DIVIDE, HARMONIZE, THEN CONQUER IT: SHOOTING MULTI-COMMODITY FLOW PROBLEMS WITH MULTIMODAL LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

The multi-commodity flow (MCF) problem is a fundamental topic in network flow and combinatorial optimization, with broad applications in transportation, communication, and logistics, etc. Nowadays, the rapid expansion of allocation systems has posed challenges for existing optimization engines in balancing optimality and tractability. In this paper, we present PRAM, the *first* ML-based method that leverages the reasoning power of multimodal language models (MLMs) for addressing the trade-off dilemma—a key requirement of service providers. As part of our proposal, PRAM (i) quickly computes allocations by *dividing* the original problem into independent subproblems, and (ii) ensures global consistency by *harmonizing* these subproblem "agents" via a multi-agent reinforcement learning (RL) algorithm. Theoretically, we show that PRAM, which learns to perform gradient descent in context, provably converges to the optimum within the family of MCF problems. Empirically, on real-world datasets and public benchmarks, PRAM approaches even outperforms state-of-the-art linear programming (LP) solver (very close to the optimal solution), and substantially lower runtimes (one to two orders of magnitude faster). These results establish PRAM as a principled alternative that enables near-optimal and rapid decision making for general MCF problems. The anonymous codebase is available at https://anonymous.4open.science/r/PRAM, with experimental datasets attached in the supplementary material.

1 INTRODUCTION

Suppose we want to send multiple commodities from their sources to their respective destinations along the arcs of an underlying network, with the objectives of achieving low link utilization, high throughput, and fairness among commodities. This scenario results in what we call a multi-commodity flow (MCF) problem (Assad, 1978). Its importance has been underscored by wide-ranging applications in transportation, communication, logistics, energy, and cloud computing (Schrijver, 2002; Wang & Wang, 1999; Balcik et al., 2014; Blaauwbroek et al., 2015; Chang et al., 2010). Over the past decades, optimization-based algorithms built on linear programming (LP) played an important role in solving such problem, which is guaranteed for computing near-optimal solutions (Khachiyan, 1980; Karmarkar, 1984; Chen & Ye, 2024). However, the conventional wisdom in the community is that solving these LPs often takes too long for large-scale solution space (Cohen et al., 2021), while modern networks often comprise hundreds of nodes and thousands of links, handling millions of commodities with unpredictable demands (Applegate & Cohen, 2003; Wang et al., 2006). More recently, the advancements of machine learning (ML) have prompted extensive research into ML-based algorithms for MCF problems (Valadarsky et al., 2017; Bernárdez et al., 2021). While faster, these methods, particularly reinforcement learning (RL), suffer from sensitivity to uneseen environments and rely on fragile optimization with extensive hyperparameter tuning (Henderson et al., 2018), leaving scalability and uncertainty challenges largely unresolved.

To address these limitations, we take a different perspective. Rather than relying solely on monolithic optimization, our key idea is to scale with the growing network size by decomposing the original problem into smaller subproblems through the partition of topology and demands (Abuzaid et al., 2021; Cohen et al., 2021), and directly solving them in parallel through deep neural networks (DNNs) given historical data. Building on this idea, we propose **PRAM**: **P**artitioned **R**esource **A**llocation with **M**ultimodal Language Models (MLMs). First, we divide the MCF problem by commodity flow

Figure 1: **Various real-world examples of multi-commodity flow.** From left to right: wide-area network traffic engineering, urban mobility management, delivery route optimization, regional power dispatch, and tenant-aware flow control, all of which involve a very large solution space nowadays.

sources, then allocates flows for each subset individually using a shared "agent" model. We here choose MLMs as the agent backbone for two major reasons: ① they exhibit emergent abilities that were not explicitly programmed into them during pre-training on massive data, such as mathematical problem reasoning and generalization to unforeseen conditions (Zhang et al., 2024; Wu et al., 2024a); and ② they mitigate the costs of retraining and handcrafting specialized DNNs for complex inputs, by processing topology and demand data as images and text tokens respectively. Second, we present a novel multi-agent reinforcement learning (RL) algorithm for fine-tuning the MLM backbone using counterfactual policy gradients (Foerster et al., 2018), with lightweight communication enabled by trainable low-rank matrices and context. This allows each agent to estimate its individual contribution to the global objective. Further, we show that if the MCF objective satisfies suitable convexity/concavity property, once adapted, PRAM provably attains the optimum by simulating gradient descent (GD) internally. Therefore, the effectiveness of PRAM is theoretically sound. Our evaluation encompasses topologies of different scales, heterogeneous demand distributions, and multiple MCF objectives. In comparison with a variety of solutions, including RL, LP and heuristics, the main results show that: ① PRAM *achieves near-optimal performance.* It consistently outperforms previous RL-based allocation scheme, with an average performance gap of less than 8% from the optimal solution. ② PRAM *accelerates flow allocation at scale.* It speeds 10× to 100× faster than solving LPs on large-scale topologies. ③ PRAM *generalizes well to new environment.* It exhibits strong robustness to demand distributions, flow dynamics and network failures. Through ablation studies and visualizations, we delve deeper into the factors contributing to PRAM's effectiveness.

In conclusion, we regard our exploration of MLM-based MCF optimization as an initial yet groundbreaking step, and we discuss the limitations of PRAM that we hope future research can address.

2 PROBLEM FORMULATION

We formulate the multi-commodity flow (MCF) problem on a directed graph (or topology) $\mathcal{G}(\mathcal{V}, \mathcal{E}, c)$, where $\mathcal{V}$ is the set of vertices (or nodes), $\mathcal{E}$ the set of directed edges (or links), and $c : \mathcal{E} \to \mathbb{R}^+$ specifies link capacities. Each commodity (s, t) is associated with a *predefined* set of candidate paths $P_{s,t}$ and *historical* demand $\mathcal{D}_{s,t}$. A configuration π distributes each commodity flow using weights r_p. The objective is to determine near-optimal π under different objectives, e.g., minimizing maximum link utilization for resilience, maximizing total flow for profit or concurrent flow for fairness (see Appendix B.1 for details). This problem is known to be NP-complete (Even et al., 1975) and finds many important applications across various domains as listed in Figure 1.

3 PRAM: PARTITIONED RESOURCE ALLOCATION WITH MLMS

Complex MCF problems are often intractable as a whole but can be decomposed into subproblems defined over subsets of commodities and links. We propose to leverage MLMs for solving these subproblems in parallel, exploiting their mathematical reasoning capacity to yield high-quality allocations. We call this framework Partitioned Resource Allocation with MLMs (or PRAM for short). In the rest of this section, we describe the motivations, benefits, and designs of PRAM.

3.1 MOTIVATIONS BEHIND PRAM

Achilles' Heel of LP-based Methods The MCF problem is typically formulated as a mathematical program, and in theory an optimal solution can always be obtained (or infeasibility certified) using LPs. In practice, however, solving these programs can be prohibitively expensive: the worst-case complexity of LP with d variables is about $\mathcal{O}(d^{2.3729})$ (Lee & Sidford, 2015; Cohen et al., 2021; Narayanan et al., 2021), and modern systems may involve millions of variables (e.g., at least one

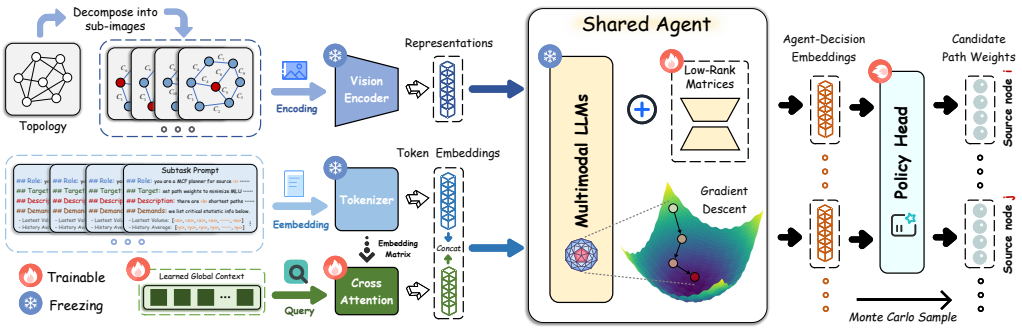


Figure 3: **Overview of PRAM.** It consists of three core components: *partition* module to divide task into smaller sub-tasks, *MLM-based agent* module to to generate sub-task-specific answers, and *adaptation* module to efficiently learn global knowledge for MCF optimization.

per source–destination pair), resulting in extremely long runtimes (e.g., in hours). Even worse, the optimal solution assumes full knowledge of all current commodities, yet real-time measurement at scale is often impractical. As a result, inputs are typically derived from historical flows or their predictions. But given the uncertain and unpredictable nature of data such as network traffic (Wang et al., 2006; Perry et al., 2023), the final performance achieved is often highly sub-optimal.

Where ML-based Methods Fall Short Recent studies have sought to bypass the iterative optimization process using machine learning, where deep neural networks (DNNs) directly determine routing decisions. Although promising, existing ML-based methods fall short in several key limitations, as we list in the following. ① *High engineering costs.* Their success is heavily dependent on training and engineering DNN models, such as graph neural networks (GNNs) (Wu et al., 2020; Bernárdez et al., 2021), for the target scenarios, which, however, can be labor-intensive due to the complex structures. ② *Poor generalization.* DNNs trained on specific data distributions/environments may not perform well on unseen ones. ③ *Curse of dimensionality.* Since representing the flow allocation needs $\mathcal{O}\left(|\mathcal{V}|^2\right)$ path weights, the scaling challenge remains. For instance, in a topology of 1,000 nodes with 4 candidate paths, the output layer alone would require up to 4 million dimensions.

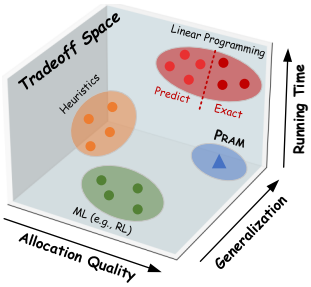


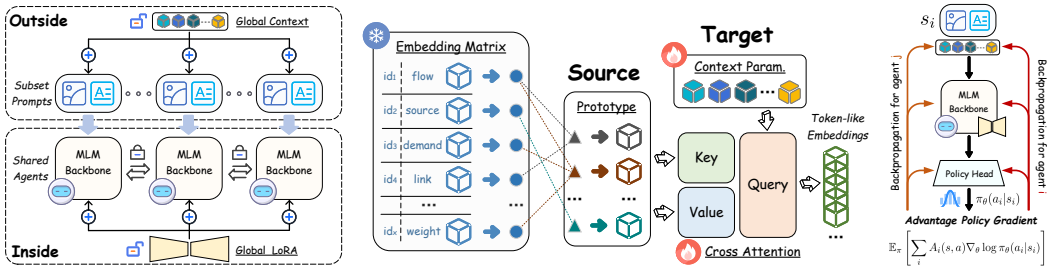
Figure 2: Tradeoff space.

Faced with the above dilemma, an intuitive way of accelerating allocation and reducing complexity is to decompose MCF optimization into sub-tasks, applying solver(s) simultaneously in each subproblem and merging their results at the end. For example, dividing commodities and demands evenly among k sub-problems will reduce the number of variables in each sub-problem by k^2 (Cohen et al., 2021). Holding the premise, we argue that with adequate generalization and judicious design, ML can also have the potential to attain near-optimal practical performance. This is exemplified by the recent popular pretrained (multimodal) large language models, opening new opportunities that allow PRAM to be effective, high-accuracy but faster as seen in Figure 2.

3.2 MULTIMODAL PROBLEM PARTITION

The first step of PRAM is a partition procedure. We aim to maximize parallelism without excessive decomposition, as the latter can impede convergence and even increase solving time at scale. As an example, decomposing by individual commodity flow may generate millions of subproblems, requiring multiple batching rounds even on modern GPUs. So taking a step back, PRAM focuses on solutions at a node level (by treating commodities from the same source as one subset), reducing the complexity from $\mathcal{O}(|\mathcal{V}|^2)$ to generally tractable $\mathcal{O}(|\mathcal{V}|)$.

No need for any specially designed modules such as GNNs or RNNs, thanks to the off-the-shelf interface of MLMs, our design can be realized in a convenient manner. As illustrated on the left of Figure 3, the procedure amounts to instructing the MLM with subgraphs (in image form) together with the commodity information (in text form). Concretely, we plot all routing links from the source node to every other node once to obtain a visual representation, which is then fed to MLMs through a vision encoder (e.g., CLIP (Radford et al., 2021)). At the same time, we introduce subtask-aware prompting,



(a) Inter-agent communication (b) Reprogramming learned context embeddings (c) MARL

Figure 4: **Illustration of PRAM's adaptation framework.** In (a), PRAM builds inter-agent communication through LoRA and reprogramming context using cross attention in (b). In (c), policy gradient flow is computed from each agent's difference reward to estimate the contribution of its actions to the team's global reward.

in which each recent demand in the commodity subset is explicitly paired with the subtask's key descriptions (e.g., source node specification) and statistics (e.g., historical rolling average demand). A prompt example is in Appendix C.1. We then expect the MLM "agent" to see, understand and jointly reason over both modalities for a good MCF configuration.

3.3 LIGHTWEIGHT MULTI-AGENT ADAPTATION

In our implementation, we promote efficiency of PRAM by sharing the MLM backbone among the subproblem agents. The agents can still behave differently because they receive different observations, and thus evolve different hidden states. However, as the model is not inherently specialized for this task, the agents are unable to perceive each other's presence. This necessitates the introduction of additional (communication) parameters and raises the challenge of fine-tuning them. Below, we show how these issues are resolved in PRAM.

Communication Our scheme of inter-agent communication is in Figure 4(a). The global (trainable) parameters are added in two ways: (i) Inside the model, we introduce low-rank matrices for attention weights to approximate the changes needed in the MLM backbone parameters using the LoRA technique (Hu et al., 2022). (ii) Outside the model, inspired by LM's in-context learning (ICL) mechanics (Brown et al., 2020; Lin et al., 2024): an ability to flexibly adjust their prediction based on additional data given in context (i.e. in the input sequence itself), we create a set number of learnable "global context" embeddings as prefix of input prompts using the reprogramming technique (Li et al., 2023; Jin et al., 2024a). As illustrated in Figure 4(b), we connect the context to a frozen input embedding matrix of the tokenizer associated with the MLM, and we perform alignment with a multi-head cross attention layer. Specifically, we model text prototypes as keys and values by linearly aggregating original token embeddings, while the context embeddings act as queries to extract global information or guidance the agent requires. Upon packing the prompt and context embeddings, as shown in Figure 3, we feedforward them through the MLM backbone to produce the agent-decision embeddings. As our evaluation results in § 5 show, these designs impose limited memory overhead; in other words, PRAM is lightweight overall.

Fine-Tuning While independent adaptation of different agents via gradients from the global objective (loss) is straightforward, the lack of information exchange during fine-tuning limits the ability to learn coordinated strategies or to assess an individual agent's contribution to the system. To address this, we adopt multi-agent RL (MARL) algorithms (Busoniu et al., 2008; Kraemer & Banerjee, 2016; Foerster et al., 2018; Lyu et al., 2021; 2023). In particular, PRAM adopts *counterfactual reasoning* (Foerster et al., 2018; Xu et al., 2023), asking: How would the global objective change if only one subset's flows were reallocated while others remained fixed? The reward difference defines the single agent's advantage relative to a counterfactual baseline, thereby quantifying its contribution to the joint outcome.

Specifically, when (history) demands arrives, PRAM provides the token embeddings of each subset as the local state s_i to the corresponding RL agent i, which then outputs an action a_i from a policy head network, i.e., a vector of path weights for its managed demands. All agents share a policy network π_θ parameterized by θ, trained via policy gradient. We then denote by s the central state composed of all local states s_i, and by a the joint action of all agents' actions a_i. After all decisions are made, a

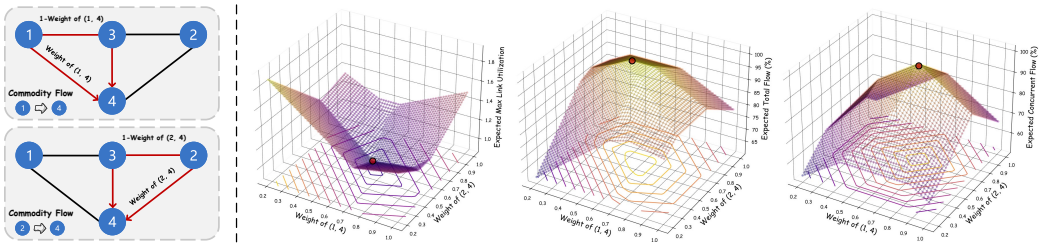


Figure 5: **A case study on the multi-commodity flow problem.** *(left)* In a simple network, both ① and ② need to transmit commodities to ④, each having two candidate paths. *(right)* The objective functions are plotted against the path weights, with their optima indicated by red dots. The curves exhibit clear convexity/concavity, lending theoretical support to the soundness of PRAM.

reward $R(s, a)$ is obtained. To compute the advantage $A_i(s, a)$, PRAM exploits the one-step nature of the MCF problem: since actions (flow allocations) do not affect future states (demands or/and paths), the expected return reduces to the immediate reward $R(s, a)$. Moreover, if agent i changes its action to a'_i while others keep theirs fixed, the new joint action (a_{-i}, a'_i) can be directly evaluated by simulating its effect on the objective, i.e., we compute the MCF objective obtained if the new joint action were to be used. Putting it together, PRAM computes the advantage for agent i as

$$A_i(s, a) = R(s, a) - \sum_{a'_i} \pi_\theta(a'_i|s_i) R\left(s, (a_{-i}, a'_i)\right), \quad g = \mathbb{E}_\pi \left[\sum_i A_i(s, a) \nabla_\theta \log \pi_\theta(a_i|s_i) \right],$$

where g is the policy gradient, and the counterfactual baseline is approximated via Monte Carlo sampling, i.e., drawing random actions $a'_i \sim \pi_\theta(\cdot|s_i)$. As shown in Figure 4(c), we fine-tune PRAM in an end-to-end manner, with θ denoting all tunable parameters, so that gradients are backpropagated seamlessly from the policy head. We include more details of the algorithm in Appendix C.3.

