WORLDGRAPH: GRAPH-NATIVE WORLD MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

World models infer latent states of an environment to capture its underlying dynamics and predict future evolution. Many real-world environments, however, are inherently relational and observed as evolving graphs, where entities, relations, and their properties change over time. Prior graph-related world models use graph structures to organize internal states or support task-specific reasoning, rather than treating an evolving graph itself as the modeled world. We instead study *graph world modeling* (GWM), where graph evolution itself constitutes the world dynamics. We formulate graph world modeling over observed graph evolution, latent graph states, and heterogeneous graph-transition predictions. Based on this formulation, we construct *GWM-Zero*, a benchmark covering node-, edge-, and graph-level transitions over eight temporal graph datasets. We propose *WorldGraph*, which combines a state-aware graph transformer for multi-granularity structural and transition-conditioned evolution modeling with transition-aware GRPO using dynamic grouping and structure-aware verifiable rewards. Extensive experiments on GWM-Zero show that WorldGraph consistently outperforms representative graph representation, temporal graph learning, graph pretraining, and graph world-model baselines across all three transition granularities.

1 INTRODUCTION

World models (Ha & Schmidhuber, 2018; Ding et al., 2026) learn compact internal states that capture how an environment evolves to support prediction of its future. Many real-world environments, however, are inherently *relational*: entities interact through structured relations, and both the entities and relations may change over time. Graphs provide a natural representation of such relational environments. More importantly, when the environment itself is observed as an evolving graph (Kurenkov et al., 2023; Savinov et al., 2018), its evolution is the world dynamics to be modeled. This motivates *graph world models* (GWMs) that model graph states and their evolution.

At time step t, the world provides an observed graph g_t, while its transition to g_{t+1} may involve updates to nodes, edges, properties, or their combinations. We denote the graph change associated with this transition by Δg_t, and view an evolving graph at a high level as $\{(g_t, \Delta g_t)\}_t$. Beyond the observed graphs, a graph world model should infer a latent world state that captures the structural and dynamic information underlying such evolution. This raises a fundamental question:

How should a world model represent the state underlying graph evolution and predict its heterogeneous transitions?

Earlier studies have introduced graphs into world modeling mainly as auxiliary structures. L^3P (Zhang et al., 2021) abstracts Markov decision process states and their reachability into a graph for long-horizon planning. C-SWM (Kipf et al., 2020) represents objects extracted from visual observations as a latent graph and uses a GNN to model their interactions. Feng et al. (Feng et al., 2025) use graphs to organize multi-modal structure and reasoning. More recently, Graph-JEPA (Skenderi et al., 2023) improves the world model's structured representation via masked subgraph latent prediction, and Liu et al. (Liu et al., 2026) conceptualized these graph-enhanced world models based on relational inductive biases. *Our focus is different: we consider graph-native worlds, where the observed environment is itself an evolving graph and graph evolution is the prediction target.* This calls for a graph-native formulation that explicitly models the latent world state underlying the graph evolution and predicts heterogeneous graph transitions across tasks of node, edge, and graph

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

levels. Unlike dynamic graph learning, which typically learns temporal representations for predefined downstream tasks, a GWM maintains a latent world state that summarizes graph evolution and uses it to model heterogeneous future transitions. Node-, edge-, and graph-level predictions can then be viewed as different observable projections of the same underlying graph-world dynamics. This raises two fundamental challenges for a general GWM: representing the latent state underlying graph evolution and modeling the heterogeneous transitions that drive it.

Challenge 1: State Modeling. A graph world state should capture not only the current graph structure but also the evolution leading to it. Different transitions may depend on different structural scales and historical contexts. For example, relation formation may require broader structural context, while local property changes may depend mainly on nearby neighborhoods and recent transitions. Existing message-passing-based GNNs (Kipf & Welling, 2017; Veličković et al., 2018; Hamilton et al., 2017) mainly capture local structure, while graph transformers and GFMs (Wu et al., 2026; 2022; Rampásek et al., 2022; Wang et al., 2025b) provide broader structural context but focus on static graph representations. Dynamic graph learning methods (Rossi et al., 2020; Peng et al., 2025) capture temporal information, but do not explicitly model a world state from graph structure and preceding transitions. The first challenge is therefore to learn a world state that jointly captures multi-granularity graph structure and historical dynamics.

Challenge 2: Transition Modeling. Graph transitions are inherently heterogeneous: they may modify node properties, relations, or local structures, while different transition patterns can occur at highly imbalanced frequencies (Liu et al., 2023). Moreover, their significance is not determined by frequency alone: a rare change involving structurally important entities may have a substantial effect on subsequent graph evolution. A general GWM should thus avoid being dominated by frequent transitions while remaining sensitive to structurally consequential changes. The second challenge is to learn heterogeneous graph transitions while accounting for both transition frequency and structural relevance.

