e^{-\lambda\|x\|_2^2}\right)}{e^{-(1-\epsilon)\lambda S}} \leq \left[\frac{1}{\sqrt{1+2\lambda} \cdot e^{(1-\epsilon)\lambda}}\right]^S = e^{S(\log(1-\epsilon)-\epsilon)/2}$$

Using the inequality $\log(1-a) < -a - a^2/2$ and selecting an appropriate $S = \mathcal{O}\left(\log T\right)$,

$$\Pr\left[\|v\|_2^2 \leq (1-\epsilon)\|u\|_2^2\right] \leq e^{-2\log T} = \frac{1}{T^2}$$

Combining the above, we can obtain

$$\Pr\left[(1-\epsilon)\|u\|_2^2 \leq \|v\|_2^2 \leq (1+\epsilon)\|u\|_2^2\right] > 1 - \frac{2}{T^2} \quad (2)$$

Finally, by setting $u = X_i - X_j$ and $v = f(X_i) - f(X_j)$, where $X_i, X_j \in \mathbb{R}^T$ and $f : \mathbb{R}^T \to \mathbb{R}^S$ is the shortcut mapping, Eqn. 2 implies that pairwise distances are preserved under the mapping with high probability (close to 1 even for modest T). The model can thus simulate state transitions in a compressed space of dimension $\mathcal{O}(\log T)$, completing our proof. □

V. LM-Driven TE Framework: LMTE

The previous section identifies LMs as compelling candidates for the decision-making tasks of interest. However, certain aspects of the theoretical analysis are difficult to implement directly within LMs, e.g., constructions of the (latent) TE automata. The central challenge is how to bridge the modality gap between lengthy symbolic specifications and pretrained LM inputs and outputs. While adaptation is a simplest fix, it can be resource-intensive, potentially at the expense of generalization, as mentioned in §I. Below, we elaborate on the design of LMTE—a **L**anguage **M**odel-driven **T**raffic **E**ngineering framework—developed to address the dilemma, including cross-modal alignment (§V-A, §V-B), prompting support (§V-C), and efficient adaptation (§V-D).

A. Invariant Spatial-Temporal Embedding

LMTE first encodes the multimodal inputs of TE problems into token-like embeddings, using a GNN encoder for network topology and an RNN encoder for historical TMs. As our formulation (§III) requires preconfigured tunnel information as an input, we explicitly form these paths using features generated by the GNN encoder [11]. Specifically, the node features are aggregated over each link and concatenated with the corresponding capacity to model edge features. For each tunnel, the features of all traversed edges are then concatenated to construct a tunnel-level representation[1]. For example, the embedding of tunnel 1→2→4→3 is $\left[E^{1,2}; E^{2,4}; E^{4,3}\right]$, where $E^{i,j}$ denotes the edge feature connecting nodes #i and #j. In doing so, LMTE remains robust under various WAN changes (e.g., links, tunnels, or capacities). We refer to the property as *state invariance*, where embedding$\left(\boldsymbol{T} \circ \boldsymbol{X}\right) = \boldsymbol{T} \circ$ embedding$\left(\boldsymbol{X}\right)$ for an arbitrary transformation $\boldsymbol{T}$ and WAN state $\boldsymbol{X}$. Fig. 5(a) summarizes the embedding pipeline. To unify the structural and temporal representations, we then reproject the temporal embeddings $\boldsymbol{R}^{\mathcal{D}} \in \mathbb{R}^{S \times D}$ using a learned matrix $\mathbf{W}$ derived from the tunnel embeddings $\boldsymbol{R}^{\mathcal{T}} \in \mathbb{R}^{P \times L}$, where S denotes the historical window size and P the number of tunnels. The resulting embedding is computed as

$$\boldsymbol{R}^{\mathcal{F}}_{s,c} = \sum_d \boldsymbol{R}^{\mathcal{D}}_{s,d} \cdot \mathbf{W}_{d,c}, \text{ with } \mathbf{W} = \phi\left(f_\theta\left(\left[\boldsymbol{R}^{\mathcal{T}}_1; \ldots; \boldsymbol{R}^{\mathcal{T}}_P\right]\right)\right).$$

[1]Zero-padding is used to standardize all embeddings to a fixed length L.