4 UNDERSTANDING PRAM: CASE STUDY AND THEORY

Through the Looking Glass of MCF To delve into the MCF problem, we conduct an illustrative case study in Figure 5. This study is based on a toy topology consisting of 4 nodes and 5 links, each with a capacity of 1. Now suppose nodes ① and ② need to send commodity flows to ④. With equal probability, their demands is either $\frac{3}{2}, \frac{6}{7}$ or $\frac{7}{6}, \frac{3}{2}$. The candidate paths from ① to ④ are $(1, 4)$ and $(1, 3, 4)$, while those from node ② to ④ are $(2, 4)$ and $(2, 3, 4)$. In this case, the configuration π is fully determined once the weights on links $(1, 4)$ and $(2, 4)$ are specified. In Figure 5 (right), we plot the curves of all three (expected) objective functions as the two weights vary, and mark the optimal values with red dots. We observe that the objective possesses a favorable property—*convexity/concavity* with respect to the path weights. such property ensures that simple first-order methods, i.e., gradient descent (GD), can effectively solve the problem. In particular, by iteratively updating the configuration in the direction of the steepest descent, one can guarantee convergence to the global optimum. We formalize this insight in the following theorem:

Theorem 1. *(Solving MCF with GD) Consider a GD algorithm with update rule $\pi^{(t+1)} = \pi^{(t)} - \eta v_t$. Then, there exists a step size $\eta > 0$ and a finite number of iterations T such that $\mathcal{L}(\pi)$, the MCF objective function, attains the optimum up to an arbitrarily small error.*

Appendix D.3 presents the proof of Theorem 1, which hinges on the convexity/concavity of the three objective functions. In our simple example, executing GD requires precise knowledge of the distribution of demands. However, It is almost impossible to estimate the expected objective without exact prediction of the future demands. To address this, a more practical approach is to implicitly model the probability distribution using a DNN trained on extensive empirical data (like PRAM).

What Makes PRAM Tick Given the above case study, to approximate the optimal mapping from recent observations to configurations, the theoretical analysis of PRAM's effectiveness is intrinsically aligned with the GD algorithm. Before proceeding, we first present the convergence guarantee of PRAM's multi-agent adaptation:

Lemma 1. *Assume that both $R(s, a)$ and its Hessian $\nabla^2_\theta R(s, a)$ are bounded. Let $\{\alpha_k\}_{k=0}^{\infty}$ be any step-size sequence satisfying $\lim_{k\to\infty} \alpha_k = 0$, $\sum_k \alpha_k = \infty$; and $\theta^{(k+1)} = \theta^{(k)} + \alpha_k v_k$, where v_k*

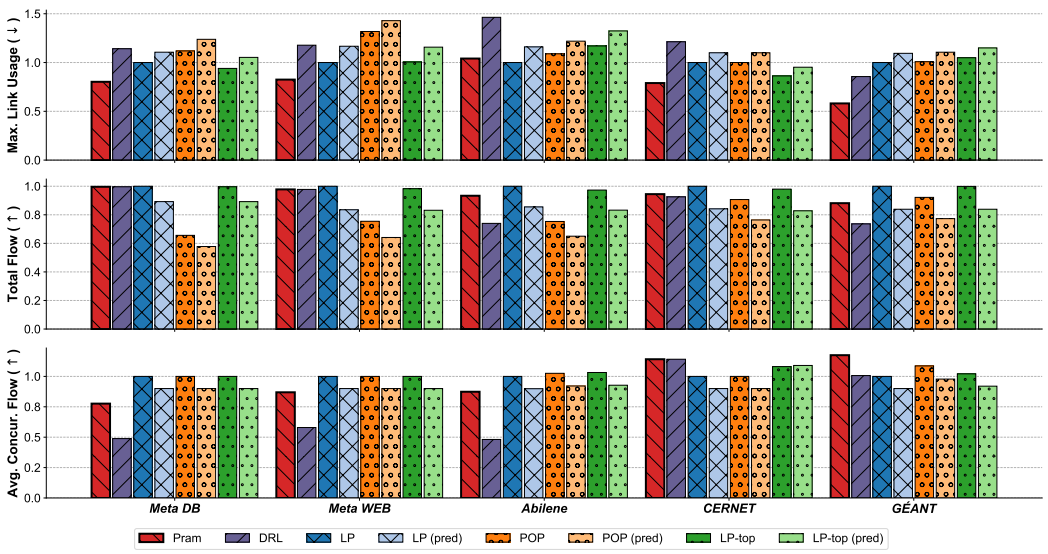


Figure 6: Real-world evaluation results of maximum link utilization (lower the better), allocated total flow (higher the better) and concurrent flow (higher the better). PRAM achieves top-tier performance.

is in the direction of the gradient. Then the policy iteration converges such that the expected gradient $\lim_{k\to\infty} \mathbb{E}_\pi \left[\sum_i A_i(s,a)\, \nabla_{\theta^{(k)}} \log \pi_{\theta^{(k)}}(a_i \mid s_i)\right] = 0$.

The proof of Lemma 1 (see Appendix D.4) refers to the approximation results from previous actor-critic algorithms (Bertsekas & Tsitsiklis, 1996; Sutton et al., 1999). Lemma 1 establishes the convergence of PRAM to a locally optimal agent, which implies that, once the fine-tuning is complete, our model can be regarded as a global mapping from observed demands to allocation schemes. Now back to MCF problems, our intuition that MLMs capable of running the GD algorithm is again in the light of ICL. Concretely, it can approximate gradient-based few-shot learning within its forward pass (Fu et al., 2023), thereby "reasoning" its way to the solution. We formalize this intuition in Theorem 2, where we assume the MCF problem in the token space of PRAM as a linear regression.

Theorem 2. *(*PRAM *Learns to Implement GD) Let the learned context satisfy Assumption 2, and let the problem be well-defined under Assumption 3. Then there exists an adapted MLM with constant depth and constant width that can simulate multiple steps of GD updates on the problem objective.*

The proof is deferred to Appendix D.5, which follows a simple weight construction introduced by Ahn et al. (2023). Taken together, the convexity of MCF problems (Theorem 1) and the ability of MLMs to simulate gradient descent (Theorem 2) establish that PRAM, once adapted, internally acts as an optimizer on the distance between initial configurations (input token embeddings) and near-optimal configurations (agent-decision embeddings)—consistent with our empirical results (§ 5).

5 MAIN RESULTS

We evaluate PRAM on open-source multimodal language models, with Qwen2.5-VL-7B-Instruct (Bai et al., 2025) as a standard. Unless otherwise noted, we truncate and fine-tune the first 8 layers of its LM as the backbone. Implementation specifics are in Appendix C, and key experimental setups are summarized below with further details in Appendix B. This section sequentially includes comparative experiments on real-world and large-scale generated datasets, generalization analysis, and ablation studies. We also provide additional supplementary experiments in the Appendix E.

Datasets. Our evaluation is conducted on two groups of datasets. First, we leverage five real-world datasets, namely *Meta DB, Meta WEB, Abilene, CERNET*, and *GÉANT*. These datasets are relatively small in scale, each containing fewer than 30 nodes. To complement it, we also incorporate five large-scale topologies with synthetic data, including *GtsCe, Colt, UsCarrier, Cogentco*, and *Kdl*. The sizes of these topologies range from 100 to 800 nodes. In § 5.2, we employ a gravity model (Roughan et al., 2002) to generate synthetic demands. Additionally, we select 12 most recently observed demands as history information.

Baselines. We compare PRAM with four representative baselines: linear programming (*LP*), partition-based optimization (*POP*), heuristic approach (*LP-top*), and reinforcement learning (*DRL*). Since

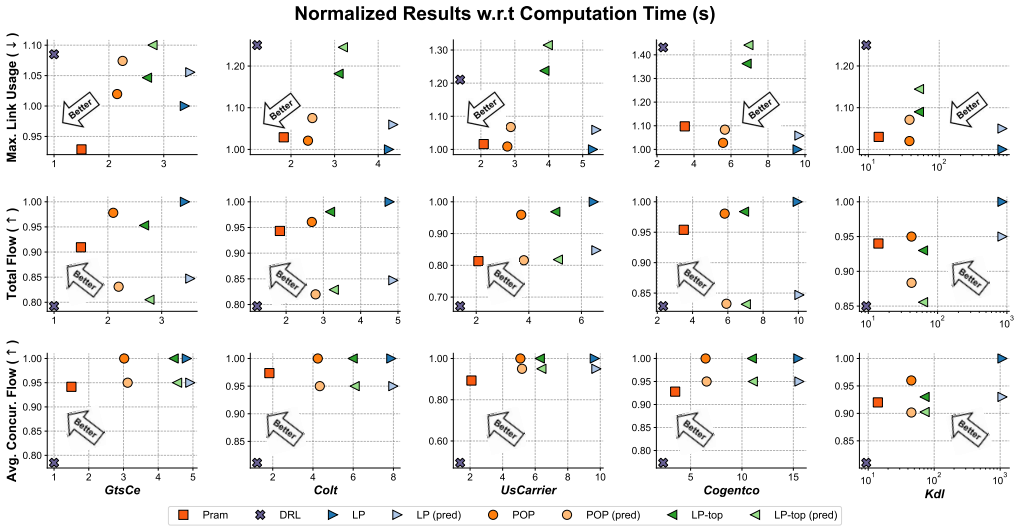


Figure 7: Comparison of PRAM with baselines on large-scale topologies using synthetic data. As the solution space expands, PRAM demonstrates scalable performance, achieving comparable or superior results with reduced computation time.

non-learning methods rely on the ground-truth demand, we also evaluate their predicted variants *(pred)* using the default moving-average forecasting, which we found to be simple and effective.

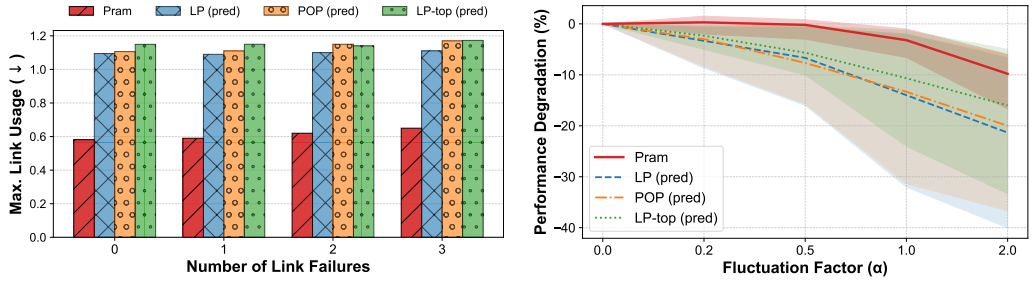
Metrics. We consider three standard objectives for the MCF problem: minimizing *link utilization*, maximizing *total flow*, and maximizing *concurrent flow*. To enable consistent comparison across datasets, all reported results are *normalized*, which is the ratio of the result to the (near-)optimal result obtained using the LP optimization solver. We also measure the *computation time* of each approach on the same machine to assess their efficiency.

5.1 EXPERIMENT ON REAL-WORLD DATASETS

Our first experiment aims to answer: **how does PRAM perform in practice?** We present the results of the experiment in Figure 6, which evaluates the performance of our proposed PRAM over publically available real-world datasets. In total we run three independent sub-experiments, one for each MCF allocation objective. First, the results reveal substantial room for improvement when flows are allocated by the prior ML-based approach (i.e., DRL), likely due to the well-known training challenges of RL with randomly initialized networks. Second, the gap between the optimal solution and the partitioned solution produced by POP is evident, making POP the second-worst method. Third, while LP-top often performs comparably to LP, it exhibits less stability because it focuses exclusively on optimizing large flows. From the figure, we can also see that relative to their exact-demand counterparts, all predictive variants suffer performance degradation (no less than 10%), with the note that the ML-based methods use historical demand as input. Finally, we find PRAM achieves the second-best average performance across the three objectives, ranking just behind the LP solver that has a perfect future prediction, indicating it is able to achieve near-optimal by leveraging the reasoning power of MLMs. One surprising observation is that, in the case of minimizing link usage, PRAM's output configurations are even better than that of LPs (in particular, about 21% lower in CERNET and 45% in GÉANT). This may be because MLU exhibits stronger convexity properties—consistent with our theoretical results—and demonstrates more stable variation ranges, making it especially amenable to fine-tuning with MLMs.

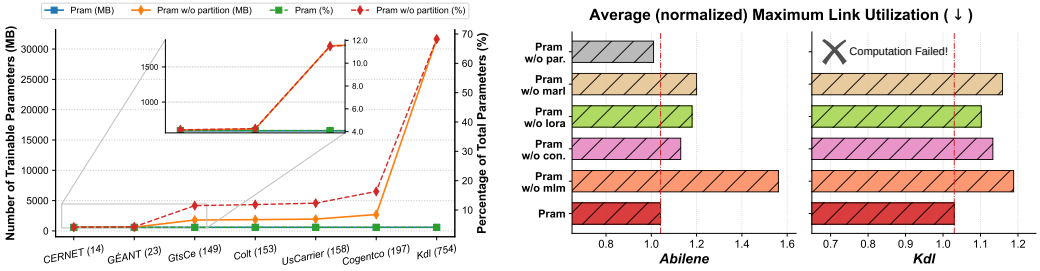
5.2 EXPERIMENT ON LARGE-SCALE DATASETS

Next, we conduct experiment on larger network topologies to explore the following question: **how does PRAM work at scale?** Analogously, Figure 7 compares online testing results of different methods together with their running time. As the network size grows, PRAM demonstrates the intended scalability due to our dividing operation, delivering high-quality allocations with lower computation time. In particular, on the largest Kdl topology (754 nodes, 1,790 links, and over 2 million candidate path weights), PRAM completes each flow allocation in under 25 seconds on



(a) MLU change with random link failures (b) Performance decline with increased fluctuation

Figure 8: Handling different unforeseen conditions on GÉANT; The right plot shows the mean and deviation from 10th to 90th percentile. The performance variations of PRAM are not significant.



(a) # of trainable parameters with(out) partition (b) Ablation study of PRAM's key components

Figure 9: Scalability and link utilization performance comparison between PRAM and its variants.

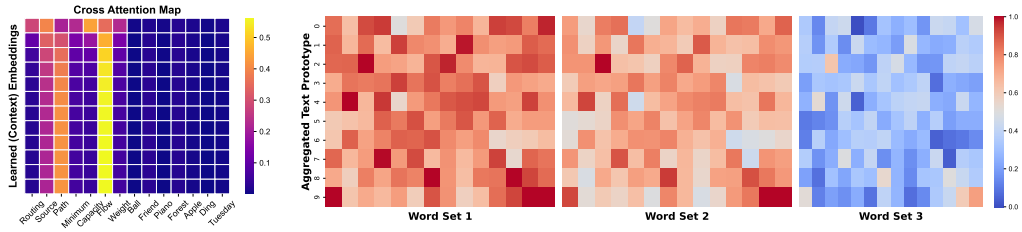
average—5× faster than POP, 7× faster than LP-top, and 100× faster than LP. For demand prediction, our non-parametric approach introduces negligible overhead. DRL, benefiting from a small policy network without MLM's backbone, achieves the fastest inference, but fails to approach near-optimal allocation quality. This is because the simple network also limits its expressive capability, and lacks effective modeling of complex structures. In contrast, PRAM satisfies > 90% of the performance of the best-performing LP scheme that is close enough to the optimal. Since we inject Gaussian noise to mimic temporal demand variations, prediction-based schemes achieve slightly better results than with real-world traces, yet still lag behind PRAM. In summary, ML-based methods are fast but inaccurate, while LP-based methods perform well but are inefficient. By striking a balance, PRAM offers both efficiency and accuracy, making it a compelling choice for production MCF systems.

5.3 GENERALIZATION TO UNSEEN SITUATIONS

In this subsection, we assess our model's performance on unforeseen conditions, i.e., asking: **can PRAM generalize well?** Due to space constraints, we hereafter report results primarily in terms of link utilization. We consider two most representative situations, where unforeseen changes affect the two inputs of the MCF problem: topology (link failure) and commodity demand (flow fluctuation).

(1) *Coping with link failures.* PRAM addresses link failures by first reassigning the affected commodity flows using simple heuristics. Take the case of three candidate paths having weights of $(\frac{1}{2}, \frac{1}{5}, \frac{3}{10})$. If the first path experiences a failure, the adjusted weights would be proportionally reset as $(0, \frac{2}{5}, \frac{3}{5})$. Such reassignment requires minimal computation time, and the entire process operates at the millisecond-level scale, comparable to a standard inference step. In addition, the capacity of failed links will be marked as zero, and it will be depicted in the sub-image which is then fed into PRAM. Figure 8(a) compare the performance of PRAM with prediction-based methods (access to failures), in terms of link utilization, when different numbers of links fail in the GÉANT topology. Our results show that PRAM still outperforms all baselines, indicating a good robustness to failures.

(2) *Reacting to flow fluctuation.* Our approach for increasing flow variability is as follows: we introduce independent noise into each source-destination pair by sampling from a Gaussian distribution $\mathcal{N}(\mu, \sigma_{s,d}^2)$, where $\mu = 0$ and $\sigma_{s,d}$ is the standard deviation of the demand $\mathcal{D}_{s,d}$. These noises are then multiplied by a factor α to control the fluctuation intensity. Also, figure 8(b) shows that the performance degradation of PRAM is no higher than 16% even for α=2 in the 90th percentile. Meanwhile, the three baselines struggle in this scenario, with a drop more than twice that of PRAM.



(a) Cross-attention map (b) Visualization of ten different learned text prototypes

Figure 10: Learned representation interpretation of the context embeddings. In (a), each row represents a context embedding, while columns correspond to selected words. In (b), each row represents a linear aggregated prototype, while columns correspond to words from different sets.

5.4 Ablation Study and Visualization

At the end of this section, we would like to investigate the final question: **How is PRAM effective?** We perform an ablation study to assess the impact of PRAM's key features on its overall performance.

(1) *Impact of Problem Partition.* Figure 9(a) compares the number of tunable parameters in PRAM with its non-partitioned variant, i.e., PRAM w/o partition (or w/o par. in short), across topologies of different sizes. Without partitioning, the parameter scale grows uncontrollably; for example, on the KDL topology, the number of trainable parameters is about 31,600 MB, which nearly matches that of full-parameter MLM adaptation. In contrast, PRAM localizes its parameters within LoRA and context modules—both inherently lightweight as shown in the figure—so its model size hardly increases with network scale. These results underscore the critical role of problem partition in handling MCF optimizations with large solution spaces.