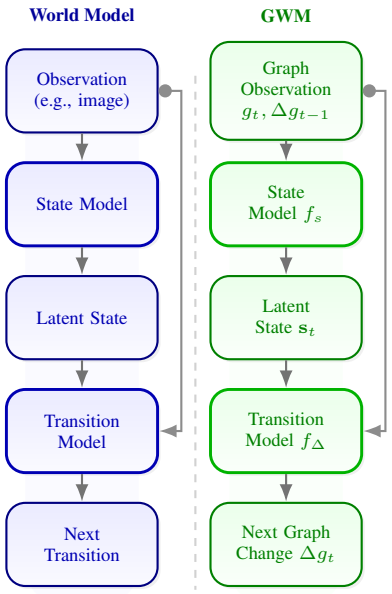


Figure 1: **World Model (Ha & Schmidhuber, 2018) vs. Our Proposed GWM**

To address these challenges, we propose *WorldGraph*, which jointly models latent world states and heterogeneous transitions. For **Challenge 1**, we develop a *state-aware graph transformer* that constructs the world state by integrating multi-granularity structural encoding and history-aware state encoding. It combines hop- and path-level graph contexts to capture structural dependencies at different scales, while incorporating preceding graph transitions to model how the current graph state has evolved over time. For **Challenge 2**, we develop a *transition-aware GRPO* with *dynamic grouping* and *structure-aware verifiable rewards*. Dynamic grouping allocates more training signal to rare transitions, while structure-aware rewards emphasize changes involving structurally important and historically variable entities. Together, these techniques enable WorldGraph to model graph dynamics across node-, edge-, and graph-level transitions within a unified framework.

Our contributions are threefold. **1)** We formulate graph-native world modeling through observed graph states, transitions, and latent world states, and construct *GWM-Zero*, a benchmark covering node-, edge-, and graph-level transition prediction tasks over **8** graph datasets. **2)** We propose *WorldGraph*, a unified framework that instantiates this formulation with a state-aware graph transformer and transition-aware RL. **3)** Extensive experiments demonstrate that WorldGraph achieves an average model quality improvement of **10.77%** over the strongest baselines, with the largest improvement of **50.66%** in F1 score on the TGBN-Trade (Huang et al., 2023) dataset.

2 Problem Formulation

We formalize graph world modeling through observed graph states, transitions, and world states.

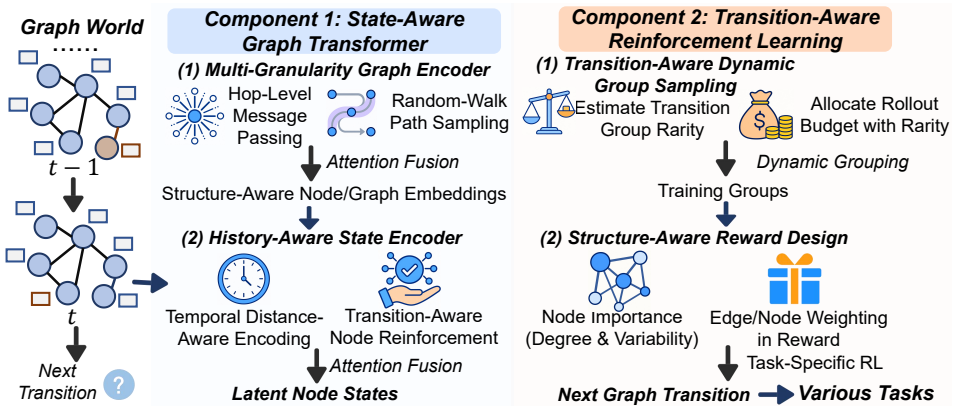


Figure 2: **Overview of WorldGraph**. The *state-aware graph transformer* constructs latent world states by integrating multi-granularity graph structure with transition-conditioned history. The *transition-aware RL algorithm* learns heterogeneous graph transitions through dynamic grouping and structure-aware verifiable rewards across node-, edge-, and graph-level tasks.

Observed Graph Evolution. At time step t, the observed *graph state* is represented as $g_t = (V_t, E_t, X_t) \in \mathcal{G}$, where V_t, E_t, and X_t denote the entities, relations, and associated properties, respectively. The evolution from g_t to g_{t+1} forms a *graph transition*. We use Δg_t to denote the *graph change* associated with this transition, which may involve node or edge addition/deletion, property changes, or their combinations. Accordingly, an evolving graph world can be viewed as a sequence of graph states and their changes, $\{(g_t, \Delta g_t)\}_{t=1}^{T}$.

Latent World States. While g_t describes the graph observed at time t, the underlying dynamics of graph evolution are not directly observed. We therefore introduce a world state $\mathbf{s}_t \in \mathcal{S}$ that summarizes both the current graph structure and the preceding evolution:

$$\mathbf{s}_t = f_s(g_t, \Delta g_{t-1}, \mathbf{s}_{t-1}), \tag{1}$$

where f_s denotes the *state model*.

Definition 1 (Graph World Model). *A graph world model (GWM) models graph-world dynamics through a state model f_s and a transition model f_Δ. Given the current graph observation g_t, the preceding graph change Δg_{t-1}, and the previous world state $\mathbf{s}_{t-1}$, the state model infers $\mathbf{s}_t = f_s(g_t, \Delta g_{t-1}, \mathbf{s}_{t-1})$, while the transition model predicts the next graph change as $\Delta g_t = f_\Delta(g_t, \Delta g_{t-1}, \mathbf{s}_t)$.*

The two components capture complementary aspects of graph-world dynamics. The state model captures the latent structural and dynamic information of the evolving graph world, while the transition model predicts its subsequent evolution.

A complete graph change Δg_t may contain multiple structural events and may be observed at different granularities. We therefore instantiate graph-transition prediction at three levels: node-level changes in entities and properties, edge-level changes in relations, and graph-level changes in local structures. Importantly, these tasks are different observable projections of the same graph-world transition Δg_t rather than separate forms of the world dynamics. Accordingly, graph world modeling consists of two coupled problems: *state modeling* for learning the latent world state $\mathbf{s}_t$, and *transition modeling*, which predicts observable graph transitions from this state.

3 WorldGraph

WorldGraph instantiates the two components of a GWM introduced in Section 2. For state modeling, it constructs s_t by combining multi-granularity structure in the current graph with transition-conditioned history. For transition modeling, it learns heterogeneous graph transitions with frequency-aware sampling and structure-aware rewards. Figure 2 gives an overview of the framework.