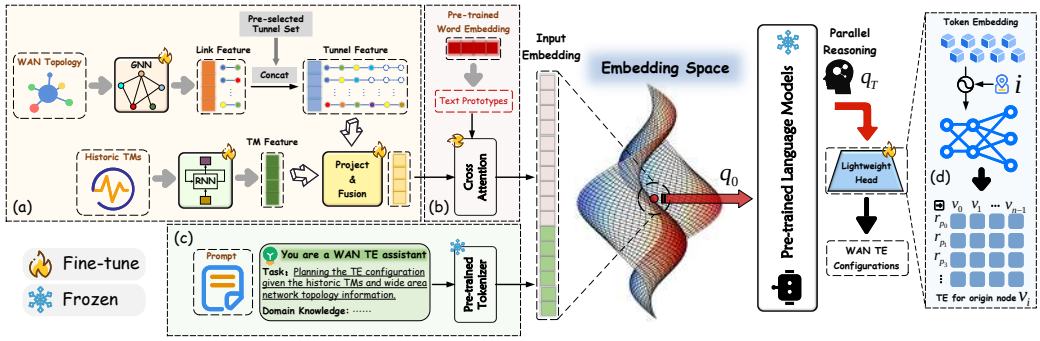


Fig. 5. An overview of the proposed LMTE, the first LM–driven WAN TE framework. **Left:** input embedding alignment to bridge the modality gap. **Middle:** a frozen LM backbone, reused without any modification. **Right:** a lightweight LM head that jointly outputs TE configurations.

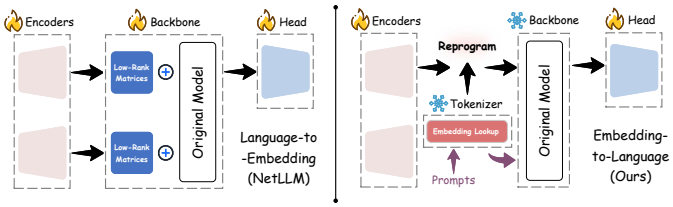


Fig. 6. Illustration of our embedding-to-language alignment in comparison of language-to-embedding alignment (e.g., LoRA used by NETLLM [17]).

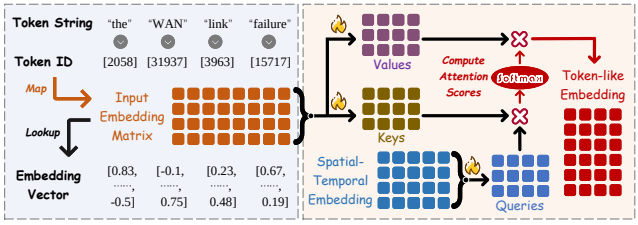


Fig. 7. Reprogramming spatial-temporal representations using original input embeddings of LMs. **(left)** The tokenizer maps and lookups token embedding vectors. **(right)** We obtain token-like embeddings via cross attention.

Here, $f_\theta(\cdot)$ denotes a neural transformation (e.g., an MLP) applied to the concatenated tunnel features, and $\phi(\cdot)$ reshapes the output into $\mathbf{W}$. This mechanism enables the input model to align temporal dynamics with structural context by conditioning the projection on the underlying topology. An additional advantage of this design is that, since the output dimension C is tunable via f_θ, the fused representation $\boldsymbol{R}^{\mathcal{F}} \in \mathbb{R}^{S\times C}$ is much more compact than a naïve stack of the two modalities.

B. Embedding-to-Language Alignment

Now, the framework must align tokenized representations understood by LMs with the input ones. A common language-to-embedding approach is LoRA [17], which injects trainable low-rank matrices into selected layers of the backbone. Compared to full-parameter fine-tuning, this strategy is technically feasible for bridging heterogeneous data modalities in task-specific settings. However, as it still modifies the backbone parameters, it may gradually forget previously acquired knowledge, diminishing generalization—an issue known as catastrophic forgetting [34]. Instead, LMTE adopts an embedding-to-language approach, leaving the backbone unchanged while offering even greater cost-efficiency (see Fig. 2). The core idea is to reprogram the spatial-temporal embeddings into the source NL space [35], as illustrated in Fig. 6. Before delving into the details, we first clarify how input tokens are embedded in an LM: each token is mapped to a unique identifier, which is then used to retrieve a vector from a learned input embedding matrix. The pipeline is summarized on the left of Fig. 7. Since this matrix inherently serves as a set of *text prototypes* that ground the LM's understanding, our alignment mechanism then reduces to a linear lookup over these pre-trained word embeddings, guided by the WAN information.