(2) *Effect of Adaptation Choices.* To assess the importance of PRAM's core components, we retrain five variants on the Abilene and KDL topologies: ① PRAM w/o mlm: the MLM backbone is removed, with sub-topologies handled by a GNN and demands by FC layers; ② PRAM w/o con.: the global context is excluded from the input prompt; ③ PRAM w/o lora: all low-rank adapters are removed from the backbone; ④ PRAM w/o marl: the model is directly fine-tuned with the objective function end to end; ⑤ PRAM w/o par.: identical to PRAM w/o partition. The results are presented in Figure 9(b). We can see that using MLM significantly improves performance in minimizing MLU especially for Abilene. Moreover, all three designs (i.e., LoRA, learned context and muli-agent RL) used in our adaptation contribute to the optimization of the objective function. Although PRAM w/o partition achieves a lower MLU on the smaller network, it is too "gigantic" to make decisions on large Kdl.

(3) *Visualization of Learned Representation.* We present an interpretation of the learned context module on Abilene dataset in Figure 10. As shown in Figure 10(a), we visualize the cross-attention map of a single head by replacing text prototypes with token embeddings of words either related or unrelated to the task. Each cell reflects the relevance between a row and a column (with brighter values indicating stronger correlation). To further examine how context is represented through different prototype combinations, we randomly select ten prototypes and visualize them in Figure 10(b). For this analysis, we consider three word sets — word set 1 (commodity):{"Flow", "Demand", . . . }, word set 2 (topology): {"Capacity", "Node", . . . }, and word set 3 (others): {"Monday", "February", . . . }. From the heatmaps, we make two key observations: first, prototypes exhibit strong relevance to words describing MCF problems; and second, PRAM successfully aligns its learned context with task-relevant word embeddings. So we can reasonably conclude that context satisfies our assumption.

6 Conclusion and Future Work

In this paper, we present PRAM, a brand-new MCF solution that takes advantage of MLMs to optimize practical flow allocation performance. The MLM is equipped with a partition method to "divide" complexity and transform the original problem into a multi-agent decision-making problem. Then we design a communication and adaptation framework to "harmonize" the learning of these agents. We also theoretically study the effectiveness of PRAM in well-studied theoretical models. Our evaluations demonstrate that the PRAM is accurate, fast and robust, highlighting the potential of leveraging MLMs for "shooting" MCF problems. However, fine-tuning PRAM remains resource-intensive, even after truncating. We observed that processing identical sub-images consumes a considerable amount of memory. Thus, future work could explore more efficient memory storage and retrieval mechanisms.

REFERENCES

Firas Abuzaid, Srikanth Kandula, Behnaz Arzani, Ishai Menache, Matei Zaharia, and Peter Bailis. Contracting wide-area network topologies to solve flow problems quickly. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pp. 175–200, 2021. Cited on pages 1 and 18.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. Cited on page 18.

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *Advances in Neural Information Processing Systems*, 36:45614–45650, 2023. Cited on pages 6, 19, 29, 30, and 36.

Nasrin Akhter and Mohamed Othman. Energy aware resource allocation of cloud data center: review and open issues. *Cluster computing*, 19(3):1163–1182, 2016. Cited on page 18.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2023. Cited on page 19.

David Applegate and Edith 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*, pp. 313–324, 2003. Cited on pages 1, 18, and 22.

Arjang A Assad. Multicommodity network flows—a survey. *Networks*, 8(1):37–91, 1978. Cited on page 1.

Yossi Azar, Edith Cohen, Amos Fiat, Haim Kaplan, and Harald Racke. Optimal oblivious routing in polynomial time. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pp. 383–388, 2003. Cited on page 18.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. URL https://huggingface.co/collections/Qwen/qwen25-vl-6795ffac22b334a837c0f9a5. Cited on pages 6 and 37.

Burcu Balcik, Seyed Iravani, and Karen Smilowitz. Multi-vehicle sequential resource allocation for a nonprofit distribution system. *IIE Transactions*, 46(12):1279–1297, 2014. Cited on pages 1 and 18.

Guillermo Bernárdez, José Suárez-Varela, Albert López, Bo Wu, Shihan Xiao, Xiangle Cheng, Pere Barlet-Ros, and Albert Cabellos-Aparicio. Is machine learning ready for traffic engineering optimization? In *2021 IEEE 29th International Conference on Network Protocols (ICNP)*, pp. 1–11. IEEE, 2021. Cited on pages 1, 3, and 18.

Dimitri Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996. Cited on pages 6 and 29.

Niels Blaauwbroek, Phuong H Nguyen, Mente J Konsman, Huaizhou Shi, Ren IG Kamphuis, and Wil L Kling. Decentralized resource allocation and load scheduling for multicommodity smart energy systems. *IEEE Transactions on Sustainable Energy*, 6(4):1506–1514, 2015. Cited on pages 1 and 18.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. Cited on page 4.

Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008. Cited on page 4.

Konstantinos J Chalvatzis, Hanif Malekpoor, Nishikant Mishra, Fiona Lettice, and Sonal Choudhary. Sustainable resource allocation for power generation: The role of big data in enabling interindustry architectural innovation. *Technological Forecasting and Social Change*, 144:381–393, 2019. Cited on page 18.

Ching Chang, Wen-Chih Peng, and Tien-Fu Chen. Llm4ts: Two-stage fine-tuning for time-series forecasting with pre-trained llms. *CoRR*, 2023. Cited on page 19.

Fangzhe Chang, Jennifer Ren, and Ramesh Viswanathan. Optimal resource allocation in clouds. In *2010 IEEE 3rd International Conference on Cloud Computing*, pp. 418–425. IEEE, 2010. Cited on pages 1 and 18.

Li Chen and Mingquan Ye. High-accuracy multicommodity flows via iterative refinement. In *51st International Colloquium on Automata, Languages, and Programming (ICALP 2024)*, pp. 45–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024. Cited on page 1.

Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. Llaga: large language and graph assistant. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 7809–7823, 2024. Cited on page 19.

Xingwu Chen and Difan Zou. What can transformer learn with varying depth? case studies on sequence learning tasks. In *International Conference on Machine Learning*, pp. 7972–8001. PMLR, 2024. Cited on page 19.

Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *Journal of the ACM (JACM)*, 68(1):1–39, 2021. Cited on pages 1, 2, and 3.

Shimon Even, Alon Itai, and Adi Shamir. On the complexity of time table and multi-commodity flow problems. In *16th annual symposium on foundations of computer science (sfcs 1975)*, pp. 184–193. IEEE, 1975. Cited on page 2.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. Cited on page 19.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. Cited on pages 2, 4, 29, and 35.

Deqing Fu, CHEN Tianqi, Robin Jia, and Vatsal Sharan. Transformers learn higher-order optimization methods for in-context learning: A study with linear models. 2023. Cited on page 6.

Javier González and Aditya Nori. Does reasoning emerge? examining the probabilities of causation in large language models. *Advances in Neural Information Processing Systems*, 37:117737–117761, 2024. Cited on page 18.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL https://www.gurobi.com. Cited on page 23.

Aric Hagberg, Pieter J Swart, and Daniel A Schult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2008. Cited on page 25.

Elad Hazan, Kfir Levy, and Shai Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. *Advances in neural information processing systems*, 28, 2015. Cited on page 30.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. Cited on page 1.

Oliver Hope and Eiko Yoneki. Gddr: Gnn-based data-driven routing. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pp. 517–527. IEEE, 2021. Cited on page 18.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations*, 2022. Cited on page 4.

Jianhao Huang, Zixuan Wang, and Jason D Lee. Transformers learn to implement multi-step gradient descent with chain of thought. In *The Thirteenth International Conference on Learning Representations*, 2025. Cited on page 19.

John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03): 90–95, 2007. Cited on page 25.

Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. B4: Experience with a globally-deployed software defined wan. *ACM SIGCOMM Computer Communication Review*, 43(4):3–14, 2013. Cited on page 18.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations*, 2024a. Cited on pages 4 and 19.

Ming Jin, Yifan Zhang, Wei Chen, Kexin Zhang, Yuxuan Liang, Bin Yang, Jindong Wang, Shirui Pan, and Qingsong Wen. Position: What can large language models tell us about time series analysis. In *Forty-first International Conference on Machine Learning*, 2024b. Cited on page 19.

Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pp. 302–311, 1984. Cited on page 1.

Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980. Cited on page 1.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Cited on page 28.

Simon Knight, Hung X Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011. URL http://www.topology-zoo.org/. Cited on page 21.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016. Cited on page 4.

Umesh Krishnaswamy, Rachee Singh, Nikolaj Bjørner, and Himanshu Raj. Decentralized cloud wide-area network traffic engineering with {BLASTSHIELD}. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pp. 325–338, 2022. Cited on page 18.

Praveen Kumar, Yang Yuan, Chris Yu, Nate Foster, Robert Kleinberg, Petr Lapukhov, Chiun Lin Lim, and Robert Soulé. {Semi-oblivious} traffic engineering: The road not taken. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pp. 157–170, 2018. Cited on page 18.

Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *2015 IEEE 56th annual symposium on foundations of computer science*, pp. 230–249. IEEE, 2015. Cited on page 2.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. Cited on page 4.

Xing Li, Congxiao Bao, Maoke Chen, Hong Zhang, and Jianping Wu. The china education and research network (cernet) ivi translation design and deployment for the ipv4/ipv6 coexistence and transition. Technical report, 2011. Cited on page 21.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. In *The Twelfth International Conference on Learning Representations*, 2024. Cited on page 19.

Licong Lin, Yu Bai, and Song Mei. Transformers as decision makers: Provable in-context reinforcement learning via supervised pretraining. In *The Twelfth International Conference on Learning Representations*, 2024. Cited on pages 4 and 18.

Zhiyuan Liu, Sihang Li, Yanchen Luo, Hao Fei, Yixin Cao, Kenji Kawaguchi, Xiang Wang, and Tat-Seng Chua. Molca: Molecular graph-language modeling with cross-modal projector and uni-modal adapter. In *EMNLP*, 2023. Cited on page 19.

Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 844–852, 2021. Cited on page 4.

Xueguang Lyu, Andrea Baisero, Yuchen Xiao, Brett Daley, and Christopher Amato. On centralized critics in multi-agent reinforcement learning. *Journal of Artificial Intelligence Research*, 77: 295–354, 2023. Cited on page 4.

Arvind V Mahankali, Tatsunori Hashimoto, and Tengyu Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. In *The Twelfth International Conference on Learning Representations*, 2024. Cited on page 19.

Marvin B Mandell. Modelling effectiveness-equity trade-offs in public service delivery systems. *Management Science*, 37(4):467–482, 1991. Cited on page 18.

Meta. Llama-3.2-11b-vision-instruct, 2024. URL "https://huggingface.co/meta-llama/Llama-3.2-11B-Vision-Instruct". Cited on page 37.

Pooria Namyar, Behnaz Arzani, Ryan Beckett, Santiago Segarra, Himanshu Raj, and Srikanth Kandula. Minding the gap between fast heuristics and their optimal counterparts. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, pp. 138–144, 2022. Cited on page 24.

Deepak Narayanan, Fiodar Kazhamiaka, Firas Abuzaid, Peter Kraft, Akshay Agrawal, Srikanth Kandula, Stephen Boyd, and Matei Zaharia. Solving large-scale granular resource allocation problems efficiently with pop. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles (SOSP 21)*, pp. 521–537, 2021. URL https://github.com/stanford-futuredata/POP. Cited on pages 2, 18, and 23.

Eshaan Nichani, Alex Damian, and Jason D Lee. How transformers learn causal structure with gradient descent. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 38018–38070, 2024. Cited on page 19.

Md Noor-A-Rahim, Zilong Liu, Haeyoung Lee, GG Md Nawaz Ali, Dirk Pesch, and Pei Xiao. A survey on resource allocation in vehicular networks. *IEEE transactions on intelligent transportation systems*, 23(2):701–721, 2020. Cited on page 18.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. Cited on page 24.

Yarin Perry, Felipe Vieira Frujeri, Chaim Hoch, Srikanth Kandula, Ishai Menache, Michael Schapira, and Aviv Tamar. {DOTE}: Rethinking (predictive){WAN} traffic engineering. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pp. 1557–1581, 2023. Cited on pages 3 and 29.

Harald Racke. Minimizing congestion in general networks. In *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pp. 43–52. IEEE, 2002. Cited on page 18.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021. Cited on page 3.

Matthew Roughan, Albert Greenberg, Charles Kalmanek, Michael Rumsewicz, Jennifer Yates, and Yin Zhang. Experience in measuring backbone traffic variability: Models, metrics, measurements and meaning. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, pp. 91–92, 2002. Cited on pages 6 and 22.

Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren. Inside the social network's (datacenter) network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 123–137, 2015. Cited on page 20.

Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. Understanding transformer reasoning capabilities via graph algorithms. *Advances in Neural Information Processing Systems*, 37:78320–78370, 2024. Cited on page 19.

Alexander Schrijver. On the history of the transportation and maximum flow problems. *Mathematical programming*, 91(3):437–445, 2002. Cited on pages 1 and 18.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. URL https://github.com/PredWanTE/DOTE/tree/main/openai_baselines. Cited on page 23.

Lingfeng Shen, Aayush Mishra, and Daniel Khashabi. Position: Do pretrained transformers learn in-context by gradient descent. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 44712–44740, 2024. Cited on page 19.

Chengshuai Shi, Kun Yang, Jing Yang, and Cong Shen. Transformers as game players: Provable in-context game-playing capabilities of pre-trained models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. Cited on page 19.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999. Cited on pages 6, 29, and 31.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 491–500, 2024. Cited on page 19.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024. Cited on page 18.

Claudia Tebaldi and Mike West. Bayesian inference on network traffic using link count data. *Journal of the American Statistical Association*, 93(442):557–573, 1998. Cited on page 22.

Steve Uhlig, Bruno Quoitin, Jean Lepropre, and Simon Balon. Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM Computer Communication Review*, 36(1): 83–86, 2006. URL https://totem.info.ucl.ac.be/dataset.html. Cited on page 21.

Asaf Valadarsky, Michael Schapira, Dafna Shahaf, and Aviv Tamar. Learning to route. In *Proceedings of the 16th ACM workshop on hot topics in networks*, pp. 185–191, 2017. Cited on pages 1, 18, and 23.

Hao Wang, Haiyong Xie, Lili Qiu, Yang Richard Yang, Yin Zhang, and Albert Greenberg. Cope: Traffic engineering in dynamic networks. In *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 99–110, 2006. Cited on pages 1, 3, and 18.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. Can language models solve graph problems in natural language? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. Cited on page 19.

Siwei Wang, Yifei Shen, Shi Feng, Haoran Sun, Shang-Hua Teng, and Wei Chen. Alpine: Unveiling the planning capability of autoregressive learning in language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. Cited on page 19.

Yufei Wang and Zheng Wang. Explicit routing algorithms for internet traffic engineering. In *Proceedings Eight International Conference on Computer Communications and Networks (Cat. No. 99EX370)*, pp. 582–588. IEEE, 1999. Cited on pages 1 and 18.

Taylor Webb, Keith J Holyoak, and Hongjing Lu. Emergent analogical reasoning in large language models. *Nature Human Behaviour*, 7(9):1526–1541, 2023. Cited on page 18.

Edward Weiner. *Urban transportation planning in the United States*. Springer, 1987. Cited on page 18.

Wann-Ming Wey and Kuei-Yang Wu. Using anp priorities with goal programming in resource allocation in transportation. *Mathematical and computer modelling*, 46(7-8):985–1000, 2007. Cited on page 18.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992. Cited on page 35.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019. Cited on page 24.

Duo Wu, Xianda Wang, Yaqi Qiao, Zhi Wang, Junchen Jiang, Shuguang Cui, and Fangxin Wang. Netllm: Adapting large language models for networking. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pp. 661–678, 2024a. Cited on page 2.

Xixi Wu, Yifei Shen, Caihua Shan, Kaitao Song, Siwei Wang, Bohang Zhang, Jiarui Feng, Hong Cheng, Wei Chen, Yun Xiong, and Dongsheng Li. Can graph learning improve planning in LLM-based agents? In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. Cited on page 19.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020. Cited on page 3.

Yang Xiao, Jun Liu, Jiawei Wu, and Nirwan Ansari. Leveraging deep reinforcement learning for traffic engineering: A survey. *IEEE Communications Surveys & Tutorials*, 23(4):2064–2097, 2021. Cited on page 18.

Zhiying Xu, Francis Y Yan, Rachee Singh, Justin T Chiu, Alexander M Rush, and Minlan Yu. Teal: Learning-accelerated optimization of wan traffic engineering. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pp. 378–393, 2023. Cited on page 4.

Zhiyuan Xu, Jian Tang, Jingsong Meng, Weiyi Zhang, Yanzhi Wang, Chi Harold Liu, and Dejun Yang. Experience-driven networking: A deep reinforcement learning based approach. In *IEEE INFOCOM 2018-IEEE conference on computer communications*, pp. 1871–1879. IEEE, 2018. Cited on page 18.

Huimin Yan, Meng Li, and Xi Lin. Time-dependent on-street parking planning in a connected and automated environment. *Transportation Research Part C: Emerging Technologies*, 142:103745, 2022. Cited on page 18.

Jin Y Yen. Finding the k shortest loopless paths in a network. *management Science*, 17(11):712–716, 1971. Cited on page 24.

Junjie Zhang, Minghao Ye, Zehua Guo, Chen-Yu Yen, and H Jonathan Chao. Cfr-rl: Traffic engineering with reinforcement learning in sdn. *IEEE Journal on Selected Areas in Communications*, 38 (10):2249–2259, 2020. Cited on page 18.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, et al. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pp. 169–186. Springer, 2024. Cited on page 2.

Yin Zhang, Zihui Ge, Albert Greenberg, and Matthew Roughan. Network anomography. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pp. 30–30, 2005. URL https://www.cs.utexas.edu/~yzhang/research/AbileneTM/. Cited on page 21.

Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, et al. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36:43322–43355, 2023. Cited on page 19.

Zihao Zhou and Rose Yu. Can LLMs understand time series anomalies? In *The Thirteenth International Conference on Learning Representations*, 2025. Cited on page 19.