3.1 State-aware Graph Transformer

The latent world state s_t should summarize both what the current graph looks like and how it arrived there. WorldGraph constructs this state in two stages: a multi-granularity graph encoder captures the current structure, and a history-aware state encoder integrates the preceding evolution.

3.1.1 STAGE 1: MULTI-GRANULARITY GRAPH ENCODER

Different graph transitions depend on structural information at different scales[1]. We combine hop-based message passing and random-walk path sampling to obtain complementary structural views.

Hop-Level Message Passing. Given a maximum receptive field constraint (i.e., the maximum number of hops $L_{\max}$) and the graph state $g_t = (V_t, E_t, X_t)$, for every node $v \in V_t$, we obtain the hop-level node embeddings $\{\mathbf{x}_v^{(\ell)}\}_{1 \le \ell \le L_{\max}}$, where $\mathbf{x}_v^{(\ell)}$ aggregates the node features from all neighbors within the ℓ-hop range of node v by iteratively applying message passing (Hamilton et al., 2017).

Random-Walk Path Sampling. To further enhance the richness of structural representations, we also perform random-walk sampling to collect M paths starting from every node $v \in V_t$. Then, for each node $v \in V_t$, we obtain the path-level embeddings $\{\mathbf{x}_v^{\mathrm{path}_i}\}_{i=1}^{M}$, where $\mathbf{x}_v^{\mathrm{path}_i}$ aggregates all node features along path_i by applying an attention mechanism (Vaswani et al., 2017).

Multi-Granularity Attention Fusion. For each node $v \in V_t$ at time t, after obtaining $\{\mathbf{x}_v^{(\ell)}\}_{\ell=1}^{L_{\max}}$ and $\{\mathbf{x}_v^{\mathrm{path}_i}\}_{i=1}^{M}$, we fuse these embeddings via an attention-based aggregation[2] to obtain a structure-aware node embedding $\mathbf{z}_{v,t}$. For g_t, we further obtain the structure-aware graph embedding $\mathbf{z}_t$ by applying mean pooling over the node embeddings $\{\mathbf{z}_{v,t}\}_{v \in V_t}$.

3.1.2 STAGE 2: HISTORY-AWARE STATE ENCODER

The current graph structure alone does not reveal how the graph arrived at its present state. We therefore integrate preceding graph changes into the latent world state, while accounting for both their temporal distance and their relevance to recent evolution.

Temporal Distance-Aware Encoding. Because the states that are closer in time have a greater impact on the current transition (Wang et al., 2025a), we consider incorporating the temporal distance (i.e., the difference between the current time and historical time) into the latent states at first.

Given the current time t and a historical time $\tau < t$, we obtain the transition representation of Δg_τ as follows: $\mathbf{x}_{\Delta g_\tau} = \mathrm{Type}(\Delta g_\tau)$, where $\mathrm{Type}(\cdot)$ encodes the graph transition type[3]. Then, for every node $v \in V_t$, we calculate the node memory embeddings $\{\mathbf{m}_{v,\tau}\}$ for $\tau < t$[4], where

$$\mathbf{m}_{v,\tau} = \mathbf{W}_x \mathbf{z}_{v,\tau} + \mathbf{W}_z \mathbf{z}_\tau + \mathrm{Dist}(t, \tau) + \mathbf{W}_a \mathbf{x}_{\Delta g_\tau}, \tag{2}$$

where $\mathbf{W}_x$, $\mathbf{W}_z$, and $\mathbf{W}_a$ are learnable weight matrices used for projection. $\mathrm{Dist}(t, \tau)$ encodes the relative temporal distance $t - \tau$[4]. Then, the node memory embedding is used to obtain the latent state. $\mathbf{W}_a \mathbf{x}_{\Delta g_\tau}$ allows the node memory embedding to incorporate the graph transition type.

History-Aware Attention Fusion. For each node v at time t, we construct a query embedding $\mathbf{q}_{v,t}$ by concatenating $\mathbf{z}_{v,t}$, $\mathbf{z}_t$, $\mathbf{x}_{\Delta g_{t-1}}$, and its previous latent state $\mathbf{s}_{v,t-1}$. Then, we aggregate its historical memory $\{\mathbf{m}_{v,\tau}\}$ with multi-head attention. For one attention head, the attention score from history τ to the current time t is: $\beta_{v,t,\tau} = \frac{(\mathbf{q}_{v,t}\mathbf{W}_Q)(\mathbf{m}_{v,\tau}\mathbf{W}_K)^\top}{\sqrt{d}} + \mathbf{p}_{t-\tau} + \Pi(\Delta g_\tau, \Delta g_{t-1})$, where $\mathbf{W}_Q$ and $\mathbf{W}_K$ are learnable linear projection matrices, and d is the dimension of the projected node embedding. $\Pi(\Delta g_\tau, \Delta g_{t-1}) = (\mathbf{x}_{\Delta g_\tau})^\top \mathbf{x}_{\Delta g_{t-1}}$ is the similarity between the two transitions. This similarity is used as the reinforcement to account for how previous transitions (Δg_τ) influence recent dynamics (Δg_{t-1}, which is the most recently observed transition).

Then, we obtain $\tilde{\beta}_{v,t,\tau}$ using exponential normalization across τ. Based on this, the history embedding for node v (one attention head) is $\mathbf{h}_{v,t} = \sum_\tau \tilde{\beta}_{v,t,\tau} (\mathbf{m}_{v,\tau}\mathbf{W}_V)$, where $\mathbf{W}_V$ is the learnable linear projection matrix. The final history embedding is obtained by mixing multiple attention heads with an MLP, yielding $\tilde{\mathbf{h}}_{v,t}$.