As shown on the right of Fig. 7, LMTE employs a cross attention to realize the embedding-to-language alignment. Concretely, $\boldsymbol{R}^{\mathcal{F}}$ is linearly projected to the query $\mathbf{W}_k^Q \boldsymbol{R}^{\mathcal{F}} \in \mathbb{R}^{n_q\times d_k}$, for a attention head k. The same operation applies to the key $\mathbf{W}_k^K \boldsymbol{R}^{\mathcal{N}} \in \mathbb{R}^{n_k\times d_k}$ and value $\mathbf{W}_k^V \boldsymbol{R}^{\mathcal{N}} \in \mathbb{R}^{n_k\times d_v}$, where $\boldsymbol{R}^{\mathcal{N}}$ denotes the result of a linear projection to the input embedding matrix. We then calculate the cross-attention as

$$\boldsymbol{R}_k^{\mathcal{A}} = \text{SOFTMAX}\left(\frac{\left(\mathbf{W}_k^Q \boldsymbol{R}^{\mathcal{F}}\right)\left(\mathbf{W}_k^K \boldsymbol{R}^{\mathcal{N}}\right)^\top}{\sqrt{d_k}}\right)\mathbf{W}_k^V \boldsymbol{R}^{\mathcal{N}}.$$

When multi-head operation is applied, the outputs of each head are concatenated. The underlying rationale of the above procedure is to enable each spatial-temporal representation to query a shared bank of linguistic prototypes, so that LMTE can naturally seek interpretation of the task in the NL space.

C. Task-Specific Prompting

Another notable strength of LMTE over existing LM-based networking methods (i.e., NETLLM [17]) lies in its ability to maintain LMs' prompt input compatibility (see Fig. 6). This is particularly beneficial in our task-specific settings, where well-crafted prompts function as a vital interface between expert intent and model behavior. An example is shown in Fig. 5(c). On one hand, these prompts contribute to knowledge activation and task grounding. For instance, they can specify the LM's role (e.g., "You are a WAN traffic engineer"), and include TE-specific context such as descriptions of the WAN or statistics of historical TMs. On the other hand, they provide a flexible mechanism for encoding ad hoc constraints, such as network changes or potential traffic pattern. Consider the case of link failures, by incorporating failure-related descriptions into the prompt (e.g., "Link 1–3 is currently down"), LMTE is able to adapt its output accordingly without retraining. The task-specific prompting thus not only improves interpretability but also enhances the robustness of LMTE, enabling flexible responses to unseen and dynamic conditions in volatile WAN environments. The overall prompt structure is shown in Fig. 8.

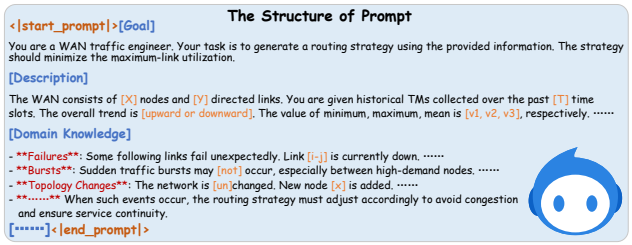


Fig. 8. The overall structure of the task-specific prompt used by LMTE.

Why not include all tunnels directly in the prompt? A WAN with over 120 nodes and 8 tunnels per OD pair yields >110K entries—exceeding LM's maximum input limits.

D. Lightweight End-to-End Adaptation

In this part, we present how to obtain globally optimal allocations and fine-tune the model of LMTE. The first challenge here is LMTE's scalability to keep pace with the growing size of the WAN. For a topology with thousands of nodes, the solution space can span millions of dimensions, resulting in an overwhelming number of trainable parameters. To break the "curse of dimensionality", inspired by multi-agent policy [9], LMTE generates split ratios independently via a shared, lightweight prediction head. As illustrated in Fig. 5(d), parameters are shared across origin nodes (or routers), which is specified to the projector using the Transformer sinusoidal positional embedding [26]. This design reduces the neural network's computational complexity from $\mathcal{O}(|\mathcal{V}|^2)$ to $\mathcal{O}(|\mathcal{V}|)$, substantially lowering the memory overhead. Regarding model adaptation, we adopt an end-to-end direct optimization strategy. Similar to DOTE [8], the LM-driven system tries to infer fine-grained configurations for $\mathcal{D}_t^i$ from the recent sequence $\{\mathcal{D}_{t-1}^i, \mathcal{D}_{t-2}^i, \ldots, \mathcal{D}_{t-H-1}^i\}$, topology $\mathcal{G}(\mathcal{V}, \mathcal{E}, c)$, and its corresponding tunnels. After computing the final softmax outputs for all OD pairs, the system evaluates the network-wide MLU loss in Eqn. 1, whose gradients are computable for any past realization [8]. LMTE then updates its trainable parameters using the gradients of the expected MLU computed over each batch of data points. Once adapted, the alignment and reasoning of LMs can be done automatically. It is worth mentioning that only the lightweight encoding and head networks need to be deployed locally, while the *frozen* backbone can be hosted remotely (as in Fig. 1) for resource-constrained environments.

VI. Evaluation

In this section, we begin by describing the experimental setup in §VI-A, followed by a comparison of LMTE with SOTA TE methods in §VI-B. We then evaluate its performance under various conditions (i.e., link failures, traffic bursts, and natural drift) in §VI-C, and its overhead in §VI-D. Finally, additional experiments & visualizations are presented in §VI-E.