Appendix

TABLE OF CONTENTS

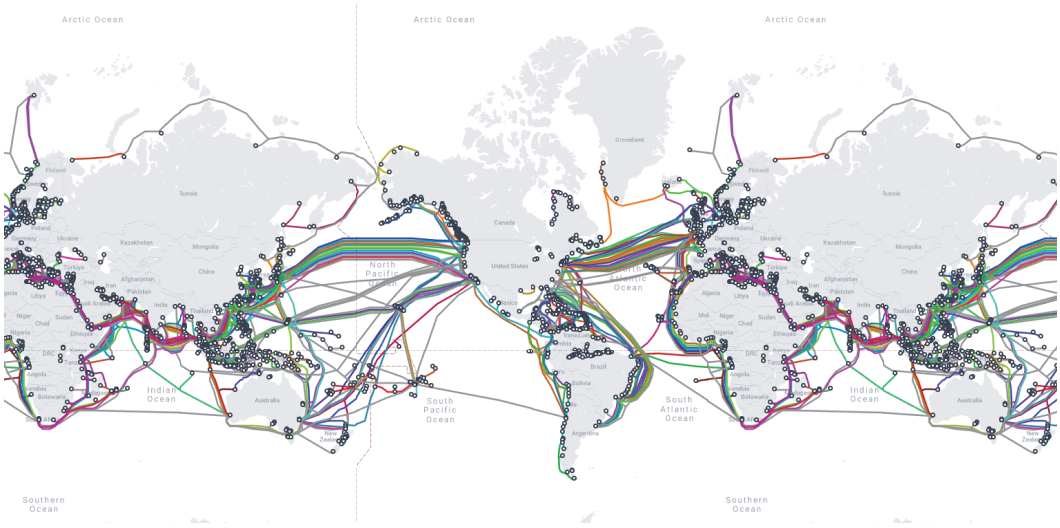


Figure 11: Submarine cables serve as choke points in large-scale topologies for network communication, providing a critical yet expensive infrastructure; the figure is drawn from this URL.

A Related Works

Solving Multi-Commodity Flow Problems The multi-commodity flow (MCF) is a very classic resource allocation problem widely found in various fields such as network (Wang & Wang, 1999; Applegate & Cohen, 2003; Jain et al., 2013; Kumar et al., 2018; Abuzaid et al., 2021; Krishnaswamy et al., 2022), power (Blaauwbroek et al., 2015; Chalvatzis et al., 2019), transportation (Weiner, 1987; Schrijver, 2002; Wey & Wu, 2007; Noor-A-Rahim et al., 2020), logistics (Mandell, 1991; Balcik et al., 2014; Yan et al., 2022) and could services (Chang et al., 2010; Akhter & Othman, 2016) and so on, playing a critical role in service and provider infrastructures. This part will primarily discuss works related to network traffic engineering (TE), which is the most extensively studied MCF problem. Unlike adaptive TE, Oblivious routing often optimizes worst-case link utilization across all possible demand matrices (Racke, 2002; Applegate & Cohen, 2003; Azar et al., 2003). To fit current demands and react to failures, SMORE (Kumar et al., 2018) proposed semi-oblivious routing by dynamically adapting sending rates. An improved work is COPE (Wang et al., 2006), which achieves optimization within the space spanned by historical DMs without compromising worst-case performance bounds. However, this combined approach of generalizing oblivious routing and performing multi-matrix optimization incurs substantial computational overhead (especially for large-scale systems shown in Figure 11). Although NCFlow (Abuzaid et al., 2021) and POP (Narayanan et al., 2021) significantly speed up multi-commodity flow computations via problem decomposition and parallelization, their methods inherit the limitations of prediction-dependent approaches. Nowadays, researchers have also explored ML-based allocation (Xiao et al., 2021). These methods typically use Demand-prediction and RL approaches (Valadarsky et al., 2017; Xu et al., 2018; Zhang et al., 2020; Hope & Yoneki, 2021; Bernárdez et al., 2021). For example, CFR-RL (Zhang et al., 2020) learns a policy to select critical flows for each given DM automatically. To explore the feasibility of combining GNNs with DRL, GDDR (Hope & Yoneki, 2021) and MARL-GNN (Bernárdez et al., 2021) transformed TE into online optimization problems over graphs to handle topologies of various structures. They represent the state-of-the-art recently, yet still exhibits obvious limitations—it often comes with high computational complexity and sensitivity to parameter settings. Compared to the above methods, PRAM is not only easy to train, but also several orders of magnitude faster than traditional methods when solving large-scale MCF problems.

Investigating Reasoning Ability of Language Models Recent advancements in large language models have exhibited exceptional proficiency across diverse NLP tasks. Commercial LLMs such as ChatGPT (Achiam et al., 2023) and Gemini (Team et al., 2024) currently embody the state-of-the-art in reasoning capabilities. Webb et al. (2023) first claimed that LLMs like GPT-3 have acquired an emergent ability to find zero-shot solutions to a broad range of analogy problems. González & Nori (2024) extended it beyond mathematics to vision domains. And Lin et al. (2024) investigated the application of in-context learning (ICL) in pre-trained models for decision-making

tasks, while Shi et al. (2024) explored its efficacy in game-playing scenarios. For theoretical analysis of LMs' reasoning capabilities, Wang et al. (2024) show Transformers learn planning via path-finding by encoding adjacency/reachability matrices. Subsequently, Sanford et al. (2024) sharply separate Transformers from other architectures and provides width and depth separations via graph algorithms. Moreover, Li et al. (2024) demonstrates that chain-of-thought (CoT) enables bounded-depth transformers to solve serial computation problems, and Chen & Zou (2024) studied the capabilities of the transformer architecture with varying depth. Huang et al. (2025) proved that a transformer with CoT prompting can learn to perform multi-step gradient descent autoregressively. Several works have also established the existence of deep transformers capable of implementing gradient descent across different domains (Akyürek et al., 2023; Ahn et al., 2023; Mahankali et al., 2024; Nichani et al., 2024; Shen et al., 2024). In the theoretical part of our study, we also leverage the LMs' advanced reasoning power to guide and assist in resource allocation.

Adapting Language Models for TS or Graph Here, we review previous work that employs LLMs for time series (TS) and graph applications. The first category is to boost time series related tasks (Jin et al., 2024b), such as forecasting, imputation and anomaly detection. To adapt LLMs for time series forecasting, Chang et al. (2023) employed a two-stage approach: fine-tuning GPT-2's transformer module and redesigning positional encoding. Besides, Zhou et al. (2023) propose a parameter-efficient adaptation method for pre-trained language models, named "One Fits All", that preserves their original self-attention and feedforward architectures. Time-LLM (Jin et al., 2024a) further proposed an embedding reprogramming technique to align the language model's word embedding space with time-series representations. Moreover, Zhou & Yu (2025) conducted a comprehensive investigation into LLMs' understanding of time series data and their anomaly detection capabilities. The second category encompasses graph tasks, which typically include graph learning and graph algorithms. For node classification tasks in text-attributed graphs, GraphGPT (Tang et al., 2024) processed the input graph through a GNN encoder for tokenization before LLM integration. Following this paradigm, MolCA (Liu et al., 2023) applied similar GNN-based encoders to handle molecular structures when predicting their properties. To enable LLMs to reason over graph-structured data, NLGraph (Wang et al., 2023) and Talk like a Graph (Fatemi et al., 2024) introduced a natural language translation step, followed by few-shot prompting or CoT techniques for effective inference. LLaGA (Chen et al., 2024) reformatted the center node and its neighborhood into a structure-aware textual representation, and Wu et al. (2024b) conducted a pioneering investigation into graph-learning methodologies for task planning in language agents. Unlike existing approaches, this paper advances MCF solutions as a novel framework to bridge the gap between these two research trajectories.

B Detailed Experimental Setup

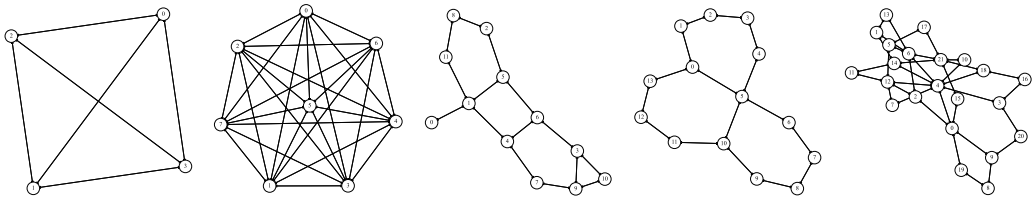
In this section, we provide the detailed experimental setup to ensure clarity of our paper. We first outline the objectives of the experiments, followed by the datasets used in our study. We then introduce the baseline methods for comparison and, finally, describe the path selection strategy adopted in our framework.

B.1 MCF Objectives

In this paper, we use three optimization objectives—maximum link utilization, maximum total flow and maximum concurrent flow, to evaluate allocation configurations. Using the notations introduced in § 2, we define three objectives as follows.

(1) **maximum link utilization**: It refers to the maximum value of all link utilization ratios in the topology, which is a classical allocation objective. A lower link utilization suggests greater resilience. Equation (1) summarizes the formulation.

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



(a) Meta DB (b) Meta WEB (c) Abilene (d) CERNET (e) GÉANT

Figure 12: Five real-world network topologies of various systems.

(2) **maximum total flow**: The objective computes a policy that satisfies the demand and capacity constraints while maximizing permissible flow. We here additionally define a pair-wise capacity ω_p that represents the maximum permissible flow between s to t along the tunnel p. ω_p is also an optimizable component of the configuration. Then the optimization problem is formulated as follows.

$$\begin{aligned}
\textbf{maximize} \quad & \alpha = \sum_{s,t\in\mathcal{V}} \sum_{p\in P_{s,t}} f_p \\
\textbf{subject to} \quad & f_p \le \mathcal{D}_{s,t} \cdot r_p, \quad \forall s,t \in \mathcal{V},\ \forall p \in P_{s,t} \\
& f_p \le \omega_p, \quad \forall s,t \in V,\ \forall p \in P_{s,t} \\
& r_p \ge 0,\ \omega_p \ge 0,\ f_p \ge 0, \quad \forall s,t \in \mathcal{V},\ \forall p \in P_{s,t} \\
& \sum_{s,t\in V} \sum_{p\in P_{s,t}} \sum_{e\ni p} \omega_p \le c_e, \quad \forall e \in \mathcal{E} \\
& r_p \ge 0, \quad \forall s,t \in \mathcal{V},\ \forall p \in P_{s,t}
\end{aligned} \tag{2}$$

(3) **maximum concurrent flow**: In maximum-multicommodity-flow problems, the objective is to compute a configuration that maximizes the total network throughput. Specifically, it depends on a surrogate bound α, which is an α-fraction of each $\mathcal{D}_{s,t}$ is routed concurrently. By defining $f_{s,t}^{(p)}$ as the actual commodity flow on the tunnel p between s and t, the surrogate problem can be formally expressed as

$$\begin{aligned}
\textbf{maximize} \quad & \alpha \in [0,1] \\
\textbf{subject to} \quad & \sum_{p\in P_{s,t}} f_{s,t}^{(p)} \ge \alpha \cdot \mathcal{D}_{s,t}, \quad \forall s,t \in \mathcal{V} \\
& \sum_{s,t\in V} \sum_{p\in P_{s,t}} \sum_{e\ni p} f_{s,t}^{(p)} \le c(e), \quad \forall e \in \mathcal{E} \\
& f_{s,t}^{(p)} \ge 0, \quad \forall s,t \in \mathcal{V},\ \forall p \in P_{s,t} \\
& f_{s,t}^{(p)} \le \mathcal{D}_{s,t} \cdot r_p, \quad \forall s,t \in \mathcal{V},\ \forall p \in P_{s,t} \\
& r_p \ge 0, \quad \forall s,t \in \mathcal{V},\ \forall p \in P_{s,t}
\end{aligned} \tag{3}$$

However, since in many cases the minimum concurrent flow is either very close to zero or exactly zero and thus not representative, in the experiments we instead use the average satisfaction ratio across all flows to capture the intended fairness effect. And we adopt a 7:1:2 ratio for training, validation, and testing across all datasets.

B.2 Experimental Datasets

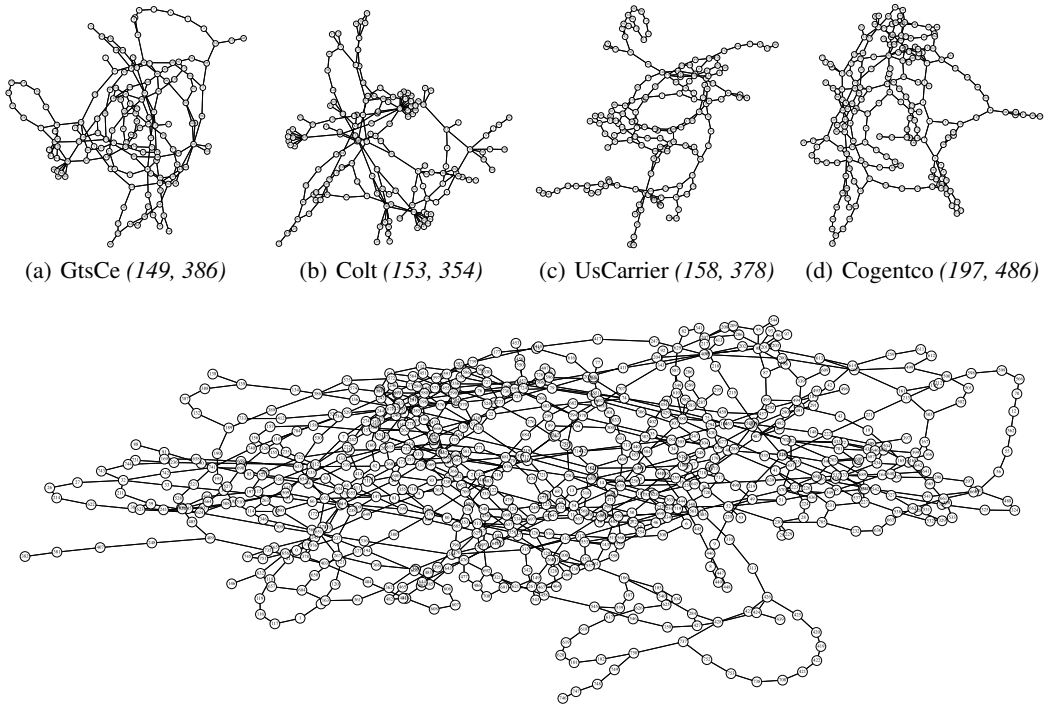
In this subsection, we provide detailed information on the datasets used in all experiments. Note that all these datasets and topologies are publicly available, and their corresponding access links can be found in the references.

B.2.1 Real-World Datasets

We first consider two small-scale social network datasets, **Meta DB** and **Meta WEB** (Roy et al., 2015). The Meta DB cluster consists of MySQL servers that store user data and process SQL queries,

Topology	# Nodes	# Links	# Total DMs	Granularity	Collection Time
Meta DB	4	12	10,000	\	\
Meta WEB	8	56	10,000	\	\
Abilene	12	30	48,384	5 minutes	March $\sim$ September 2004
CERNET	14	32	10,000	5 minutes	February $\sim$ March 2013
GÉANT	23	74	10,733	15 minutes	January $\sim$ April 2004

Table 1: Real-world network topology statistics



(a) GtsCe *(149, 386)* (b) Colt *(153, 354)* (c) UsCarrier *(158, 378)* (d) Cogentco *(197, 486)*

(e) Kdl *(754, 1790)*

Figure 13: Five large-scale topologies used in our evaluation.

while the Meta WEB cluster handles web traffic. Both clusters operate at the Point of Delivery (PoD) level, which is fully connected. We also use three real-world WAN datasets: US Internet2 Network (**Abilene**, Zhang et al. (2005)), Pan-European Research Network (**GÉANT**, Uhlig et al. (2006)) and China Education and Research Network (**CERNET**, Li et al. (2011)), each with public traffic matrices captured at different snapshots. Table 1 presents key statistics (i.e., number of nodes, links and samples) for these datasets. Besides, we visualize their topologies in Figure 12.

B.2.2 Synthetic Datasets

To evaluate PRAM's ability to scale to larger systems, we also use several topologies from the Internet Topology Zoo (Knight et al., 2011). They are **GtsCe** with 149 nodes and 386 edges, **Colt** with 153 nodes and 354 edges, **UsCarrier** with 158 nodes and 378 edges, **Cogentco** with 197 nodes and 486 edges, and **Kdl** with 754 nodes and 1790 edges. Figure 13 shows visualizations for the five used topologies; note that we set 1,000 for all link capacity. Also note that we only take the largest strong connected component in these topologies for our evaluations.

We synthesis demand matrices (DMs) for above large topologies based on three typical types of distribution or method[1]. All datasets except for Kdl, the number of generated samples is 3,000 while that of Kdl is 500 due to its super-high dimension. We present their details as following:

• *Gravity* (Roughan et al., 2002): The distribution has been an empirical success in that it accurately predicts trade flows between countries for many goods and services. Its key idea is that the total demand leaving a node is proportional to the total capacity on the node's outgoing links; this demand is divided among other nodes proportional to the total capacity on their incoming links. We present the Python pseudocode for the corresponding synthetic data in Algorithm 1. Specifically, the algorithm first computes the total inbound and outbound capacities for all nodes. It then derives the DMs by sampling from a Gaussian distribution centered at the expected fraction, with variance proportional to the same value. Finally, the resulting demand matrix is rescaled by a factor r to match the desired total resource level.

Algorithm 1 Gravity-based Demands Generation in a Python style

```
# G: graph or topology
# r: scaling factor

num_nodes = len(G.nodes) # number of nodes
in_cap, out_cap = {}, {}
A = zeros(num_nodes, num_nodes) # initialize DMs

for u in G.nodes:
    in_cap[u] = sum(cap(v, u) for v in predecessors(u))
    out_cap[u] = sum(cap(u, v) for v in successors(u))

for u in G.nodes:
    norm_u = out_cap[u] / sum(out_cap)
    for v in G.nodes:
        frac = norm_u * in_cap[v] / (sum(in_cap) - in_cap[u])

        # allocate resource with randomness
        A[u, v] = max(Gaussian(frac, frac/4), 0)

A = A * r # scale by constant factor
```

• *Poisson* (Tebaldi & West, 1998): This model can be formally expressed as $P(\lambda, \delta)$, where the demand between nodes s and t follows a Poisson random variable with mean $\lambda\delta^{d_{s,t}}$. Here, $d_{s,t}$ denotes the hop distance of the shortest path between s and t, and $\delta \in [0, 1]$ is a decay factor controlling demand concentration. Choosing δ close to 0 yields highly concentrated demands (favoring nearby nodes), while values close to 1 generate more uniform demands across the topology. The corresponding Python-style pseudocode is given in Algorithm 2. The algorithm first computes the all-pairs shortest path distances. For each source–destination pair, the expected demand is set to $\lambda\delta^{d_{s,t}}$, which is then sampled from a Poisson distribution to incorporate randomness. The diagonal entries are zeroed to exclude self-demands. Finally, the entire demand matrix is scaled by factor r to adjust the total resource level.