Finally, we sum $\mathbf{z}_{v,t}$, the history embedding $\tilde{\mathbf{h}}_{v,t}$, and the previous latent state $\mathbf{s}_{v,t-1}$ to obtain the

[1] For example, in the UN Vote dataset (Poursafaei et al., 2022), two countries without a direct co-voting edge may be linked by a multi-hop chain of shared co-voting partners, so predicting a new edge requires long-range context. In the TGBN-Genre dataset (Huang et al., 2023), predicting whether a user's music preferences change mainly depends on that user's recent preference history, so this local information is sufficient.

[2] Here, we fuse the hop-level embeddings and the path-level embeddings, while also incorporating global context from a linear-attention-aware subgraph (Wu et al., 2026).

[3] We use a codebook that maps each transition type to its corresponding learnable embedding.

[4] In practice, we compute the node memory embeddings for the history period up to the past $t_{\max}$ (i.e., $t - t_{\max} \le \tau < t$), which is discussed in Appendix H.3.

latent node state at the current time $\mathbf{s}_{v,t}$ (LayerNorm (Ba et al., 2016) is used to stabilize training):

$$\mathbf{s}_{v,t} = \mathrm{LayerNorm}(\tilde{\mathbf{h}}_{v,t} + \mathbf{z}_{v,t} + \mathbf{s}_{v,t-1}). \tag{3}$$

3.1.3 Theoretical Analysis of State-Aware Graph Transformer

Here, we provide a theoretical analysis of why our state-aware graph transformer has advantages over the current state-of-the-art graph transformer, i.e., SGFormer (Wu et al., 2026), in terms of state modeling. For this analysis, we start from the message-passing mechanisms of SGFormer and WorldGraph, and then derive an upper bound on the difference between the final structure-aware node embedding they produce and the ideal structure-aware node embedding (i.e., the embedding obtained when the sampled subgraph provides full coverage).

Since SGFormer is trained using a subgraph with a fixed hop size, let the obtained subgraph be $g^{(\mathrm{SG})}$. In contrast, our graph transformer uses multiple-hop subgraphs and random-walk sampled subgraphs. These subgraphs are denoted as $\{g_i^{(\mathrm{OURS})}\}_{i=1}^{L_{\max}+M}$. Ideally, the subgraphs are generated from neighborhoods spanning arbitrarily large hop distances, and the random-walk sampling is performed with a sufficiently large number of steps. Let ζ denote the (sufficient) number of subgraphs. We obtain $\{g_i^{(\mathrm{IDEAL})}\}_{i=1}^{\zeta}$. By substituting these into $\mathbf{s}_{v,t}$, we can obtain the latent states produced by different mechanisms, which we denote as $\mathbf{s}_{v,t}^{(\mathrm{SG})}$, $\mathbf{s}_{v,t}^{(\mathrm{OURS})}$, and $\mathbf{s}_{v,t}^{(\mathrm{IDEAL})}$.

Theorem 1. *The difference between* $\mathbf{s}_{v,t}^{(\mathrm{OURS})}$ *and* $\mathbf{s}_{v,t}^{(\mathrm{IDEAL})}$ *is* $\Delta_{\mathrm{OURS}} = \left\|\mathbf{s}_{v,t}^{(\mathrm{OURS})} - \mathbf{s}_{v,t}^{(\mathrm{IDEAL})}\right\|_2$, *and the difference between* $\mathbf{s}_{v,t}^{(\mathrm{SG})}$ *and* $\mathbf{s}_{v,t}^{(\mathrm{IDEAL})}$ *is* $\Delta_{\mathrm{SG}} = \left\|\mathbf{s}_{v,t}^{(\mathrm{SG})} - \mathbf{s}_{v,t}^{(\mathrm{IDEAL})}\right\|_2$. *The upper bound of* Δ_{OURS} *is smaller than that of* Δ_{SG}.

Proof. The detailed proof is provided in Appendix C.1. □

3.2 Transition-Aware Reinforcement Learning

Given the latent world state s_t, the transition model predicts heterogeneous graph changes with highly imbalanced frequencies and varying structural relevance. To account for these differences during optimization, we formulate transition learning with verifiable rewards and adapt GRPO to graph-world transitions. Transition-aware dynamic grouping emphasizes rare transitions, while structure-aware rewards account for structural relevance.

3.2.1 Transition-Aware Dynamic Group Sampling

Here, we introduce transition-aware dynamic grouping: the group assignment at time step t is biased towards rare historical transitions (i.e., the transitions that appear less frequently in the history). This design increases the coverage of low-frequency transitions, providing them with more training signal. As a result, the algorithm can learn robust policies for rare transitions.

Dynamic Transition Frequency Estimation. To achieve such a dynamic group sampling algorithm, we need to estimate the transition frequency at first. At time t, let the transition-conditioned groups be $\mathcal{J}_t$. Each group $J_i \in \mathcal{J}_t$ corresponds to a specific graph transition type and stores the transitions from time steps before t that belong to this transition type. For a transition group $J_i \in \mathcal{J}_t$, we define the size of group J_i as $|J_i|$. Based on $|J_i|$, we can compute the frequency of transitions of different types before time step t as $\widehat{p}(J_i)$.

We then assign a rarity weight to each transition group, so that groups appearing less frequently in the prefix receive larger weights: $\omega(J_i) = (\widehat{p}(J_i))^{-\gamma}, \gamma > 0$[5], and the probability of selecting group J_i is $\mathrm{Prob}(J_i) = \frac{\exp(\omega(J_i))}{\sum_{J_j \in \mathcal{J}_t} \exp(\omega(J_j))}$.