A. Experimental Setup

1) Datasets: We consider three real-world WAN topologies: US Internet2 Network [36] (*Abilene*, *12* nodes, *30* edges), China Education and Research Network [37] (*CERNET*, *14* nodes, *32* edges) and Pan-European Research Network [38] (*GÉANT*, *23* nodes, *74* edges), each with traffic matrices captured at different snapshots. We also select two larger WAN topologies (*Cogentco*, *197* nodes, *486* edges and *UsCarrier*, *158* nodes, *378* edges) from the Internet Topology Zoo [39] and generate *non-stationary* synthetic TM sequences via a *gravity model* [40]. Specifically, we model them by combining trend, sinusoidal (periodic) variations, and noise. Note that we adopt a *7:1:2* split ratio for training, validation, and testing. We choose *12* as the history window size for the input TMs.

2) Baselines: We compare LMTE against the following baselines: ① two state-of-the-art ML-driven approaches: *DOTE* [8] and *FIGRET* [14], which make centralized decisions based on historical demand observations; and ② three Optimization-based approaches: *COPE* [5], *Oblivious Routing* [22], and *Gurobi-Pred*, where Gurobi (v11.0.3) [41] is employed to optimize predictions obtained by weighted moving average. Note that all the above baselines use the same set of candidate paths. Regarding the tunnel selection, we employed 4 shortest paths for experiments on the two larger topologies. For others, we employed 8 shortest paths as the default setting, consistent with previous research [8].

3) Implementing LMTE*:* We implement LMTE using PyTorch and PyTorch Geometric, and conduct experiments on a 16-core virtual machine equipped with NVIDIA RTX 4090 24GB GPUs. Unless otherwise specified, we utilize LLaMA-3-8B with the first 8 layers as the default backbone. For the RNN encoder, we adopt a bidirectional RNN architecture; for the GNN encoder, we adopt a Graph Transformer architecture. To improve the model's semantic understanding during adaptation, we randomly introduce link failures[2] and traffic bursts with a probability of 10%. Additionally, the shared output projection is parallelized to accelerate the process by prioritizing tensor calculations over for loops. All experiments are repeated $3 \sim 5$ times, and we report the averaged results.

B. TE Solution Quality

Fig. 9 compares the normal-case quality of TE configurations between LMTE and other TE schemes on five topologies. The y-axis value represents the maximum link utilization. We observe that ① **ML-driven approaches:** The two DNN-based schemes, DOTE and FIGRET, are reaching near-optimal average performance among existing methods. However, their results exhibit a trade-off as the WAN scale increases. Furthermore, on the smaller CERNET topology, their behavior appears notably anomalous. Given the more dynamic network patterns in CERNET, it is plausible that these DNNs have merely overfitted to the historical data, thus indicating a lack of generalization ability. ② **Optimization-based approaches:** (Semi-)oblivious schemes inevitably incur high link utilization even under non-worst-case scenarios. More critically, algorithms like COPE and Oblivious Routing become impractical for large-scale topologies (e.g., those exceeding 120 nodes) due to their prohibitive memory requirements. While Gurobi-Pred performs well in environments with temporally stable

[2]Note that the selected links do not disrupt any OD-pair communication.

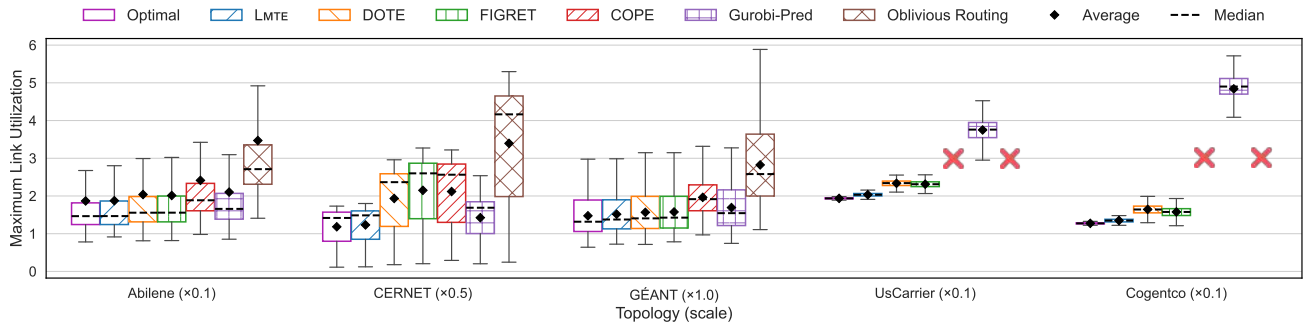
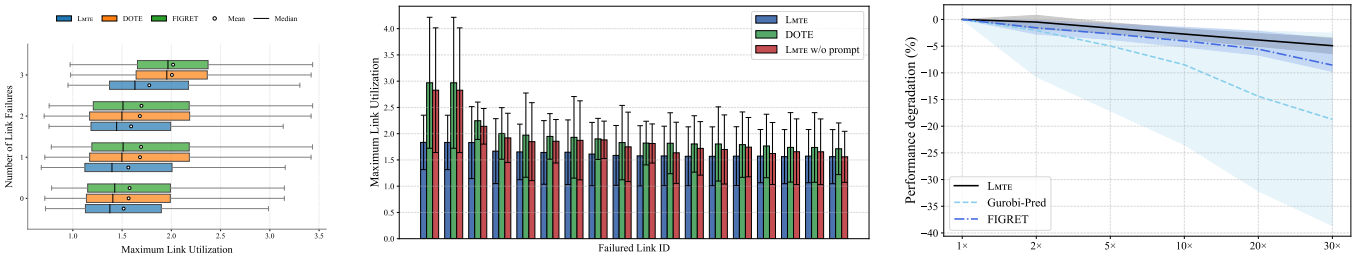