• *Bimodal* (Applegate & Cohen, 2003): In the bimodal model, a fraction p of source–destination pairs (selected uniformly at random) are assigned demands drawn from $\mathsf{Uniform}(b, c)$, while the remaining pairs receive demands from $\mathsf{Uniform}(0, a)$. This construction creates a heterogeneous demand matrix where some pairs experience significantly higher traffic than others. Algorithm 3 provides Python-style pseudocode for generating synthetic data under this model. The algorithm first initializes an empty demand matrix. It then uses a random assignment to decide the group for each entry. Depending on the group, entries are sampled from the corresponding uniform distribution. Finally, the diagonal entries are set to zero. This yields a demand matrix with a bimodal distribution of traffic intensities.

[1]The code is mainly based on this Github repository: https://github.com/netcontract/ncflow.

Algorithm 2 Poisson-based Demands Generation in a Python style

```
# G: graph or topology
# lam: base Poisson parameter
# decay: decay factor (<=1)
# r: scaling factor

num_nodes = len(G.nodes) # number of nodes

# compute all-pairs shortest distances
distances = zeros(num_nodes, num_nodes)
for src, dist_dict in shortest_path_length(G):
    for target, dist in dist_dict.items():
        distances[src, target] = dist

# generate Poisson-based demand matrix
A = array([[poisson(lam * (decay ** dist)) for dist in row]
          for row in distances], dtype=float)
np.fill_diagonal(A, 0.0) # no demand for self

A = A * r # scale by constant factor
```

Algorithm 3 Bimodal-based Demands Generation in a Python style

```
# G: graph or topology
# fraction: fraction of low-range allocations
# low_range: [min, max] for low allocations
# high_range: [min, max] for high allocations

num_nodes = len(G.nodes) # number of nodes

# initialize demand matrix
A = zeros(num_nodes, num_nodes)

p=[fraction, 1 - fraction]

# randomly choose low/high demand for each entry
inds = random_choice_0_1(num_nodes, num_nodes) with a probablity of p

# low allocations
A[inds] = uniform(low_range[0], low_range[1], sum(inds))

# high allocations
A[~inds] = uniform(high_range[0], high_range[1], sum(~inds))

fill_diagonal(A, 0.0) # no demand for self
```

B.3 BASELINES

In this paper, we compare PRAM against the following state-of-the-art MCF solutions:

• **DRL** (Valadarsky et al., 2017): It leverages deep reinforcement learning to replace explicit demand prediction with end-to-end optimization, mapping recent traffic demands to MCF configurations. Specifically, we train a policy neural network of the same size as PRAM's trainable parameters, using OpenAI's proximal policy optimization (PPO) algorithms (Schulman et al., 2017).

• **LP** (Gurobi Optimization, LLC, 2024): It solves the MCF optimization problem for all demands using linear programming (LP), in which Gurobi (version 11.0.3) optimization solver is employed.

• **POP** (Narayanan et al., 2021): It also decomposes large-scale granular flow problems into subproblems that are solved in parallel using an LP solver. Specifically, the entire topology is replicated k

Subtask Prompt Structure

Role : You are an expert in solving multi-commodity flow problems.

Task : Based on <|vision_start|><|image_pad|><|vision_end|> the all traversed links from a source node (marked in red) to others in the image and other information, determine candidate path weights for each source-destination pair in the near future. ** **Current Source node need to optimize is <i>** **.

Goal : ** **The objective is to minimize <maximum link utilization>** ** . Find a set of candidate path weights (split ratios) that minimizes the highest utilization across all links in the network.

Description : There are <num_nodes> nodes in the topology and <num_paths> shortest paths between each node pair are used as candidate paths. Note that the sum of the candidate path weights should equal 1 . All commodity demands mu-st be routed (conservation of flow), while weights must be positive floats. ····· (other descriptions)

Demands : The most recent demands from (<i>, 0) to (<i>, <num_nodes> - 1) are as follows.\n <x_1, x_2, ⋯, x_n>; and history mean demands from (<i>, 0) to (<i>, <num_nodes> - 1) are as follows. \n <m_1, m_2, ⋯, m_n>. ····· (other statistics)

Format : Make sure to follow the answer template. [source-0-path1-weight, ⋯, source-0-pathx-weight], [source-1-path1-weight, ⋯source-1-pathx-weight],⋯; And no allocation is needed for the source node to itself.

Figure 14: Example subtask prompt structure for maximum link utilization optimization.

times, with each replica assigned $\frac{1}{k}$ of the original link capacities. Commodity demands are then *randomly* distributed across these replicas, and each subproblem is solved independently in parallel. We set k according to the topology size: $k = 2$ for the five real-world datasets, $k = 12$ for four large-scale topologies other than KDL, and $k = 128$ for KDL.

• **LP-top** (Namyar et al., 2022): It implements a simple yet effective heuristic algorithm that is recently revealed as "demand pinning". It allocates the top 10% of demands using an LP solver and assigns the remaining demands to the shortest paths.

• $\sim$ **(pred)**: Since most of non-learning method (i.e., LP, POP, and LP-top) optimizes the solution with perfect knowledge of future demands, we consider their practical implementation based on predicted future demands. For simplicity and efficiency, we primarily use a non-parametric weighted moving average for prediction.

B.4 Candidate Path Selection

For each topology, unless otherwise specified, we use Yen's algorithm (Yen, 1971) to precompute up to **4 shortest paths** between every pair of nodes, which serve as candidate paths for multi-commodity flow allocation. If fewer than 4 paths exist, the last available path is repeated to ensure a total of 4.

C PRAM Implementation

We implement PRAM using PyTorch (Paszke et al., 2019) and HuggingFace Transformers (Wolf et al., 2019). Most experiments are conducted on a Linux server equipped with two NVIDIA A100 GPUs (80GB memory each). For model initialization, we employ Kaiming uniform initialization for the low-rank matrices. Additional implementation details are provided below.

C.1 Prompt Structure

We show the overall prompt structure used by PRAM for minimizing maximum link utilization in Figure 14. Our design can be roughly divided into three parts: task instructions, domain knowledge, and commodity information, as illustrated below. (i) Task instructions: The role section positions the MLM as an expert in multi-commodity flow problems, ensuring that subsequent reasoning is guided by a subtask-oriented perspective. The task and goal sections clearly specify the optimization

target—minimizing maximum link utilization—while constraining the solution to path weights that obey feasibility requirements (e.g., non-negativity and normalization); (ii) Domain knowledge: The description encodes domain knowledge, such as the number of nodes, candidate paths, and flow conservation principles, thereby narrowing the search space toward admissible allocations; (iii) Commodity information: The demands section conveys the actual commodity information, including most recent traffic volumes and their historical statistics, enabling the model to reason about both instantaneous and smoothed demand dynamics. Finally, the format clause enforces a deterministic answer template, which facilitates automatic parsing and validation of the policy head network. The template for the maximum flow and concurrent flow is essentially with the one outlined above.

C.2 Sub-image Plotting

We utilize Matplotlib (Hunter, 2007) and NetworkX (Hagberg et al., 2008) to generate visual representations of the sub-graph data. We draw all candidate paths grouped by source node. Each source gets one figure containing all its candidate paths (to all destinations). These visualizations are then provided to multimodal LLMs capable of processing image inputs (.png file with 240 dpi). Figure 15 shows all sub-image examples from the GÉANT topology, where its source node is marked in red and the link capacities are explicitly indicated in the figure.

C.3 Adaptation Algorithm

We next provide implementation details that complement the main text. We first outline how objectives (MLU, MTF, MCF) are computed efficiently as rewards using parallel matrix operations, followed by the multi-agent RL procedure adopted to fine-tune PRAM.

C.3.1 Objective (Reward) Computation

For MLU, we train PRAM to output all path weights (with Sigmoid non-linear as its final layer). The mapping between these weights and the resulted MLU can be represented using Algorithm 4. It is noteworthy that our reward computation relies on real samples (demand at future time steps), whereas the model only has access to historical information. No need for any loop operations, this mapping can be established through simple matrix operations in parallel. Regarding MTF and MCF, we force PRAM to directly predict allocated demand w_p for each candidate path (with ReLU non-linear as its final layer). As seen in Algorithm 5, we then scale $w_p = \frac{w_p}{\gamma}$, where $\gamma = \max\left(\max_{e\in\mathcal{E}} \sum_{p:e\in p} \frac{w_p}{c(e)}, 1\right)$. In this way, PRAM guarantees that no link capacity can be exceeded, and the computation is also fast.

Algorithm 4 Compute MLU in a PyTorch style

```
# w: path weights (split ratios)
# d: commodity demand vector
# C: link capacities
# PathtoEdge: path-to-edge incidence (1 if edge j in path i else 0)
# SDtoPath: SD-to-path incidence (1 if path j serves commodity i else 0)

# normalize path weights
total_w = SDtoPath @ w
w = w * (SDtoPath.T @ (1.0 / total_w))

# demand on paths and flow on edges
demand_on_paths = (SDtoPath.T @ d.T) * w
flow_on_edges = PathtoEdge.T @ demand_on_paths

# maximum link utilization
congestion = flow_on_edges / C
MLU = max(congestion)
```

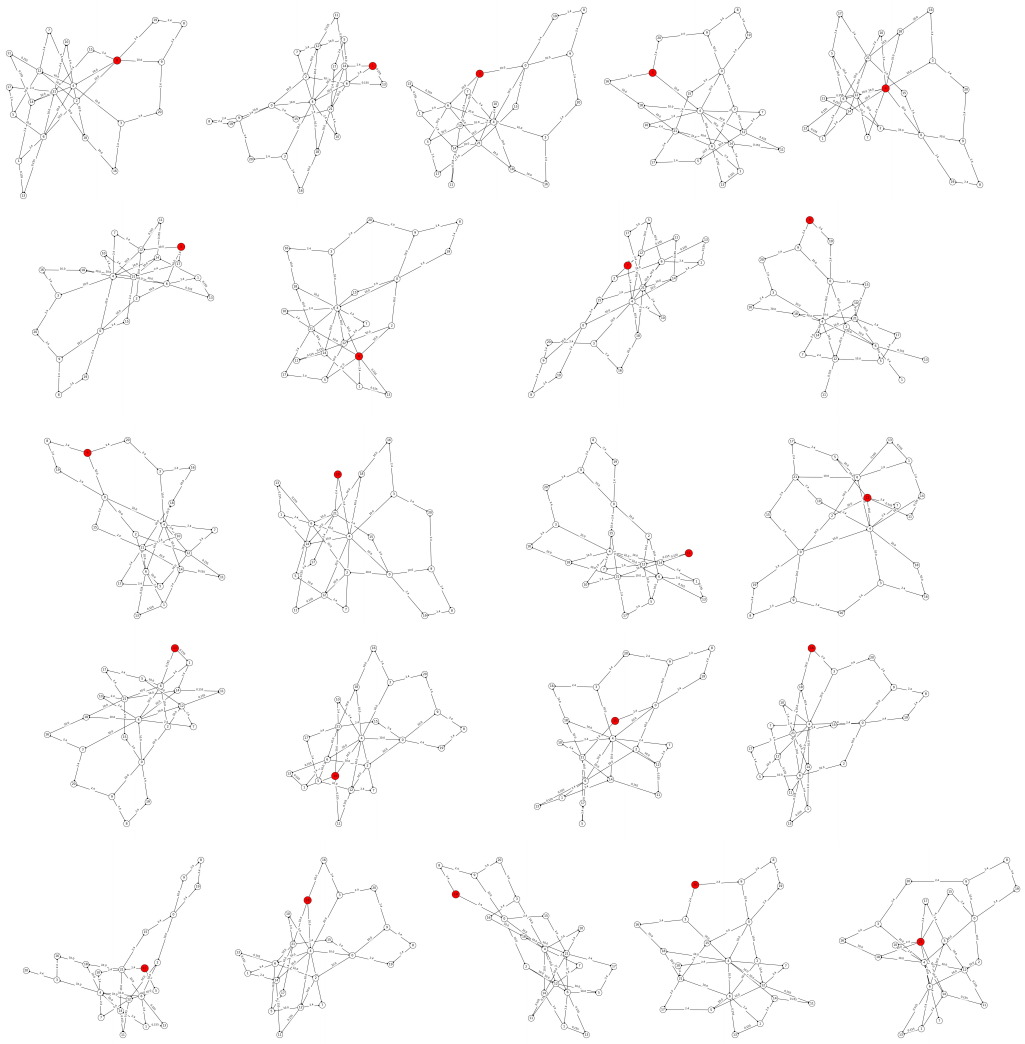


Figure 15: Examples of input sub-images on GÉANT topology. In each image, the starting point of the commodity flow is marked in red, only the links that exist in the candidate paths are drawn, and the link capacities are directly indicated on the links.

Algorithm 5 Compute MTF & MCF in a PyTorch style

```
# w_p: predicted demand vector
# d: commodity demand vector
# C: link capacities
# PathtoEdge: path-to-edge incidence (1 if edge j in path i else 0)
# SDtoPath: SD-to-path incidence (1 if path j serves SD i else 0)

# scale planned demands on each path
w_p_scaled = w_p / max((PathtoEdge.T @ w_p) / C)

# compute flow per SD pair, limited by demand
flow = min((SDtoPath @ w_p_scaled).T, d)

# total flow
MTF = sum(flow)

# maximum concurrent ratio
MCF = min(flow[d != 0] / d[d != 0])
```

C.3.2 Fine-tuning Details

As detailed in the main text (§ 3.3), PRAM opts for employing the adaptation framework of multi-agent RL. Since MCF allocations are deterministic, we adopt the standard approach of modeling $\pi_\theta(a_i|s_i)$ as a Gaussian distribution: during training, actions are sampled from this distribution, while at deployment the mean is used as the deterministic allocation. To be more specific, we employ a separate one-layer MLP to predict the mean, while keeping the standard deviation fixed[2]. After sampling from the corresponding Gaussian distribution, the log probability is calculated as: $\log \pi_\theta(a_i|s_i) = -\frac{1}{2}\left(\log(2\pi\sigma_i^2) + \frac{(a_i-\mu_i)^2}{\sigma_i^2}\right)$, which is implemented by torch.distributions.normal.log_prob. Algorithm 6 presents the pseudo-code for fine-tuning PRAM. It leverages the observation that allocations in MCF problems computed for a single time interval do not influence subsequent intervals (e.g., the demand matrices). This domain-specific insight enables us to simplify training by reducing the long-term return to a one-step return. Furthermore, note that the subproblem decomposition and the .png image transformation are performed only once at the beginning of the task, ensuring that they do not compromise the efficiency of model fine-tuning.

Algorithm 6 Fine-tuning PRAM in a PyTorch style

```
# Model: policy network (based on MLM)
# Optimizer: optimizer for Model
# episodes: sequence of training episodes

for observation in loader:
    # observation: pre-processed sub-images and text prompts
    loss = 0
    for step in episode:
        action, log_probability = Model(observation)     # select action
        advantage = compute_objective(action)    # get multi-agent reward

        # accumulate policy gradient
        loss += (- log_probability * advantage).mean()

    Optimizer.zero_grad()
    loss.backward()
    Optimizer.step()

    if EarlyStop and converged(): break
```

[2]We also experimented with learnable standard deviations in our early trials, but the results were suboptimal.

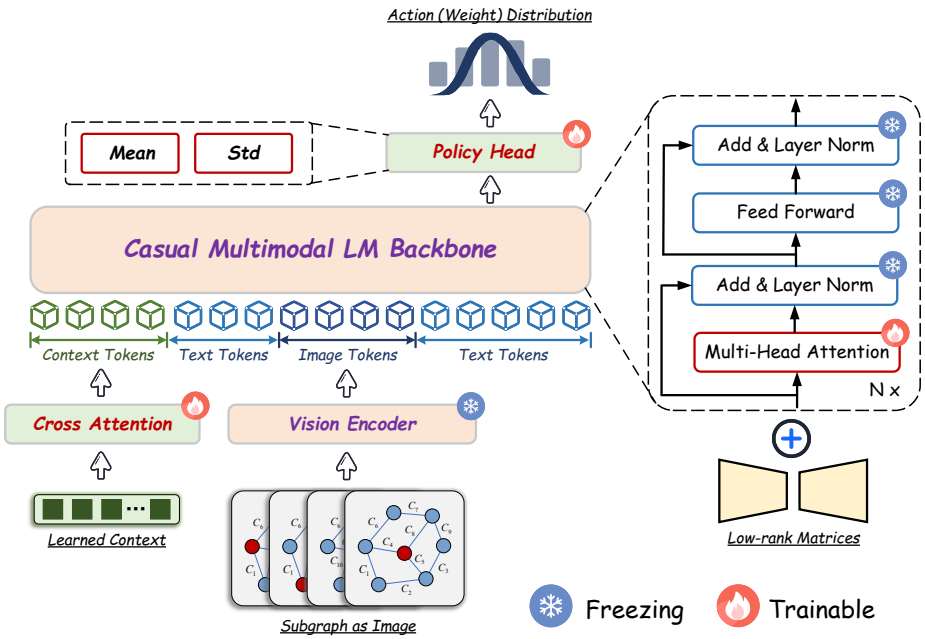


Figure 16: Model architecture of PRAM. In the figure, we use a Qwen2.5-VL-style MLM, which integrates of a vision encoder and a language model decoder to process multimodal inputs.

C.4 FRAMEWORK DETAILS

In this subsection, we first outline the architectural informantion of PRAM and provide an overview of our hyperparameter choices.

C.4.1 ARCHITECTURE

As shown in Figure 16, given a Qwen2.5-VL-style multimodal language model, we . Text and images are arranged in the order in which they appear in the original prompt content.

C.4.2 HYPERPARAMETER

Since PRAM primarily relies on the multimodal interface of the pre-trained model, most components (including token embedding dimensions) are fixed, leaving only a limited set of parameters to tune, as detailed below. First, we set the length of the learnable context proportional to the number of nodes in the topology, with a minimum of 48 tokens. Second, in the cross-attention module, the embedding dimensions of the Q, K, and V matrices are chosen from $[64, 128, 256, 512]$. Third, we take the final hidden state of the language model as input to the policy head, which consists of a single fully connected input layer, followed by two output heads (for mean and standard deviation), each implemented as a single fully connected layer (with output size $\approx$ number of paths $\times$ number of nodes). To speed up processing, we crop pre-saved sub-images to 128×128 pixels. For the sampling batch size, we select from $[8, 16, 32, 64]$. The sampling granularity is defined at the source-node level, meaning that allocations for larger topologies require multiple batches. For example, allocating resources for a 150-node topology requires at least three batches. Finally, the learning rate for fine-tuning is selected from $[10^{-3}, 5 \times 10^{-4}, 10^{-4}]$, and the adaptation of PRAM is limited to a maximum of 10 epochs with early stopping. Moreover, by default, the Adam optimizer (Kingma, 2014) is employed throughout all experiments.