Rarity-based Rollout Budget Allocation. Finally, to make earlier rare transitions receive a larger effective group size, we allocate the total rollout budget $\mathcal{B}$ across groups proportionally. We reserve the same minimum number of rollouts[6] for every group and use largest-remainder rounding for the remaining budget, such that $\sum_{J_i} B_t(J_i) = \mathcal{B}$, where $B_t(J_i)$ is the allocated number of rollouts

[5] In practice, we set γ as 1, θ as 1, and η as 1, as discussed in Appendix H.3.

[6] In practice, we set the minimum number of rollouts as 2. We do this to ensure every group receives at least a minimal number of rollouts, preventing overly noisy return estimates from groups with too few samples.

for transition group J_i at time t. Thus, groups corresponding to transitions that are rarer in the history are assigned more rollout samples at time t, increasing the relative training signal for these frequency-imbalanced transition patterns.

3.2.2 STRUCTURE-AWARE REWARD DESIGN

Graph transitions differ in their structural relevance. We therefore weight the verifiable reward using two complementary signals: structural importance and historical variability, emphasizing transitions involving more relevant nodes.

Node Importance Calculation. For a node $v \in V_t$, we define a structural importance term based on its node degree ($\deg(\cdot)$): $\mathcal{I}_{\mathbf{s}}(v) = (\deg(v))^{\theta}$, where $\theta > 0$[5] controls how strongly we emphasize highly-connected nodes. Let $f_{<t}(v)$ be the empirical frequency that node v is involved in a change (i.e., the fraction of time steps at which node v is modified), and define the variability term as: $\mathcal{I}_{\mathbf{v}}(v) = (f_{<t}(v))^{\eta}$, with $\eta > 0$[5]. We combine the two terms into a single importance: $\mathcal{I}(v) = \frac{\mathcal{I}_{\mathbf{s}}(v)\cdot\mathcal{I}_{\mathbf{v}}(v)}{\sum_{v'\in V_t}(\mathcal{I}_{\mathbf{s}}(v')\cdot\mathcal{I}_{\mathbf{v}}(v'))}$. Based on the node importance, we design a structure-aware reward.

Task-Specific Structure-Aware Reward Design. Graph-transition tasks, including node-, edge-, and graph-level tasks, produce outputs at different granularities. We therefore define a verifiable reward consistent with the output and evaluation metric. Then, we define the RL reward as follows. For node-level tasks[7], we denote the predicted and ground-truth change node sets by $\Delta\hat{Q}$ and ΔQ, respectively. A prediction is counted as a true positive only when it exactly matches a ground-truth change. Therefore, we define the precision $\mathcal{P}$, recall $\mathcal{R}$, and the structure-aware reward $R_{\text{structure}}$:

$$\mathcal{P} = \frac{\sum_{v\in\Delta\hat{Q}\cap\Delta Q}\mathcal{I}(v)}{\sum_{v\in\Delta\hat{Q}}\mathcal{I}(v)}, \quad \mathcal{R} = \frac{\sum_{v\in\Delta\hat{Q}\cap\Delta Q}\mathcal{I}(v)}{\sum_{v\in\Delta Q}\mathcal{I}(v)}, \quad R_{\text{structure}} = \frac{2\,\mathcal{P}\,\mathcal{R}}{\mathcal{P}+\mathcal{R}}. \tag{4}$$

Based on this, we design rewards for node-level, edge-level, and graph-level tasks as follows. Although the three graph-world tasks operate at different granularities, their rewards share the same principle: a prediction should identify the correct structural changes and estimate the corresponding property changes. We therefore use the unified reward $R = \lambda R_{\text{structure}} + (1-\lambda)R_{\text{property}}$, where R_{property} is calculated based on the ground-truth property and the predicted property. λ balances the two terms. The concrete definitions for node-, edge-, and graph-level tasks are provided in Appendix B.3. Based on these rewards, we train the policy network $\Delta g_t = f_{\Delta}(g_t, \Delta g_{t-1}, \mathbf{s}_t)$[8] to maximize the expected cumulative return and obtain the transition Δg_t at time step t.

3.2.3 THEORETICAL ANALYSIS OF TRANSITION-AWARE RL ALGORITHM

In GRPO (Shao et al., 2024), the group size is the same for different transitions. The distribution of group sizes across different transitions can be viewed as uniform. In our transition-aware RL algorithm, our transition group-size distribution is determined based on the historical interaction frequency of each transition. Following (Kim & Lee, 2026), we use the effective variance ratio (EVR). A smaller EVR means lower gradient noise and more stable RL training.

Theorem 2. *In GRPO, the group size is $\rho^{\text{GRPO}} = C_a$, where C_a is a constant for different graph transition types. In our RL algorithm, the group size corresponding to each transition type (the corresponding group is J_i) is $\rho^{\text{OURS}} = C_a + \text{Prob}(J_i)\cdot\mathcal{B}$. Based on (Kim & Lee, 2026), for a group size ρ_i, an upper bound on EVR is $\text{EVR}(\rho_i) = 1 + \frac{\delta-1}{2\rho_i} + M_{\text{EVR}}$, where $M_{\text{EVR}} \geq \mathcal{O}\left(\frac{1}{\rho_i^2}\right)$ is a constant and δ is the kurtosis of the reward distribution (Kim & Lee, 2026). Then, for a given transition type, the group produced by our RL algorithm has a smaller upper bound on the EVR than the group produced by GRPO.*

Proof. The detailed proof is provided in Appendix C.2. □

4 EXPERIMENTS

In this section, we conduct extensive experiments to answer the following research questions:
(RQ1): Compared with existing state-of-the-art methods for solving graph world modeling problems, does WorldGraph have advantages in terms of model quality?
(RQ2): How effective are the components of WorldGraph?