Fig. 9. TE quality of LMTE and baselines under the objective of minimizing maximum link utilization (MLU). The x-axis shows different topologies with their normalization scales, where MLU is adjusted by the scale factor. Red crosses indicate cases where no result is available due to computation failure.



(a) Reacting to random link failures (b) MLU under different single failure scenarios (c) Performance degradation with flow fluctuations

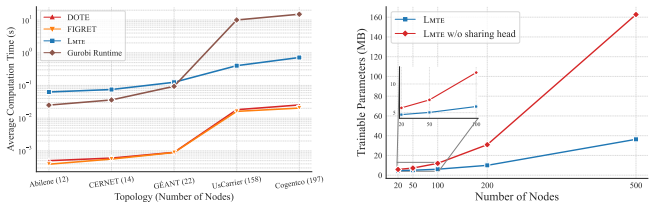
Fig. 10. Comparing the performance of LMTE and baselines on GÉANT under challenging testing scenarios with traffic bursts and link failures.

traffic, its effectiveness degrades substantially in the presence of non-stationary demands. ③ **LM-driven approach (LMTE):** Our proposed LMTE, grounded in language models, consistently achieves the lowest MLU across all topologies, outperforming both DNN- and optimization-based methods in average and median performance. For instance, compared to the best TE scheme on large and complex WANs, LMTE reduces the average MLU by $8\% \sim 15\%$. Benefiting from the pre-trained reasoning and abstraction capabilities of LMs, LMTE adapts effectively to diverse traffic patterns and topology scales without overfitting to specific network instances. These results underscore the potential of LMs as a powerful foundation for real-world, large-scale WAN TE.

C. Performance under Unseen Scenarios

1) Coping with Network Failures: Beyond prompt guidance, LMTE handles different link failures using simple heuristics that proportionally redistribute affected traffic across available paths, introducing minimal computational overhead. The same strategy is applied to the baselines. Fig. 10(a) compares LMTE, DOTE, and FIGRET under varying numbers of link failures on the GÉANT topology. While all methods maintain relatively stable performance, LMTE shows superior robustness, with the performance gap widening as failures increase. To further investigate, we simulate single-link failures on the sorted 18 most critical links. As shown in Fig. 10(b), LMTE's average MLU remains between 1.6 and 1.75, closely matching its optimal performance in Figure 9. The figure also highlights that prompt-guided LMTE performs especially well under the most severe failure scenarios.

2) Robustness to Demand Changes: To evaluate LMTE's robustness under drastically changing demand, we introduce temporal fluctuations to the test TMs, by following a similar approach to [9]. For each OD-pair, we compute the variance of its demand over time and scale it by factors of 2, 5, 10, 20, and 30 to define zero-mean normal distributions. Samples drawn from these distributions are then added to each demand at every time slot. Results are shown in Fig. 10(c). Compared to FIGRET, which is specifically designed to handle such scenarios, LMTE achieves comparable or even greater stability. The performance does not decrease by more than 5%. In contrast, prediction-based TE schemes suffer a significant degradation in MLU relative to the other two ML-based methods.



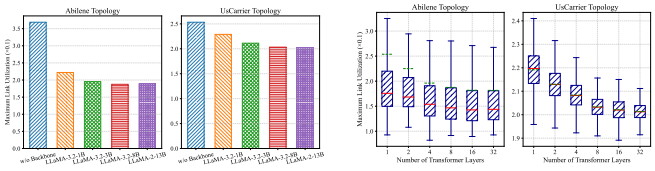
(a) Computation Time vs. WAN Scale (b) Model Size vs. WAN Scale

Fig. 11. Comparison of computation complexity across different WAN scales.