D THEORETICAL RESULTS

In this section, we provide the detailed theoretical proofs that underpin the main results presented in the paper. Our goal is to establish rigorous guarantees for the proposed method, including its

correctness, convergence, and properties. To this end, we first formalize the necessary assumptions and notations, and then proceed to derive the main lemmas and theorems step by step. In particular, our arguments borrow insights from classical results on convex optimization and more recent analyses of LMs. Some key inspirations were excerpted from prior works of Bertsekas & Tsitsiklis (1996); Sutton et al. (1999); Foerster et al. (2018); Perry et al. (2023); Ahn et al. (2023).

D.1 Notations & Definitions

Notations Below, we list our notational conventions used in this section.

- $\|\cdot\|$ is the Euclidean norm.
- n denotes the total number of nodes, i.e., $|\mathcal{V}|$.
- c_{min} denotes the minimum link capacity.
- c_{max} denotes the maximum link capacity.
- $\mathcal{D}_{max}$ is an upper bound on the maximum demand between a source-destination pair.
- $\mathcal{D}_{min}$ is a lower bound on the maximum demand between a source-destination pair. Although it can be 0, to ensure the validity of the results we replace 0 with a value arbitrarily close to 0.
- p denotes the number of candidate paths interconnecting a source-destination pair.
- π denotes an allocation configuration, which can be represented as a vector with $n^2 \times p$ components. It is noteworthy that each source-destination element in π is itself a vector whose components sum to 1, specifying its weights $r_{(\cdot)}$ across at most p candidate paths.
- $\mathcal{L}_l$ denotes the maximum link utilization objective function, which, in its simplest form, is $\mathcal{L}_l(\pi) = \max_{e\in\mathcal{E}} \frac{f_e(\pi)}{c_e}$.
- $\mathcal{L}_t$ denotes the maximum total flow objective function. For convenience, we reformulate it as follows: Let the allocated flow on each link $x_e(\pi) = \sum_{e\ni p} x_p(\pi)$ where $x_p(\pi)$ is the allocated flow on path p. Then the actual flow on p is $f_p(\pi) = x_p(\pi) \cdot \min\left(\min_e\left(\frac{c_e}{x_e(\pi)}, 1\right)\right)$, and the actual flow successfully transmitted between s and t is $f_{s.t}(\pi) = \sum_{p\in P_{s,t}} f_p(\pi)$. Now, we can define $\mathcal{L}_t(\pi) = \sum_{s,t} f_{s,t}(\pi)$.
- $\mathcal{L}_c$ denotes the maximum concurrent flow objective function. Similar to $\mathcal{L}_t$, we reformulate it as follows: $\mathcal{L}_c(\pi) = \min\left(\left\{\frac{f_{s,t}(\pi)}{\mathcal{D}_{s,t}} \mid s,t \in \mathcal{V}\right\}\right)$.
- $\bar{R}(\cdot)$ denotes the long-term expected reward per step. Given a policy π, it is defined as $\bar{R}(\pi) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \mid s_0, \pi\right]$, where γ, s_0 and r_t denote discount rate, start state, and reward at time t respectively.
- $Q^{(\cdot)}(\cdot,\cdot)$ is an action-value function. Given a policy π, it is defined as $Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k} \mid s_t = s, a_t = a, \pi\right]$, where a_t is the action at time t.
- $d^{(\cdot)}(\cdot)$ is the stationary distribution of states. Given a policy π, it is defined as $d^{\pi}(s) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t \Pr\{s_t = s \mid s_0, \pi\}\right]$.

Definition 1. *(Convexity) A set C in a vector space is convex if for any two vectors $u, v \in C$ and $\alpha \in [0,1]$, we have that $\alpha u + (1-\alpha) v \in C$. A function $f : C \to \mathbb{R}$ is convex if $f(\alpha u + (1-\alpha) v) \leq \alpha f(u) + (1-\alpha) f(v)$.*

Definition 2. *(Lipschitzness) Let $C \subset \mathbb{R}^d$. A function $f : \mathbb{R}^d \to \mathbb{R}^k$ is ρ-Lipschitz over C if for every $w_1, w_2 \in C$ we have that $\|f(w_1) - f(w_2)\| \leq \rho \|w_1 - w_2\|$.*

Definition 3. *(Subgradients) Let S be an open convex set, and $f : S \to \mathbb{R}$ be a convex function. A vector v that satisfies $f(w) + \langle u - w, v\rangle \leq f(u)$ for any $u \in S$, is called a subgradient of f at w. The set of subgradients of f at w is called the differential set and denoted $\partial f(w)$.*

Definition 4. *(Gradient Descent, GD) Gradient descent is an iterative optimization method that, at each step, updates the solution by moving in the direction of the negative gradient of the objective function evaluated at the current point.*

Definition 5. *(Markov Decision Process, MDP) A MDP is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$. The state, action, and reward at each time t are denoted $s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$ and $r_t \in \mathbb{R}$ respectively. $P(s', s, a) = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$ denotes the transition kernel, $R(s, a)$ the expected rewards $\mathbb{E}[r_{t+1} \mid s_t = s, a_t = a]$, and $\gamma \in [0, 1)$ the discount factor. The agent's decision making procedure at each time is characterized by a policy, $\pi_\theta(a \mid s) = \Pr\{a_t = a \mid s_t = s, \theta\}$, where θ is a parameter vector. We also write just $\pi(s, a)$ for $\pi_\theta(a \mid s)$.*

D.2 Assumptions & Useful Lemmas

Assumption 1. *(Convex or Concave Objective[3]) All three objective functions, i.e., maximum link utilization, maximum total flow and maximum concurrent flow, are convex/concave with respect to the input candidate path weights.*

Assumption 2. *In each attention block, there exist token embeddings of observation–decision pairs (x_i, y_i) from the training set that are aligned with the embeddings of the learned context.*

Assumption 3. *In each attention block, the token embeddings of the observation–decision pairs (x_i, y_i) approximately satisfy a linear relation[4] $y_i = W x_i$, where $W \in \mathbb{R}^{N_y \times N_x}$ is a weight matrix.*

Lemma 2. *Let A be a nonempty open convex set, and let $f : A \to \mathbb{R}$ be a convex function. Let $\rho > 0$. Then the following are equivalent: ① f is ρ-Lipschitz continuous on A. ② At every point $w \in A$, every subgradient $v \in \partial f(w)$ satisfies $\|v\| \le \rho$.*

Proof. First, we assume that f is ρ-Lipschitz. Now choose $w \in A$ and $v \in \partial f(w)$. Since A is open, we have $u = w + \frac{v}{\|v\|}\epsilon \in A$ by choosing appropriate $\epsilon > 0$. We can write $\langle u - w, v\rangle = \epsilon\|v\|$ and $\|u - v\| = \epsilon$. Using Definition 3, i.e., $f(u) \ge f(w) + \langle u - w, v\rangle$, we obtain $f(u) - f(w) \ge \epsilon\|v\|$. Recall that $f(u) - f(w) \le \rho\|u - w\| = \rho\epsilon$, which means $\|v\| \le \rho$.

Second, we assume that $\|v\| \le \rho$. From the definition of the subgradient, $f(w) - f(u) \le \langle v, w - u\rangle$. Now using Cauchy-Schwartz inequality we have $f(w) - f(u) \le \|v\|\|w - u\| \le \rho\|w - u\|$. By repeating similar steps, also we have $f(u) - f(w) \le \rho\|w - u\|$. Therefore, $\|f(u) - f(w)\| \le \rho\|w - u\|$ — f is ρ-Lipschitz. □

Lemma 3. *Consider a convex, ρ-Lipschitz function f. Let w^* be a vector minimizing $f(w)$, in which $\|w\|$ is bounded by B, and GD algorithm with an update rule of the form $w^{(t+1)} = w^{(t)} - \eta v_t$ where $v_t \in \partial f(w)$, $w^{(1)} = 0$ and $\bar{w} = \frac{1}{T}\sum_{t=1}^{T} w^{(t)}$. If we run the algorithm on f for $T \ge \frac{B^2\rho^2}{\epsilon^2}$ steps with $\eta = \sqrt{\frac{B^2}{\rho^2 T}}$, then the output vector $\bar{w}$ satisfies $f(\bar{w}) - f(w^*) \le \epsilon, \forall \epsilon > 0$.*

Proof. We start by Jensen's inequality,

$$f(\bar{w}) - f(w^*) = f\left(\frac{1}{T}\sum_{t=1}^{T} w^{(t)}\right) - f(w^*) \le \frac{1}{T}\sum_{t=1}^{T} f\left(w^{(t)}\right) - f(w^*).$$

Thus we have $f(\bar{w}) - f(w^*) \le \frac{1}{T}\sum_{t=1}^{T}\left(f\left(w^{(t)}\right) - f(w^*)\right)$. using Definition 3, we can write

$$f(\bar{w}) - f(w^*) \le \frac{1}{T}\sum_{t=1}^{T}\left(f\left(w^{(t)}\right) - f(w^*)\right) \le \frac{1}{T}\sum_{t=1}^{T}\left\langle w^{(t)} - w^*, v_t\right\rangle.$$

To bound the right-hand side, we focus on $\left\langle w^{(t)} - w^*, v_t\right\rangle$. Given the positive strength η,

$$\begin{aligned}\left\langle w^{(t)} - w^*, v_t\right\rangle = \frac{1}{\eta}\left\langle w^{(t)} - w^*, \eta v_t\right\rangle &= \frac{1}{2\eta}\left(\left\|w^{(t)} - w^*\right\|^2 - \left\|w^{(t)} - w^* - \eta v_t\right\|^2 + \eta^2\left\|v_t\right\|^2\right) \\ &= \frac{1}{2\eta}\left(\left\|w^{(t)} - w^*\right\|^2 - \left\|w^{(t+1)} - w^*\right\|^2 + \eta^2\left\|v_t\right\|^2\right).\end{aligned}$$

[3]Note that this assumption can actually be relaxed slightly; as long as pseudo-convexity (Hazan et al., 2015) is satisfied, it does not affect the related results in this paper.

[4]Note that we only assume the attention inputs after token embedding are approximately linear, not that the original problem is linear. Indeed, a linear representation model (e.g., followed by an MLP) can express any nonlinear function (Ahn et al., 2023).

Then we have that

$$\sum_{t=1}^{T}\left\langle w^{(t)}-w^*, v_t\right\rangle = \frac{1}{2\eta}\sum_{t=1}^{T}\left(\left\|w^{(t)}-w^*\right\|^2 - \left\|w^{(t+1)}-w^*\right\|^2 + \eta^2\left\|v_t\right\|^2\right)$$
$$= \frac{1}{2\eta}\underbrace{\left\|w^{(1)}-w^*\right\|^2}_{w^{(1)}=0} - \frac{1}{2\eta}\underbrace{\left\|w^{(t+1)}-w^*\right\|^2}_{>0} + \frac{\eta}{2}\sum_{t=1}^{T}\left\|v_t\right\|^2$$
$$\leq \frac{1}{2\eta}\left\|w^*\right\|^2 + \frac{\eta}{2}\sum_{t=1}^{T}\left\|v_t\right\|^2$$

Now according to Lemma 2, $\|v_t\| \leq \rho$, and $\|w^*\| \leq B$. Plugging them along with $\eta = \sqrt{\frac{B^2}{\rho^2 T}}$ in the above inequality, we have $\frac{1}{T}\sum_{t=1}^{T}\left\langle w^{(t)}-w^*, v_t\right\rangle \leq \frac{B\rho}{\sqrt{T}}$. Therefore, for all $\epsilon > 0$, after $T \geq \frac{B^2\rho^2}{\epsilon^2}$ iterations of gradient descent, the objective function converges, i.e., $f(\bar{w}) - f(w^*) \leq \epsilon$, which concludes the proof. □

Lemma 4. *(Policy Gradient Theorem, Sutton et al. (1999)) For any MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$,*

$$\frac{\partial \bar{R}}{\partial \theta} = \sum_s d^\pi(s)\sum_a \frac{\partial \pi(s,a)}{\partial \theta} Q^\pi(s,a).$$

Proof. We begin with the following observation:

$$Q^\pi(s,a) = \mathbb{E}\left[r_{t+1} + \gamma\sum_{k=2}^{\infty}\gamma^{k-2}r_{t+k}\,\middle|\, s_t = s, a_t = a, \pi\right]$$
$$= \mathbb{E}\left(r_{t+1} \mid s_t = s, a_t = a\right) + \sum_{s'}\gamma\Pr\left\{s_{t+1} = s' \mid s_t = s, a_t = a\right\}\mathbb{E}\left[\sum_{k=2}^{\infty}\gamma^{k-2}r_{t+k}\,\middle|\, s_{t+1} = s', \pi\right]$$
$$= R(s,a) + \sum_{s'}\gamma P(s',s,a)\sum_a \pi(s',a)Q^\pi(s',a).$$

Defining the value function as $V^\pi(s) = \sum_a \pi(s,a)Q^\pi(s,a), \forall s \in \mathcal{S}$, we have that

$$\frac{\partial V^\pi(s)}{\partial \theta} = \frac{\partial}{\partial \theta}\sum_a \pi(s,a)Q^\pi(s,a)$$
$$= \sum_a\left[\frac{\partial \pi(s,a)}{\partial \theta}Q^\pi(s,a) + \pi(s,a)\frac{\partial}{\partial \theta}Q^\pi(s,a)\right]$$
$$= \sum_a\left[\frac{\partial \pi(s,a)}{\partial \theta}Q^\pi(s,a) + \pi(s,a)\frac{\partial}{\partial \theta}\left[R(s,a) + \sum_{s'}\gamma P(s',s,a)V^\pi(s')\right]\right]$$
$$= \sum_a\left[\frac{\partial \pi(s,a)}{\partial \theta}Q^\pi(s,a) + \pi(s,a)\sum_{s'}\gamma P(s',s,a)\frac{\partial V^\pi(s')}{\partial \theta}\right],$$

By expanding the recursion iteratively, i.e., $\frac{\partial V^\pi(s)}{\partial \theta} \to \frac{\partial V^\pi(s')}{\partial \theta} \to \frac{\partial V^\pi(s'')}{\partial \theta} \to \ldots$, we obtain

$$\frac{\partial V^\pi(s)}{\partial \theta} = \sum_x\sum_{k=0}^{\infty}\gamma^k Pr(s \to x, k, \pi)\sum_a \frac{\partial \pi(x,a)}{\partial \theta}Q^\pi(x,a),$$

where $\Pr(s \to x, k, \pi)$ denotes the probability of reaching state x from state s in exactly k steps under policy π. Also we have

$$V^\pi(s) = \sum_a \pi(s,a)\,\mathbb{E}\left[\sum_{k=1}^{\infty}\gamma^{k-1}r_{t+k} \mid s_t = s, a_t = a, \pi\right] = \mathbb{E}\left[\sum_{k=1}^{\infty}\gamma^{k-1}r_{t+k} \mid s_t = s, \pi\right].$$

Finally, the gradient of the expected discounted return $\bar{R}$ follows directly:

$$\begin{aligned}\frac{\partial \bar{R}}{\partial \theta} &= \frac{\partial}{\partial \theta}\mathbb{E}\left[\sum_{t=1}^{\infty}\gamma^{t-1}r_t \mid s_0, \pi\right] = \frac{\partial}{\partial \theta}V^{\pi}(s_0) \\ &= \sum_s \sum_{k=0}^{\infty}\gamma^k Pr(s_0 \to s, k, \pi)\sum_a \frac{\partial \pi(s,a)}{\partial \theta}Q^{\pi}(s,a) \\ &= \sum_s d^{\pi}(s)\sum_a \frac{\partial \pi(s,a)}{\partial \theta}Q^{\pi}(s,a).\end{aligned}$$

Hence, the proof is complete. □

Lemma 5. *Let $f_w : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ be a parametric approximation to $Q^{\pi}(s,a)$ with parameter w. If f_w is locally optimal and satisfies $\frac{\partial f_w(s,a)}{\partial w} = \frac{1}{\pi(s,a)}\frac{\partial \pi(s,a)}{\partial \theta}$, then $\frac{\partial \bar{R}}{\partial \theta} = \sum_s d^{\pi}(s)\sum_a \frac{\partial \pi(s,a)}{\partial \theta}f_w(s,a)$.*

Proof. We consider learning f_w by sampling trajectories under policy π and updating w via a squared-error loss: $\Delta w_t \propto \frac{\partial}{\partial w}\big[\hat{Q}^{\pi}(s_t,a_t) - f_w(s_t,a_t)\big]^2 \propto \big[\hat{Q}^{\pi}(s_t,a_t) - f_w(s_t,a_t)\big]\frac{\partial f_w(s_t,a_t)}{\partial w}$, where $\hat{Q}^{\pi}(s_t,a_t)$ denotes an unbiased estimator of $Q^{\pi}(s_t,a_t)$. At convergence to a local optimum, the update direction vanishes in expectation, yielding

$$\sum_s d^{\pi}(s)\sum_a \pi(s,a)\big[Q^{\pi}(s,a) - f_w(s,a)\big]\frac{\partial f_w(s,a)}{\partial w} = 0. \tag{4}$$

Substituting the assumption $\frac{\partial f_w(s,a)}{\partial w} = \frac{1}{\pi(s,a)}\frac{\partial \pi(s,a)}{\partial \theta}$, we obtain $\sum_s d^{\pi}(s)\sum_a \frac{\partial \pi(s,a)}{\partial \theta}\big[Q^{\pi}(s,a) - f_w(s,a)\big] = 0$. This shows that the approximation error of f_w is orthogonal to the gradient of the policy parameterization. Now recall the policy gradient theorem (Lemma 4) and subtract the orthogonality condition above, we obtain

$$\begin{aligned}\frac{\partial \bar{R}}{\partial \theta} &= \sum_s d^{\pi}(s)\sum_a \frac{\partial \pi(s,a)}{\partial \theta}Q^{\pi}(s,a) - \sum_s d^{\pi}(s)\sum_a \frac{\partial \pi(s,a)}{\partial \theta}\big[Q^{\pi}(s,a) - f_w(s,a)\big] \\ &= \sum_s d^{\pi}(s)\sum_a \frac{\partial \pi(s,a)}{\partial \theta}f_w(s,a).\end{aligned}$$