[7] For edge-level (or graph-level) tasks, we use the change edge (or graph) sets, respectively.
[8] $\mathbf{s}_t$ includes all latent node states $\{\mathbf{s}_{v,t}\}$ in g_t. f_{Δ} is an MLP.

Table 1: **Overall Performance on the 3 Graph-Transition Tasks.** Add., Rem., and Sem. denote addition, deletion, and property-change F1, respectively. Higher is better for F1, Macro-F1, and NDCG@10, while lower is better for MAE and RMSE. The best results are shown in **bold** and the second best results are underlined. We use an existing Controller design (Dinella et al., 2020) to enable the baselines in the first three categories to predict node-, edge-, and graph-level changes.

(a) Node-Level Tasks

Method	Type	TGBN-Trade				TGBN-Genre				TGBN-Reddit			
		Add. F1	Rem. F1	Sem. F1	NDCG@10	Add. F1	Rem. F1	Sem. F1	NDCG@10	Add. F1	Rem. F1	Sem. F1	NDCG@10
GCN	MPNN	0.4718	0.1540	0.5741	0.6357	0.7112	0.8488	0.6375	0.2469	0.9706	0.8976	0.6284	0.2161
GAT	MPNN	0.5248	0.1385	0.6275	0.6386	0.7253	0.8598	0.6523	0.2699	0.9720	0.8972	0.6440	0.2237
GraphSAGE	MPNN	0.5864	0.0000	0.6595	0.6246	0.7256	0.8538	0.6496	0.2780	0.9705	0.8902	0.6435	0.2519
SGFormer	Graph Transformer	0.6874	0.0308	0.6732	0.6324	0.7252	0.8564	0.6497	0.2822	0.9731	0.9024	0.6445	0.2413
NodeFormer	Graph Transformer	0.7069	0.0878	0.6669	0.6336	0.7080	0.8394	0.6470	0.2344	0.8011	0.7329	0.5409	0.1275
GraphGPS	Graph Transformer	0.8033	0.0000	0.6542	0.6155	0.7302	0.8596	0.6533	0.2824	0.9684	0.8931	0.6449	0.2376
Graph-JEPA	Pretrained	0.5585	0.0419	0.5777	0.6357	0.6563	0.7634	0.6143	0.2487	0.9573	0.8861	0.5999	0.1520
MDGFM	Pretrained	0.4497	0.1517	0.5616	0.6406	0.7194	0.8426	0.6177	0.2107	0.9558	0.8607	0.6151	0.1676
TGN	Temporal Model	0.8492	0.0559	0.6666	0.6181	0.7378	0.8698	0.6505	0.2683	0.9726	0.8982	0.6457	0.2259
TIDFormer	Temporal Model	0.8442	0.1129	0.6740	0.6299	0.7409	0.8754	0.6529	0.2556	0.9705	0.9066	0.6429	0.2150
WorldGraph	**GWM**	0.9245	0.6430	0.7117	**0.6484**	0.7460	0.8801	0.6557	0.4418	0.9755	0.9317	0.6458	0.3838
WorldGraph-Pretrain	**GWM**	**0.9696**	**0.6606**	**0.7354**	0.6469	**0.7474**	**0.8849**	**0.6685**	**0.4448**	**0.9779**	**0.9561**	**0.6492**	**0.3901**

(b) Edge-Level Tasks

Method	Type	TGBN-Trade			UN Vote			Contact			SocialEvo		
		Add. F1	Rem. F1	Macro-F1	Add. F1	Rem. F1	Macro-F1	Add. F1	Rem. F1	Macro-F1	Add. F1	Rem. F1	Macro-F1
GCN	MPNN	0.2633	0.2973	0.2803	0.1337	0.1025	0.1181	0.0146	0.7946	0.4046	0.2387	0.5554	0.3970
GAT	MPNN	0.2967	0.3841	0.3404	0.1393	0.0926	0.1159	0.0146	0.7518	0.3832	0.2337	0.5897	0.4117
GraphSAGE	MPNN	0.3114	0.3865	0.3489	0.1649	0.1031	0.1340	0.0170	0.8174	0.4172	0.2629	0.5556	0.4093
SGFormer	Graph Transformer	0.3174	0.3915	0.3544	0.1403	0.1058	0.1230	0.0141	0.8144	0.4142	0.2175	0.5233	0.3704
NodeFormer	Graph Transformer	0.3031	0.3797	0.3414	0.1610	0.0840	0.1225	0.0124	0.5443	0.2784	0.1532	0.3496	0.2514
GraphGPS	Graph Transformer	0.3153	0.3926	0.3540	0.1270	0.0952	0.1111	0.0272	0.8847	0.4559	0.2611	0.5198	0.3905
Graph-JEPA	Pretrained	0.2571	0.2933	0.2752	0.1308	0.1083	0.1196	0.0165	0.8659	0.4412	0.2180	0.5312	0.3746
MDGFM	Pretrained	0.2187	0.2975	0.2581	0.1188	0.1041	0.1114	0.0112	0.7174	0.3643	0.2285	0.5184	0.3734
TGN	Temporal Model	0.3202	0.4010	0.3606	0.1886	0.1112	0.1499	0.0221	0.8776	0.4498	0.2518	0.5691	0.4104
TIDFormer	Temporal Model	0.3217	0.4005	0.3611	0.2132	0.3089	0.2611	0.0329	0.8583	0.4456	0.2651	0.5219	0.3935
WorldGraph	**GWM**	**0.3952**	0.5325	0.4640	**0.4195**	0.5395	0.4795	0.1628	**0.9009**	0.5319	0.3841	0.6567	0.5204
WorldGraph-Pretrain	**GWM**	0.3948	**0.5344**	**0.4646**	0.4165	**0.5996**	**0.5080**	**0.1819**	0.8991	**0.5405**	**0.3890**	**0.6609**	**0.5249**