TABLE I
PERFORMANCE DECLINE WITH NATURAL DISTRIBUTION DRIFT.

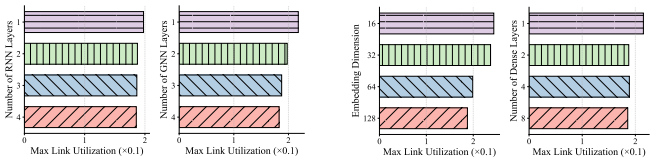
Topology	Scheme	Training traffic matrices time segments		
		0%∼25%	25%∼50%	50%∼75%
Abilene	LMTE	$6.9\% \pm 0.50\%$	$4.7\% \pm 0.46\%$	$3.8\% \pm 0.81\%$
	FIGRET	$10.5\% \pm 0.53\%$	$7.6\% \pm 0.49\%$	$8.5\% \pm 0.76\%$
CERNET	LMTE	$7.8\% \pm 0.84\%$	$6.5\% \pm 0.56\%$	$3.2\% \pm 0.51\%$
	FIGRET	$10.6\% \pm 1.08\%$	$9.1\% \pm 0.57\%$	$16.7\% \pm 1.55\%$
GÉANT	LMTE	$15.3\% \pm 1.36\%$	$16.7\% \pm 1.60\%$	$2.5\% \pm 0.18\%$
	FIGRET	$20.8\% \pm 0.61\%$	$30.6\% \pm 0.73\%$	$5.8\% \pm 0.34\%$

3) Handling Natural Distribution Drift: We test the impact of natural train-test distribution shifts on LMTE by fine-tuning with progressively earlier TM segments (0∼25%, 25∼50%,



(a) Performance of different backbones (b) Performance of different layers

Fig. 12. Exploring the importance of backbone and layers of pre-trained LMs.



(a) Tokenization (input) module (b) Detokenization (output) module

Fig. 13. Sensitivity analysis of LMTE's hyperparameters on Abilene.

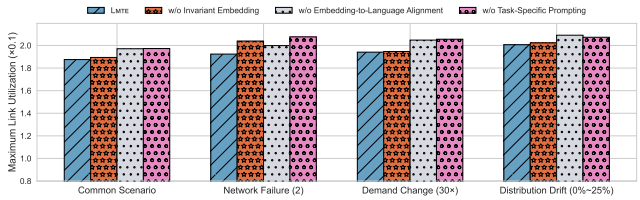


Fig. 14. Ablation study of LMTE's three key designs on Abilene.

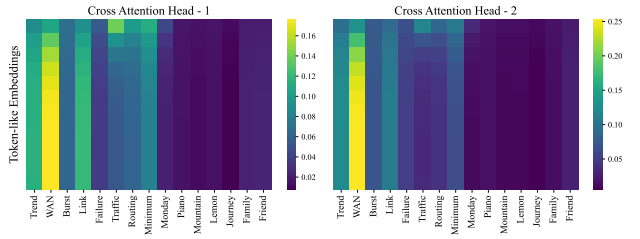


Fig. 15. Learned cross-attention maps from the input alignment module.

and 50∼75%), validating on 75∼85%, and testing on the final 15%. This setup highlights the performance drop compared to training on the first 75% of the TM. A detailed comparison with FIGRET is presented in Tab. I. LMTE shows strong resilience, with less than 10% degradation even when only one-third of the traffic data is used for adaptation—except on the GÉANT dataset, where the data distribution is highly imbalanced. The increased volatility observed in FIGRET underscores the advantage of pretrained language models in few-shot adaptation and generalization.

D. Computation Complexity

Fig. 11(a) compares the runtime of LMTE against other ML-based methods and Gurobi. DOTE and FIGRET, both relying on lightweight MLP architectures, achieve faster inference than LMTE. However, LMTE still delivers over an order-of-magnitude speedup compared to Gurobi on large topologies, suggesting its runtime remains practical for real-world deployment. For instance, loading a 7B LM like LLaMA-2-7B takes about 0.1s $\sim$ 0.3s to generate one answer, and it takes about 0.04s for smaller LMs such as OPT-1.3B to generate one answer [17]. This provides ample time for measurement and follow-up actions within each TE cycle. We also report the total number of tunable parameters in Fig. 11(b). As the number of nodes increases, the memory required by LMTE decreases substantially, owing to our shared LM head design.

E. Additional Results & Discussions

1) Impact of LMs' Parameter Scales: To assess the applicability of LMTE across different model scales, we evaluate it on three additional LMs beyond 3.2-8B: 3.2-1B, 3.2-3B, and 2-13B. A control variant without any backbone is also included to examine the LM's standalone reasoning capability. As shown in Fig. 12(a), larger models generally yield better performance, with diminishing returns beyond 8B. Moreover, all adapted LMs outperform the control variant on both topologies, demonstrating the effectiveness of leveraging LMs.