This establishes the claim. □

Lemma 6. *Let f be any differentiable function, and w_t be a sequence generated by a GD-style update rule $w_{t+1} = w_t + \eta_t v_t$, where for all t, v_t satisfies ① $c_1\|\nabla f(w_t)\|^2 \le -\nabla f(w_t)\, v_t$; ② $\|v_t\| \le c_2\|\nabla f(w_t)\|$, where c_1 and c_2 are some positive scalars. Assume that for some constant $L > 0$, for all $w, \bar{w} \in \mathbb{R}^n$, we have $\|\nabla f(w) - \nabla f(\bar{w})\| \le L\|w - \bar{w}\|$, and that $\eta_t \to 0$, $\sum_{t=0}^{\infty}\eta_t = \infty$. Then either $f(w_t) \to -\infty$ or else $f(w_t)$ converges to a finite value and $\lim_{t\to\infty}\nabla f(w_t) = 0$.*

Proof. We begin by establishing a standard descent inequality. Fix $w, z \in \mathbb{R}^n$, and define $g(\xi) = f(w + \xi z)$ for $\xi \in [0,1]$. By the chain rule, $g'(\xi) = \langle z, \nabla f(w + \xi z)\rangle$. Applying the fundamental theorem of calculus,

$$\begin{aligned}f(w+z) - f(w) = g(1) - g(0) &= \int_0^1 g'(\xi)\, d\xi = \int_0^1 \langle z, \nabla f(w+\xi z)\rangle\, d\xi \\ &= \langle z, \nabla f(w)\rangle + \int_0^1 \langle z, \nabla f(w+\xi z) - \nabla f(w)\rangle\, d\xi \\ &\le \langle z, \nabla f(w)\rangle + \int_0^1 \|z\| \cdot \|\nabla f(w+\xi z) - \nabla f(w)\|\, d\xi \\ &\le \langle z, \nabla f(w)\rangle + \|z\|\int_0^1 L\xi\|z\|\, d\xi \\ &= \langle z, \nabla f(w)\rangle + \frac{L}{2}\|z\|^2.\end{aligned}$$

Now set $z = \eta_t v_t$. Using also condition ① and ②, we can write

$$\begin{aligned} f(w_{t+1}) &\le f(w_t) + \langle \eta_t v_t, \nabla f(w_t)\rangle + \frac{L}{2}\|\eta_t v_t\|^2 = f(w_t) + \eta_t \langle v_t, \nabla f(w_t)\rangle + \frac{L}{2}\eta_t^2 \|v_t\|^2 \\ &\le f(w_t) - \eta_t c_1 \|\nabla f(w_t)\|^2 + \frac{L}{2}\eta_t^2 c_2^2 \|\nabla f(w_t)\|^2 \\ &= f(w_t) - \eta_t \left(c_1 - \frac{L c_2^2}{2}\eta_t\right) \|\nabla f(w_t)\|^2. \end{aligned}$$

Since $\eta_t \to 0$, there exists a positive constant c such that for all t beyond some index $\bar{t}$, $\bar{t}$, $f(w_{t+1}) \le f(w_t) - \eta_t c \|\nabla f(w_t)\|^2$. This inequality implies that for $t \ge \bar{t}$, the sequence $f(w_t)$ is monotonically non-increasing. Consequently, either $f(w_t) \to -\infty$, in which case the proof is complete, or $f(w_t)$ converges to a finite limit. We now proceed under the latter.

To prove that $\lim_{t\to\infty} \nabla f(w_t) = 0$, suppose the contrary, namely that $\limsup_{t\to\infty} |\nabla f(w_t)| > 0$. Then there exists $\epsilon > 0$ such that $|\nabla f(w_t)| < \epsilon/2$ for infinitely many t, while $|\nabla f(w_t)| > \epsilon$ for infinitely many t. Hence, we can extract an infinite subset of indices $\mathcal{T}$ such that for each $t \in \mathcal{T}$, there exists $i(t) > t$ with

$$\|\nabla f(w_t)\| < \epsilon/2, \quad \|\nabla f(w_{i(t)})\| > \epsilon, \quad \epsilon/2 \le \|\nabla f(w_i)\| \le \epsilon, \quad \text{if } t < i < i(t).$$

Meanwhile, because

$$\|\nabla f(w_{t+1})\| - \|\nabla f(w_t)\| \le \|\nabla f(w_{t+1}) - \nabla f(w_t)\| \le L\|\underbrace{w_{t+1} - w_t}_{=\eta_t v_t}\| \le \eta_t L c_2 \|\nabla f(w_t)\|,$$

it follows that for all $t \in \mathcal{T}$ that are sufficiently large so that $\gamma_t L c_2 < 1$, we have $\frac{\epsilon}{4} \le \|\nabla f(w_t)\|$; otherwise the condition $\epsilon/2 \le |\nabla f(w_{t+1})|$ would be contradicted. Without loss of generality, we assume that this relation holds for all $t \in \mathcal{T}$.

We have for all $t \in \mathcal{T}$, using the condition $\|s_t\| \le c_2 \|\nabla f(w_t)\|$ and the Lipschitz condition:

$$\begin{aligned} \frac{\epsilon}{2} &\le \|\nabla f(w_{i(t)})\| - \|\nabla f(w_t)\| \le \|\nabla f(w_{i(t)}) - \nabla f(w_t)\| \\ &\le L\|w_{i(t)} - w_t\| \le L \sum_{i=t}^{i(t)-1} \eta_i \|v_i\| \\ &\le L c_2 \sum_{i=t}^{i(t)-1} \eta_i \|\nabla f(w_i)\| \le L c_2 \epsilon \sum_{i=t}^{i(t)-1} \eta_i, \end{aligned}$$

which gives $\sum_{i=t}^{i(t)-1} \eta_i \ge \frac{1}{2Lc_2}$. Now using $\bar{t}$, $f(w_{t+1}) \le f(w_t) - \eta_t c \|\nabla f(w_t)\|^2$, for sufficiently large $t \in \mathcal{T}$, and the relation $\|\nabla f(w_i)\| \ge \epsilon/4$ for $i = t, t+1, \ldots, i(t)-1$, we can write

$$f(w_{i(t)}) \le f(w_t) - c\left(\frac{\epsilon}{4}\right)^2 \sum_{i=t}^{i(t)-1} \eta_i, \quad \forall\, t \in \mathcal{T}.$$

Since $f(r_t)$ converges to a finite value, the preceding relation implies that $\lim_{t\to\infty,\, t\in\mathcal{T}} \sum_{i=t}^{i(t)-1} \eta_i = 0 < \frac{1}{2Lc_2}$. Therefore, the previous assumption is false, and we conclude that $\lim_{t\to\infty} \nabla f(w_t) = 0$. This concludes the proof. □

D.3 Proof of Theorem 1

Proof. The basic idea of this proof is to show that the configuration vector is upper bounded and that all three objective functions are ρ-Lipschitz. Then, since we assume the problem is convex, we can directly obtain the final result via Lemma 3. First, according to the definition of π, configuration $\|\pi\|$ is clearly bounded by $\sqrt{n^2 \times p} \doteq B$. Below, we discuss the Lipschitz properties of the functions.

Function $\mathcal{L}_l$'s Lipschitz We can observe that f_e is linear in the path weights and so in π: $f_e = \sum_{s,t\in V}\sum_{p\in P_{s,t}}\sum_{e\ni p}\mathcal{D}_{s,t}\cdot r_p \le \sum_{s,t\in V}\sum_{p\in P_{s,t}}\sum_{e\ni p}\mathcal{D}_{max}\cdot r_p$. So we have

$$\left\|\mathcal{L}_l\left(\pi^{(i)}\right)-\mathcal{L}_l\left(\pi^{(j)}\right)\right\| = \left\|\max_e \frac{f_e\left(\pi^{(i)}-\pi^{(j)}\right)}{c_e}\right\| \le \frac{\mathcal{D}_{max}}{c_{min}}\left\|\pi^{(i)}-\pi^{(j)}\right\|.$$

Therefore, $\mathcal{L}_l$ is ρ-Lipschitz, with $\rho = \frac{\mathcal{D}_{max}}{c_{min}}$.

Function $\mathcal{L}_t$'s Lipschitz From the definition of the maximum total flow, one should notice that $\mathcal{L}_t$ is the sum of f_p. Therefore, it suffices to show that f_p is ρ-Lipschitz, in which case $\mathcal{L}_t$ is simply $\sum_p \rho$-Lipschitz. To focus on $\left\|f_p\left(\pi^{(i)}\right)-f_p\left(\pi^{(j)}\right)\right\|$, we distinguish between two cases according to the value of $\min\left(\min_e\left(\frac{c_e}{x_e(\pi)}\right)\right) \doteq \psi$ and make the following observation: $\psi \le \frac{c_e}{x_e(\pi)} \le \frac{c_{max}}{x_p(\pi)}$.

- $\psi\left(\pi^{(j)}\right)=1$. We have that

$$\left\|f_p\left(\pi^{(i)}\right)-f_p\left(\pi^{(j)}\right)\right\| = \left\|x_p\left(\pi^{(i)}\right)\psi\left(\pi^{(i)}\right)-x_p\left(\pi^{(j)}\right)\right\|.$$

Without loss of generality, we assume that $f_p\left(\pi^{(i)}\right) \ge f_p\left(\pi^{(j)}\right)$. Then

$$\left\|f_p\left(\pi^{(i)}\right)-f_p\left(\pi^{(j)}\right)\right\| = x_p\left(\pi^{(i)}\right)\psi\left(\pi^{(i)}\right)-x_p\left(\pi^{(j)}\right) \le x_p\left(\pi^{(i)}\right)-x_p\left(\pi^{(j)}\right).$$

Also note that $x_p(\pi)=\mathcal{D}_{s,t}\cdot r_p$, where $r_p\in\pi$. Therefore,

$$\left\|f_p\left(\pi^{(i)}\right)-f_p\left(\pi^{(j)}\right)\right\| \le \mathcal{D}_{max}\left\|\pi^{(i)}-\pi^{(j)}\right\| \le 2\cdot\frac{c_{max}\cdot\mathcal{D}_{max}}{c_{min}}\left\|\pi^{(i)}-\pi^{(j)}\right\|.$$

- $\psi\left(\pi^{(j)}\right)<1$. Under the same assumption as above, we additionally set $\psi\left(\pi^{(j)}\right)=\frac{c_{e_j}}{x_{e_j}\left(\pi^{(j)}\right)}$.

$$\begin{aligned}
&\left\|f_p\left(\pi^{(i)}\right)-f_p\left(\pi^{(j)}\right)\right\| \\
&= x_p\left(\pi^{(i)}\right)\cdot\psi\left(\pi^{(i)}\right)-x_p\left(\pi^{(j)}\right)\cdot\psi\left(\pi^{(j)}\right) \\
&= \psi\left(\pi^{(i)}\right)\cdot x_p\left(\pi^{(j)}\right)-\psi\left(\pi^{(j)}\right)\cdot x_p\left(\pi^{(j)}\right)+\psi\left(\pi^{(i)}\right)\cdot x_p\left(\pi^{(i)}\right)-\psi\left(\pi^{(i)}\right)\cdot x_p\left(\pi^{(j)}\right) \\
&= \psi\left(\pi^{(i)}\right)\cdot\psi\left(\pi^{(j)}\right)\cdot x_p\left(\pi^{(j)}\right)\cdot\left[\frac{1}{\psi\left(\pi^{(j)}\right)}-\frac{1}{\psi\left(\pi^{(i)}\right)}\right]+\psi\left(\pi^{(i)}\right)\cdot\left[x_p\left(\pi^{(i)}\right)-x_p\left(\pi^{(j)}\right)\right] \\
&\le \psi\left(\pi^{(i)}\right)\cdot\psi\left(\pi^{(j)}\right)\cdot x_p\left(\pi^{(j)}\right)\cdot\left[\frac{x_{e_j}\left(\pi^{(j)}\right)}{c_{e_j}}-\frac{x_{e_j}\left(\pi^{(i)}\right)}{c_{e_j}}\right]+\psi\left(\pi^{(i)}\right)\cdot\left[x_p\left(\pi^{(i)}\right)-x_p\left(\pi^{(j)}\right)\right] \\
&\le \psi\left(\pi^{(i)}\right)\cdot\psi\left(\pi^{(j)}\right)\cdot x_p\left(\pi^{(j)}\right)\cdot\left\|\frac{x_{e_j}\left(\pi^{(j)}\right)}{c_{e_j}}-\frac{x_{e_j}\left(\pi^{(i)}\right)}{c_{e_j}}\right\|+\psi\left(\pi^{(i)}\right)\cdot\left\|x_p\left(\pi^{(i)}\right)-x_p\left(\pi^{(j)}\right)\right\| \\
&\le \psi\left(\pi^{(j)}\right)\cdot x_p\left(\pi^{(j)}\right)\cdot\left\|\frac{x_{e_j}\left(\pi^{(j)}\right)}{c_{e_j}}-\frac{x_{e_j}\left(\pi^{(i)}\right)}{c_{e_j}}\right\|+\left\|x_p\left(\pi^{(i)}\right)-x_p\left(\pi^{(j)}\right)\right\| \\
&\le \frac{c_{max}}{x_p\left(\pi^{(j)}\right)}\cdot x_p\left(\pi^{(j)}\right)\cdot\left\|\frac{x_{e_j}\left(\pi^{(j)}\right)}{c_{e_j}}-\frac{x_{e_j}\left(\pi^{(i)}\right)}{c_{e_j}}\right\|+\left\|x_p\left(\pi^{(i)}\right)-x_p\left(\pi^{(j)}\right)\right\| \\
&\le \frac{c_{max}}{c_{min}}\cdot\left\|x_{e_j\left(\pi^{(j)}\right)}-x_{e_j\left(\pi^{(i)}\right)}\right\|+\left\|x_p\left(\pi^{(i)}\right)-x_p\left(\pi^{(j)}\right)\right\|,
\end{aligned}$$

where the second inequality is derived using $|a+b|\le|a|+|b|$. Recall that $x_e(\pi)=\sum_{e\ni p}x_p(\pi)=\sum_{e\ni p}\mathcal{D}_{s,t}\cdot r_p$, and $x_p\left(\pi^{(i)}\right)-x_p\left(\pi^{(j)}\right)\le\mathcal{D}_{max}\left\|\pi^{(i)}-\pi^{(j)}\right\|$. Therefore,

$$\begin{aligned}
\left\|f_p\left(\pi^{(i)}\right)-f_p\left(\pi^{(j)}\right)\right\| &\le \frac{c_{max}}{c_{min}}\cdot\mathcal{D}_{max}\left\|\pi^{(i)}-\pi^{(j)}\right\|+\mathcal{D}_{max}\left\|\pi^{(i)}-\pi^{(j)}\right\| \\
&\le 2\cdot\frac{c_{max}\cdot\mathcal{D}_{max}}{c_{min}}\left\|\pi^{(i)}-\pi^{(j)}\right\|.
\end{aligned}$$

By combining two cases, we can conclude that the function f_p is ρ-Lipschitz, where ρ is at most $\frac{2\cdot c_{max}\cdot\mathcal{D}_{max}}{c_{min}}$. As a result, $\mathcal{L}_t=\sum_{s,t}\sum_{p\in P_{s,t}}f_p$ is Lipschitz, and its Lipschitz constant is at most $\sum_{s,t}\sum_{p\in P_{s,t}}\frac{2\cdot c_{max}\cdot\mathcal{D}_{max}}{c_{min}}$.

Function $\mathcal{L}_c$'s Lipschitz The proof for $\mathcal{L}_c$ follows essentially the same idea as for $\mathcal{L}_t$, as both focus on the Lipschitz property of f_p. Hence, we can directly reuse the result established in the proof of $\mathcal{L}_t$ for f_p. And $\mathcal{L}_c$ is also guaranteed to be ρ-Lipschitz, with ρ at most $\sum_{p\in P_{s,t}} \frac{2\cdot\mathcal{D}_{max}\cdot c_{max}}{\mathcal{D}_{min}\cdot c_{min}}$.

We have now established that all three objective functions $\mathcal{L}$ are ρ-Lipschitz and convex, and that $\|\pi\|$ is bounded by B. Finally, by Lemma 3, we can write: For every $\epsilon > 0$, if we run the algorithm on $\mathcal{L}$ for finite T (at least $\frac{B^2\rho^2}{\epsilon^2}$) steps with $\eta = \sqrt{\frac{B^2}{\rho^2 T}}$, then the output vector $\bar{\pi}$ satisfies $f(\bar{\pi}) - f(\pi^*) \leq \epsilon$, which completes the proof. □

D.4 Proof of Lemma 1

Proof. We consider the following expected policy gradient (Foerster et al., 2018) of PRAM:

$$g = \mathbb{E}_\pi\left[\sum_i A_i(s,a)\,\nabla_\theta \log \pi_\theta(a_i \mid s_i)\right], \quad \text{where} \quad A_i(s,a) = R(s,a) - \sum_{a_i'} \pi_\theta(a_i' \mid s_i)\, R\big(s,(a_{-i},a_i')\big).$$

Therefore,

$$g = \underbrace{-\sum_{s,a} p(s)\pi_\theta(a \mid s) \sum_i \left(\sum_{a_i'} \pi_\theta(a_i' \mid s_i)\, R(s,(a_{-i},a_i'))\right) \nabla_\theta \log \pi_\theta(a_i \mid s_i)}_{g_1}$$
$$+ \underbrace{\sum_{s,a} p(s)\pi_\theta(a \mid s) \sum_i R(s,a)\,\nabla_\theta \log \pi_\theta(a_i \mid s_i)}_{g_2}.$$

Vanishing of g_1. For each agent i, the corresponding component $g_1^{(i)}$ can be written as

$$g_1^{(i)} = \sum_s p(s) \sum_{a_{-i}} \pi_\theta(a_{-i} \mid s_{-i}) \sum_{a_i} \pi_\theta(a_i \mid s_i) \left(\sum_{a_i'} \pi_\theta(a_i' \mid s_i) R\big(s,(a_{-i},a_i')\big)\right) \nabla_\theta \log \pi_\theta(a_i \mid s_i)$$
$$= \sum_{s,a_{-i}} p(s)\pi_\theta(a_{-i} \mid s_{-i}) B_i(s,a_{-i}) \left(\sum_{a_i} \pi_\theta(a_i \mid s_i)\,\nabla_\theta \log \pi_\theta(a_i \mid s_i)\right),$$

where the inner term $B_i(s,a_{-i}) := \sum_{a_i'} \pi_\theta(a_i' \mid s_i)\, R\big(s,(a_{-i},a_i')\big)$ is independent of a_i. Since

$$\sum_{a_i} \pi_\theta(a_i \mid s_i)\,\nabla_\theta \log \pi_\theta(a_i \mid s_i) = \sum_{a_i} \nabla_\theta \pi_\theta(a_i \mid s_i) = \nabla_\theta \left(\sum_{a_i} \pi_\theta(a_i \mid s_i)\right) = \nabla_\theta(1) = 0,$$

it follows that $g_1^{(i)} = 0$ for all i, and hence $g_1 = \sum_i g_1^{(i)} = 0$.