(c) Graph-Level Tasks

Method	Type	Flights			Contact			Enron		
		MAE	RMSE	Macro-F1	MAE	RMSE	Macro-F1	MAE	RMSE	Macro-F1
GCN	MPNN	0.8047	1.2736	0.4902	0.6842	0.9884	0.4357	1.1674	1.7609	0.3955
GAT	MPNN	0.8354	1.3303	0.4173	0.7470	1.0338	0.4238	1.1887	1.8048	0.3724
GraphSAGE	MPNN	0.8163	1.2984	0.4763	0.6811	0.9832	0.4402	1.1702	1.7731	0.4052
SGFormer	Graph Transformer	0.8043	1.2768	0.4837	0.6823	0.9842	0.4334	1.1533	1.7428	0.3996
NodeFormer	Graph Transformer	0.7941	1.2835	0.3974	0.7554	1.0403	0.3970	1.1573	1.7733	0.4139
GraphGPS	Graph Transformer	0.7723	1.2482	0.4826	0.6378	0.9452	0.4482	1.1713	1.7567	0.4090
Graph-JEPA	Pretrained	0.8407	1.3243	0.4413	0.7109	1.0177	0.4160	1.2211	1.7961	0.3483
MDGFM	Pretrained	0.8440	1.3357	0.3904	0.7385	1.0326	0.4001	1.2257	1.8545	0.3703
TGN	Temporal Model	0.7956	1.2615	0.4796	0.6659	0.9713	0.4151	1.1724	1.7767	0.4155
TIDFormer	Temporal Model	0.7929	1.2480	0.4759	0.6935	0.9955	0.4250	1.1980	1.8047	0.4102
WorldGraph	**GWM**	0.7069	1.1254	0.5840	0.5648	0.8972	0.5438	1.1103	1.5447	**0.4255**
WorldGraph-Pretrain	**GWM**	**0.6841**	**1.0985**	**0.5891**	**0.5540**	**0.8865**	**0.5547**	**1.0983**	**1.5383**	0.4244

(RQ3): How sensitive is WorldGraph to its key model configurations?
(RQ4): Is it effective to integrate WorldGraph into the existing graph world model problem settings?

4.1 Experimental Setup

Benchmark. We construct the graph world benchmark, GWM-Zero, at three levels, including node, edge, and graph, to comprehensively evaluate *WorldGraph*. The benchmark is built from TGBN-Trade (Huang et al., 2023), TGBN-Genre (Huang et al., 2023), TGBN-Reddit (Huang et al., 2023), UN Vote (Poursafaei et al., 2022), Flights (Poursafaei et al., 2022), Contact (Poursafaei et al., 2022), SocialEvo (Poursafaei et al., 2022), and Enron (Poursafaei et al., 2022).

Baselines. We select **13** baselines from four perspectives. (1) From the graph-representation perspective, we include message passing-based neural networks (MPNN)—GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and GraphSAGE (Hamilton et al., 2017)—as well as graph Transformers—SGFormer (Wu et al., 2026), NodeFormer (Wu et al., 2022), and GraphGPS (Rampásek et al., 2022). (2) From the temporal-modeling perspective, we compare with TGN (Rossi et al., 2020) and TIDFormer (Peng et al., 2025), which explicitly model time-stamped interactions and temporal dependencies. (3) From the graph-pretraining perspective, we include

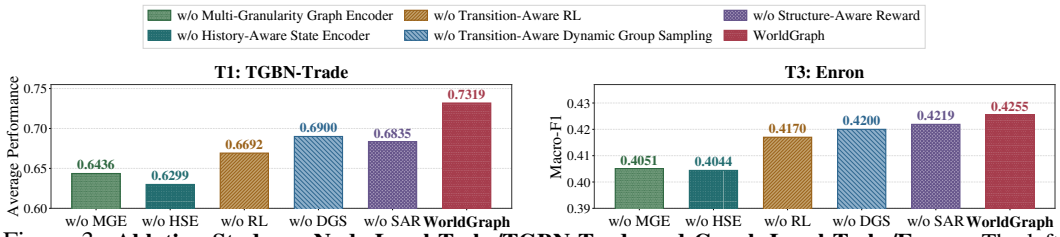


Figure 3: **Ablation Study on Node-Level Tasks/TGBN-Trade and Graph-Level Tasks/Enron.** The left panel reports the equal-weight average of the 4 TGBN-Trade metrics, while the right panel reports Macro-F1.

Graph-JEPA (Skenderi et al., 2023) and MDGFM (Wang et al., 2025b) to evaluate the benefit of self-supervised and multi-domain pretraining. (4) Under existing world-modeling problem settings, we include L^{3}P (Zhang et al., 2021), C-SWM (Kipf et al., 2020), and GWM-E (Feng et al., 2025).