2) Impact of LMs' Layers: To measure the impacts of LMs' Transformer layers on the WAN TE performance, we conduct an experiment across varying layer configurations. In Fig. 12(b), we note the findings: (i) more layers generally lead to better performance, but the gains plateau beyond a certain depth, and (ii) as the topology size increases, the optimal number of layers tends to be larger, though typically no more than ten layers are required. Overall, this is consistent with our theoretical findings (see Theorem 3).

3) Sensitivity Analysis: We further analyze the sensitivity of LMTE's performance to key hyperparameters. Specifically, Fig. 13(a) examines the impact of varying the number of RNN and GNN layers, while Fig. 13(b) explores the effects of changing the number and dimension of dense layers.

4) Ablational Study: Fig. 14 presents an ablation study evaluating the impact of LMTE's key components (i.e., embedding in §V-A, alignment in §V-B and prompting in §V-C) on its overall performance. We find that each component of the proposed framework contributes meaningfully to enhancing its TE quality, either in one scenario or across multiple scenarios.

5) Representation Interpretation: We next visualize the cross-attention maps from two randomly selected attention heads in the embedding-to-language alignment module of LMTE in Fig. 15, based on the GÉANT dataset. Each row corresponds to a random spatio-temporal input instance, and each column denotes a chosen token embedding, covering both TE-relevant terms (e.g., "WAN", "Link") and TE-irrelevant ones (e.g., "Piano", "Friend"). The heatmaps reveal strong alignment between input features and semantically relevant tokens, suggesting that the module bridges the gap between heterogeneous modalities. Our proposal is thus interpretable.

VII. Conclusion

This work provided an initial yet comprehensive exploration of pre-trained language models for WAN TE, aiming to bring advanced reasoning capabilities to the next choice of high-quality configurations both in theory and in practice. We developed LMTE, the first LM-driven TE framework that combines carefully designed, highly interpretable components. LMTE produces first-rate routing decisions while offering superior generalization compared to SOTA TE approaches. Though not the final answer, LMTE marks a meaningful step toward sustainable networking solutions powered by LMs.

REFERENCES

[1] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[2] L. Poutievski, O. Mashayekhi, J. Ong, A. Singh, M. Tariq, R. Wang, J. Zhang, V. Beauregard, P. Conner, S. Gribble *et al.*, "Jupiter evolving: transforming google's datacenter network via optical circuit switches and software-defined networking," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 66–85.

[3] F. Abuzaid, S. Kandula, B. Arzani, I. Menache, M. Zaharia, and P. Bailis, "Contracting wide-area network topologies to solve flow problems quickly," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, 2021, pp. 175–200.

[4] D. Narayanan, F. Kazhamiaka, F. Abuzaid, P. Kraft, A. Agrawal, S. Kandula, S. Boyd, and M. Zaharia, "Solving large-scale granular resource allocation problems efficiently with pop," in *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, 2021, pp. 521–537.

[5] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, "Cope: Traffic engineering in dynamic networks," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, 2006, pp. 99–110.

[6] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C. L. Lim, and R. Soulé, "{Semi-oblivious} traffic engineering: The road not taken," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, 2018, pp. 157–170.

[7] D. Applegate and E. Cohen, "Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 2003, pp. 313–324.

[8] Y. Perry, F. V. Frujeri, C. Hoch, S. Kandula, I. Menache, M. Schapira, and A. Tamar, "{DOTE}: Rethinking (predictive){WAN} traffic engineering," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023, pp. 1557–1581. [Online]. Available: https://github.com/PredWanTE/DOTE

[9] Z. Xu, F. Y. Yan, R. Singh, J. T. Chiu, A. M. Rush, and M. Yu, "Teal: Learning-accelerated optimization of wan traffic engineering," in *Proceedings of the ACM SIGCOMM 2023 Conference*, 2023, pp. 378–393.

[10] Y. Wang and Z. Wang, "Explicit routing algorithms for internet traffic engineering," in *Proceedings Eight International Conference on Computer Communications and Networks*. IEEE, 1999, pp. 582–588.

[11] A. A. AlQiam, Y. Yao, Z. Wang, S. S. Ahuja, Y. Zhang, S. G. Rao, B. Ribeiro, and M. Tawarmalani, "Transferable neural wan te for changing topologies," in *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024, pp. 86–102.

[12] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar, "Learning to route," in *Proceedings of the 16th ACM workshop on hot topics in networks*, 2017, pp. 185–191.

[13] O. Hope and E. Yoneki, "Gddr: Gnn-based data-driven routing," in *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 517–527.