Analysis of g_2. As discussed in §4, under our MCF setting (decoupling of rewards and states), the long-term return satisfies $Q^\pi(s,a) = R(s,a)$. Thus, $g_2^{(i)}$ can be rewritten as $g_2^{(i)} = \mathbb{E}_\pi\big[Q^\pi(s,a)\,\nabla_\theta \log \pi_\theta(a_i \mid s_i)\big]$, which coincides with the standard policy gradient formulation (Williams, 1992). Moreover, by Lemma 4, this admits the equivalent expression $g_2^{(i)} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta}\, Q^\pi(s,a)$.

Suppose f_w is a function approximator of Q^π, parameterized by w and locally optimal. Without loss of generality, assume $\frac{\partial f_w(s,a)}{\partial w} = \frac{1}{\pi(s,a)} \frac{\partial \pi(s,a)}{\partial \theta}$. Under this construction, Lemma 5 implies $g_2^{(i)} = \frac{\partial \bar{R}}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s,a)}{\partial \theta}\, f_w(s,a)$.

We may define the update direction $v_k = \sum_s d^{\pi_{(k)}}(s) \sum_a \frac{\partial \pi_{(k)}(s,a)}{\partial \theta}\, f_{w_{(k)}}(s,a)$, since the $\theta^{(k)}$-update follows the policy gradient direction. The boundedness of $\nabla^2_\theta R(s,a)$ ensures, via the mean value theorem, that $\nabla_\theta \bar{R}(s,a)$ is Lipschitz. Furthermore, v_k satisfies the required conditions: $\|\nabla_{\theta^{(k)}} \bar{R}(s,a)\|^2 \leq -\nabla_{\theta^{(k)}} \bar{R}(s,a)\, v_k$ and $\|v_k\| \leq \|\nabla_{\theta^{(k)}} \bar{R}(s,a)\|$, since v_k is an unbiased

estimator of the policy gradient. Together with the step-size requirements and bounded function, these are precisely the conditions needed to invoke Lemma 6, which guarantees convergence to a local optimum, i.e., $\lim_{k\to\infty} g_2 = \lim_{k\to\infty} \sum_i \nabla_{\theta^{(k)}} \bar{R}(s,a) = 0$.

Finally, we have $\lim_{k\to\infty} g = 0 + \lim_{k\to\infty} g_2 = 0$, which completes the proof. □

D.5 Proof of Theorem 2

Proof. Under Assumptions 2 and 3, we first analyze the dynamics induced by gradient descent. Consider the reference linear model $y(x) = Wx$ and a fine-tuning dataset consisting of input embeddings x_i and corresponding decision embeddings y_i. The training objective is the squared-error loss $\mathcal{L}(W) = \frac{1}{2N}\sum_{i=1}^{N} \|Wx_i - y_i\|^2$. One step of gradient descent with learning rate η updates the weights as $\Delta W = -\eta \nabla_W \mathcal{L}(W) = -\frac{\eta}{N}\sum_{i=1}^{N}(Wx_i - y_i)x_i^\top$. Substituting into the loss gives

$$\mathcal{L}(W + \Delta W) = \frac{1}{2N}\sum_{i=1}^{N} \big\|(W + \Delta W)x_i - y_i\big\|^2 = \frac{1}{2N}\sum_{i=1}^{N} \big\|Wx_i - (y_i - \Delta y_i)\big\|^2.$$

Now consider the linear self-attention update on the token e_j,

$$e_j \;\leftarrow\; e_j + \textsc{Att}(\{e_1, \ldots, e_N\}) = e_j + \sum_h P_h V_h K_h^\top q_{h,j}.$$

The key idea is to construct W_K, W_Q, W_V and the projection F such that a single Transformer update on token e_j replicates the gradient-descent dynamics: $e_j \;\leftarrow\; (x_j, y_j) + (0, -\Delta W x_j) = (x_j, y_j - \Delta y_j)$.

Specifically, consider the block-structured weight matrices for one attention head:

$$W_K = W_Q = \begin{pmatrix} I_x & 0 \\ 0 & 0 \end{pmatrix}, \qquad W_V = \begin{pmatrix} 0 & 0 \\ W_0 & -I_y \end{pmatrix},$$

where I_x and I_y are identity matrices of size N_x and N_y, and $W_0 \in \mathbb{R}^{N_y \times N_x}$ is the reference model. We also set $P = \frac{\eta}{N} I$, where I is the identity matrix of dimension $N_x + N_y$. With this construction (Ahn et al., 2023),

$$\textsc{Att}(\{e_1, \ldots, e_N\}) = \sum_h P_h \sum_i v_{h,i} \otimes k_{h,i} q_{h,j} = \sum_h P_h W_{h,V} \sum_i e_{h,i} \otimes e_{h,i} W_{h,K}^\top W_{h,Q} e_j,$$

and we obtain the following update dynamics:

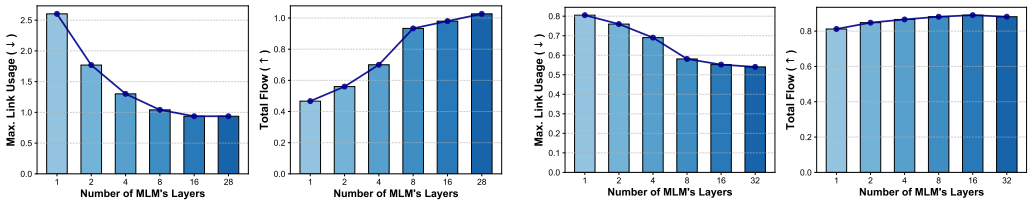
$$\begin{aligned}
\begin{pmatrix} x_j \\ y_j \end{pmatrix} \;\leftarrow\; & \begin{pmatrix} x_j \\ y_j \end{pmatrix} + \frac{\eta}{N} I \sum_{i=1}^{N} \left(\begin{pmatrix} 0 & 0 \\ W_0 & -I_y \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \right) \otimes \left(\begin{pmatrix} I_x & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} \right) \begin{pmatrix} I_x & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_j \\ y_j \end{pmatrix} \\
&= \begin{pmatrix} x_j \\ y_j \end{pmatrix} + \frac{\eta}{N} I \sum_{i=1}^{N} \left(\begin{pmatrix} 0 \\ W_0 x_i - y_i \end{pmatrix} \otimes \begin{pmatrix} x_i \\ 0 \end{pmatrix} \begin{pmatrix} x_j \\ 0 \end{pmatrix} \right) \\
&= \begin{pmatrix} x_j \\ y_j \end{pmatrix} + \begin{pmatrix} 0 \\ -\Delta W x_j \end{pmatrix}.
\end{aligned}$$

Thus, each token $e_j = (x_j, y_j)$, including the query token $e_{N+1} = e_{\text{test}} = (x_{\text{test}}, -W_0 x_{\text{test}})$, follows exactly the gradient-descent update. By stacking multiple attention blocks and heads, we can therefore simulate multiple steps of gradient descent, completing the proof. □

E Additional Empirical Results

In this section, we present supplementary experiments to further explore the effectiveness of PRAM. We first analysis the impact of multimodal language model's layers and backbones. Then, we investigate the performance of PRAM with different synthetic data distribution.

E.1 Impact of MLM's Backbone Model



(a) Abilene (b) GÉANT

Figure 18: Performance comparison with different number of Transformer layers in the MLM.

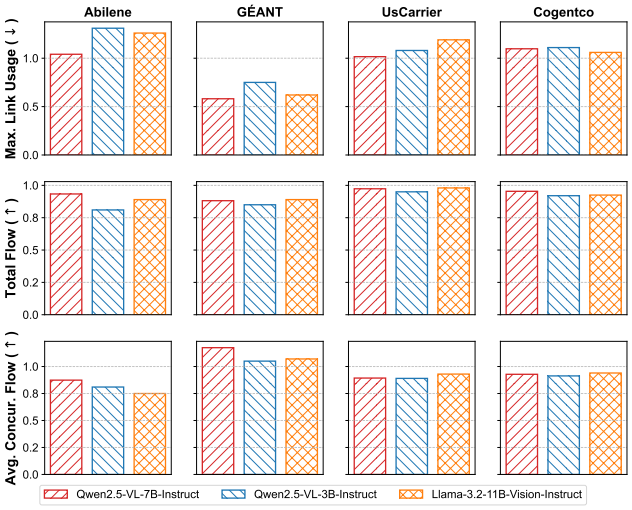


Figure 17: Performance comparison with different MLMs.

During our main experiments, we kept the LM backbone fixed to simplify presentation and ensure fair comparison. In this subsection, however, besides Qwen2.5-VL-7B-Instruct, we further report PRAM's performance with alternative pre-trained backbones of different scales, including the larger Llama-3.2-11B-Vision-Instruct (Meta, 2024), and the smaller Qwen2.5-VL-3B-Instruct (Bai et al., 2025). As described in our experimental setup, all MLMs are truncated to 8 layers. Notably, there exists a substantial architectural difference between Llama and Qwen: while Qwen encodes visual features as part of the input "prompt," Llama integrates them across multiple layers via cross-attention. To maintain comparability, we truncate only those layers of Llama-3.2-11B-Vision-Instruct's language model that do not involve cross-attention.

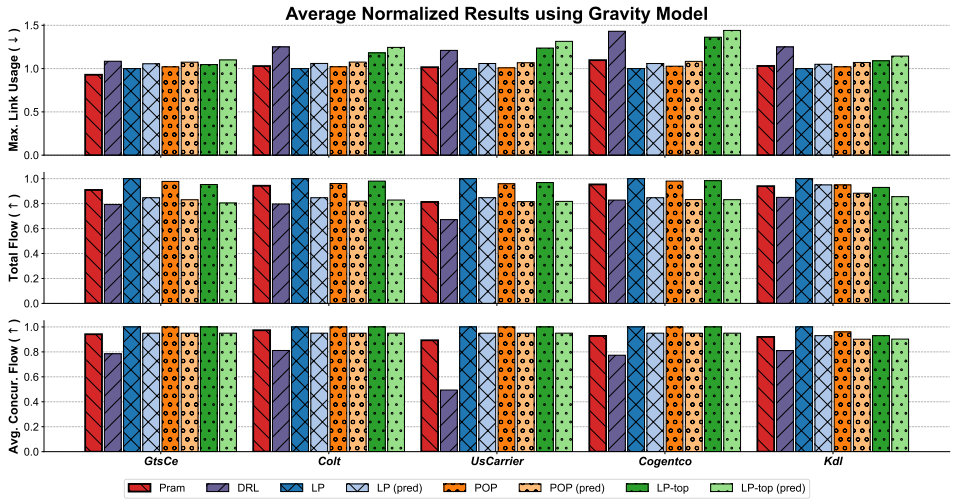
The results, presented in Figure 17, show that most MLMs perform well on the target MCF problems. Qwen2.5-VL-7B-Instruct consistently performs no worse than Qwen2.5-VL-3B-Instruct, which aligns with our expectations, as more parameters often lead to stronger understanding and reasoning capabilities. But interestingly, despite having fewer parameters, Qwen2.5-VL-7B-Instruct demonstrates stronger planning ability than the larger Llama-3.2-11B-Vision-Instruct. We attribute this to Qwen's training strategy, which emphasizes numerical reasoning and step-by-step problem solving, whereas Llama, although powerful in language generation, is less specialized in mathematical domains. This suggests that model scale alone does not guarantee superior performance on the MCF problem. Therefore, we choose Qwen2.5-VL-7B-Instruct as the default backbone for PRAM.

E.2 Impact of Number of MLM's Layers

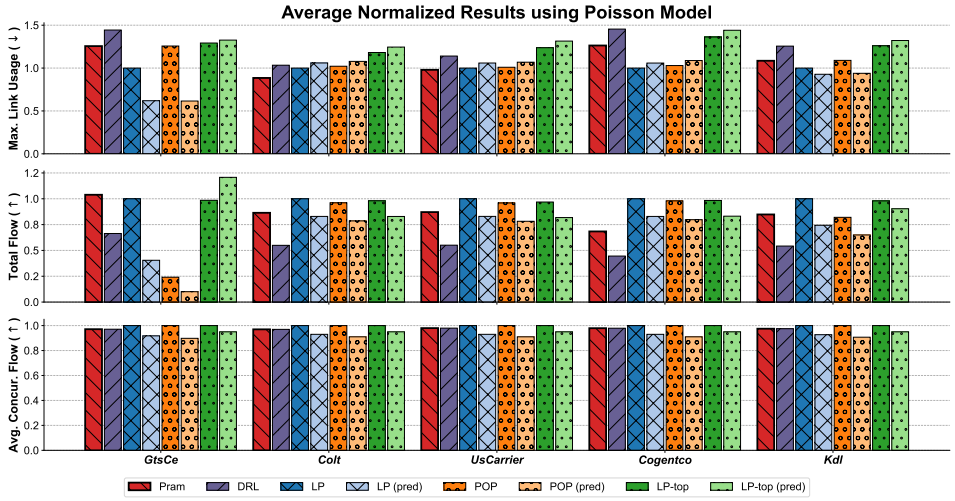
In Figure 18, we show how PRAM's link utilization and throughput (total flow) change with the number of Transformer layers across different datasets. The results clearly indicate that performance improves as the number of layers increases, with the effect being particularly pronounced on the Abilene dataset. This finding is consistent with the sharp deterioration observed in our main experiments when the backbone is entirely removed (i.e., # of Transformer layers = 0). We attribute this to the heterogeneous allocation patterns in Abilene, which demand stronger representation capacity (i.e., deeper models) to capture effectively. Nonetheless, the performance gain diminishes beyond 8 layers. To strike a balance between fine-tuning efficiency and performance, we adopt 8 layers as the default setting in the main experiments.

E.3 Results w.r.t. Synthetic Data Distributions

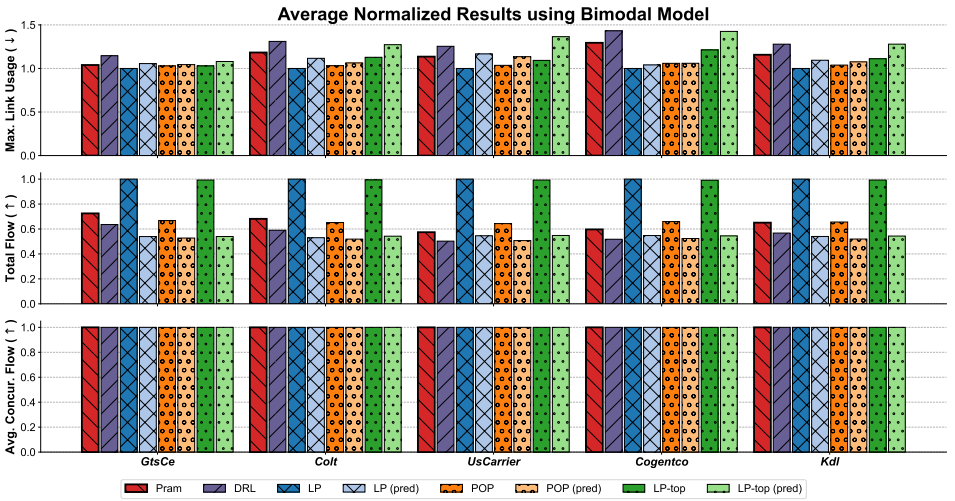
In this subsection, we provide a supplementary study to our evaluation in § 5.2, where we synthesize demand matrices for large-scale topologies using a broader set of distributions, including the Gravity,



(a) Gravity



(b) Poisson



(c) Bimodal

Figure 19: Performance comparison on large-scale topologies with different data distribution.

Poisson, and Bimodal models (see Appendix B.2.2 for details). Our results are in Figure 19. Note that the results of the gravity model in Figure 19(a) have already been reported in the main text, we refrain from elaborating on it here as a benchmark.

First, consider the Poisson model in Figure 19(b), where we observe pronounced performance fluctuations across methods as the topology changes. Particularly on the GtsCe topology, POP exhibits poor maximum flow preservation. Somewhat counterintuitively, almost all LP-based methods perform better in link utilization prediction than the exact method. We attribute this to the randomness of the Poisson distribution, which selects aggregation points independently at each step, leading to increasingly chaotic commodity flows. Among the learning-based methods, DRL consistently underperforms, while PRAM demonstrates robustness with only negligible fluctuations.

Next, as shown in Figure 19(c), our first observation under the Bimodal model is that all methods achieve near-optimal performance in terms of concurrent flow. This is expected, since the synthetic traffic is highly sparse: fewer than 10% of the flows are relatively large, while the rest are close to zero, making average parallel flow optimization straightforward. However, prediction errors become more pronounced due to the increased magnitude of the dominant flows. While PRAM is largely resilient to such errors, it performs slightly sub-optimally on the maximum flow objective, ranking just behind LP and LP-top.

To sum up, across diverse demand distributions, PRAM consistently demonstrates robustness and adaptability. Unlike DRL, which struggles with instability, or LP-based methods, whose performance is sensitive to distributional assumptions, PRAM maintains stable results across objectives and topologies. This suggests that PRAM not only generalizes beyond the traffic models used for training but also avoids overfitting to specific flow structures, thereby providing a more reliable solution in practical large-scale settings.

F ETHICS STATEMENT

We did not use any non-public data, unauthorized software, or API in our paper. In particular, the measured demand matrices, used in our evaluation, are aggregate counters between pairs of at the granularity of minutes (or coarser). They do not contain user IP addresses or packet contents. Thus, there are no privacy or other related ethical concerns.

G REPRODUCIBILITY STATEMENT

The paper fully discloses all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and conclusions of the paper. Specifically, we detailed all the experimental setups and implementation details in § 5 and Appendix B & C. Moreover, our anonymous codebase is available at https://anonymous.4open.science/r/PRAM, and the used datasets are provided in the Supplementary Material.

H ABOUT THE USAGE OF LLMS

We acknowledge the use of LLMs as a general-purpose assist tool during the preparation of this paper. The LLM was used to help with text refinement (e.g., improving clarity, grammar, and style in § 3 and § 4) and with programming assistance (e.g., generating plotting scripts and formatting code for § 5). All ideas, research methodology, analyses, and conclusions are entirely those of the authors.