[14] X. Liu, S. Zhao, Y. Cui, and X. Wang, "Figret: Fine-grained robustness-enhanced traffic engineering," in *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024, pp. 117–135. [Online]. Available: https://github.com/FIGRET/figret

[15] OpenAI, "Chatgpt," 2024.

[16] AI@Meta, "Llama 3 model card," 2024.

[17] D. Wu, X. Wang, Y. Qiao, Z. Wang, J. Jiang, S. Cui, and F. Wang, "Netllm: Adapting large language models for networking," in *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024, pp. 661–678.

[18] T. Webb, K. J. Holyoak, and H. Lu, "Emergent analogical reasoning in large language models," *Nature Human Behaviour*, vol. 7, no. 9, pp. 1526–1541, 2023.

[19] J. González and A. Nori, "Does reasoning emerge? examining the probabilities of causation in large language models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 117 737–117 761, 2024.

[20] A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. C. Zermeno, C. S. Gunn, J. Ai, B. Carlin, M. Amarandei-Stavila *et al.*, "Bwe: Flexible, hierarchical bandwidth allocation for wan distributed computing," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 1–14.

[21] H. Racke, "Minimizing congestion in general networks," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* IEEE, 2002, pp. 43–52.

[22] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Racke, "Optimal oblivious routing in polynomial time," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, 2003, pp. 383–388.

[23] Z. Xu, J. Tang, J. Meng, W. Zhang, Y. Wang, C. H. Liu, and D. Yang, "Experience-driven networking: A deep reinforcement learning based approach," in *IEEE INFOCOM 2018-IEEE conference on computer communications*. IEEE, 2018, pp. 1871–1879.

[24] J. Zhang, M. Ye, Z. Guo, C.-Y. Yen, and H. J. Chao, "Cfr-rl: Traffic engineering with reinforcement learning in sdn," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2249–2259, 2020.

[25] G. Bernárdez, J. Suárez-Varela, A. López, B. Wu, S. Xiao, X. Cheng, P. Barlet-Ros, and A. Cabellos-Aparicio, "Is machine learning ready for traffic engineering optimization?" in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*. IEEE, 2021, pp. 1–11.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[27] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 253–264, 2005.

[28] S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *16th annual symposium on foundations of computer science*. IEEE, 1975, pp. 184–193.

[29] A. Shapiro, D. Dentcheva, and A. Ruszczynski, *Lectures on stochastic programming: modeling and theory*. SIAM, 2021.

[30] C. Yun, S. Bhojanapalli, A. S. Rawat, S. Reddi, and S. Kumar, "Are transformers universal approximators of sequence-to-sequence functions?" in *The Eighth International Conference on Learning Representations*, 2020.

[31] A. Pinkus, "Approximation theory of the mlp model in neural networks," *Acta numerica*, vol. 8, pp. 143–195, 1999.

[32] B. Liu, J. T. Ash, S. Goel, A. Krishnamurthy, and C. Zhang, "Transformers learn shortcuts to automata," in *The Eleventh International Conference on Learning Representations*, 2023.

[33] C. Sanford, D. Hsu, and M. Telgarsky, "Transformers, parallel computation, and logarithmic depth," in *Proceedings of the 41st International Conference on Machine Learning*, 2024, pp. 43 276–43 327.

[34] X. Wang, T. Chen, Q. Ge, H. Xia, R. Bao, R. Zheng, Q. Zhang, T. Gui, and X. Huang, "Orthogonal subspace learning for language model continual learning," in *The 2023 Conference on Empirical Methods in Natural Language Processing*.

[35] M. Jin, S. Wang, L. Ma, Z. Chu, J. Y. Zhang, X. Shi, P.-Y. Chen, Y. Liang, Y.-F. Li, S. Pan *et al.*, "Time-llm: Time series forecasting by reprogramming large language models," in *The Twelfth International Conference on Learning Representations*, 2024.

[36] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan, "Network anomography," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, 2005, pp. 30–30. [Online]. Available: https://www.cs.utexas.edu/~yzhang/research/AbileneTM/

[37] Y. Qiao, K. Wu, and X. Yuan, "Autotomo: Learning-based traffic estimator incorporating network tomography," *IEEE/ACM Transactions on Networking*, 2024. [Online]. Available: https://github.com/duoduoqiao/AutoTomo/tree/main/Data/CERNET

[38] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 83–86, 2006. [Online]. Available: https://totem.info.ucl.ac.be/dataset.html

[39] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011. [Online]. Available: http://www.topology-zoo.org/

[40] M. Roughan, A. Greenberg, C. Kalmanek, M. Rumsewicz, J. Yates, and Y. Zhang, "Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, 2002, pp. 91–92.

[41] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com