4.2 Overall Performance (**RQ1**)

As shown in Table 1, across all 3 task levels, WorldGraph and WorldGraph-Pretrain (Pretraining details are provided in Appendix G.1.) consistently outperform the baselines. **For node-level tasks**, the gains are particularly pronounced for node deletion on TGBN-Trade, where sparse deletion events cause GraphSAGE and GraphGPS to incorrectly predict the deletion nodes, yielding $\mathrm{Rem.\ F1} = 0$, while WorldGraph-Pretrain reaches $\mathrm{Rem.\ F1} = 0.6606$, an absolute improvement of 50.66% over the second best method. For semantic-property prediction, WorldGraph-Pretrain improves over the second best methods by 16.24% on TGBN-Genre and 13.82% on TGBN-Reddit, showing that jointly modeling graph structure, historical states, and graph transitions is beneficial for heterogeneous node transition prediction tasks. **For edge-level tasks**, the advantage is more evident, with an average Macro-F1 improvement of 13.72%. Compared with the strongest baselines on the corresponding dataset, its improvements on UN Vote (↑24.69%), Contact (↑8.46%), and SocialEvo (↑11.37%) demonstrate that the learned transition policy can capture the graph changes that are difficult for representation-only and temporal baselines. **For graph-level tasks**, WorldGraph also obtains the lowest MAE and RMSE and the highest Macro-F1, with average absolute reductions of 0.0757 and 0.1376 in MAE and RMSE, respectively, and an average Macro-F1 improvement of 7.18% over the best external baseline for each dataset and metric. These indicate that its advantage extends from discrete node and edge edits to continuous local-structure evolution.

4.3 Ablation Study (**RQ2**)

We evaluate five variants that respectively remove the multi-granularity graph encoder, the history-aware state encoder, the transition-aware RL mechanism, as well as its two internal components: dynamic group sampling and structure-aware reward. Figure 3 summarizes their overall performance on Node-Level Tasks/TGBN-Trade and Graph-Level Tasks/Enron. Removing any component degrades performance on both tasks. The largest reductions are caused by removing the multi-granularity graph encoder or the history-aware state encoder, which lead to absolute drops of 8.83% and 10.20% on Node-Level Tasks/Trade and 2.04% and 2.11% on Graph-Level Tasks/Enron, respectively, confirming that both current graph structure and historical evolution are essential for representing a graph world. Removing the complete RL mechanism, or removing either of its two proposed components—dynamic group sampling or structure-aware reward—also consistently reduces performance, with absolute drops of 6.27/0.85%, 4.19/0.55%, and 4.84/0.36%, respectively. The results show that the two components make complementary contributions to the final performance.

4.4 Sensitivity Analysis (**RQ3**)

Figure 4 shows the sensitivity of WorldGraph to three key hyperparameters on Graph-Level Tasks/Flights (Macro-F1). Overall, the performance varies by only 0.31 percentage points across all settings (ranging narrowly from 58.09% to 58.40%), demonstrating that WorldGraph is robust to these hyperparameter configurations. Specifically, varying the maximum hop number $L_{\max}$ from 1 to 4 leads to small fluctuations between 58.09% and 58.40%, with

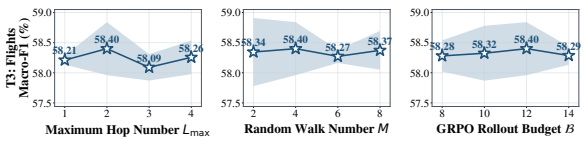


Figure 4: **Sensitivity Analysis on Graph-Level Tasks/Flights.** Performance remains remarkably stable across varying maximum hop numbers $L_{\max}$, random-walk numbers M, and GRPO rollout budgets $\mathcal{B}$.

$L_{\max} = 2$ achieving the peak performance. The random-walk number M also exhibits a very marginal impact: the Macro-F1 score remains virtually flat within the range of 58.27%–58.40% across $M \in \{2, 4, 6, 8\}$. For the GRPO rollout budget $\mathcal{B}$, performance stays stably within 58.28%–58.40% without noticeable deviation as $\mathcal{B}$ increases from 8 to 14. These results confirm that WorldGraph is not sensitive to the choice of these structural and optimization hyperparameters, further validating its stability.

4.5 Comparison on Existing GWM Problem Settings (**RQ4**)

For L^{3}P, we evaluate long-horizon planning on PointMaze (Duan et al., 2016) and FetchPickAndPlace (Plappert et al., 2018). Integrating WorldGraph as a plug-in module provides graph-structured state and history information to the landmark planner while leaving the original environment, actor, and training objective unchanged. The resulting learning curves are shown in Figure 5. Over the final ten test epochs, L^{3}P + WorldGraph leads L^{3}P by 7.11 percentage points on PointMaze (94.88% vs. 87.77%) and by 2.43 points on FetchPickAndPlace (99.18% vs. 96.74%).

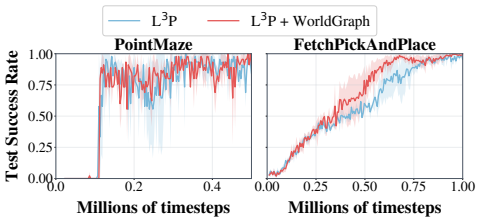


Figure 5: **Comparison with L^{3}P on PointMaze and FetchPickAndPlace.** Solid curves show the mean test success rate and shaded regions show the standard deviation.

For C-SWM, we evaluate multi-step latent-state ranking on Pong (Brockman et al., 2016) and Space Invaders (SI) (Brockman et al., 2016) environments (env.) at 1, 5, and 10 steps. The plug-in augments C-SWM's object-centric transition model with graph-aware state and transition information, while retaining its original object encoder, transition backbone, and contrastive training objective. As shown in Table 2, WorldGraph improves both H@1 and MRR at every horizon on both games. The gains become larger at longer horizons, reaching 31.33 in H@1 and 29.55 in MRR at 10 steps on SI.

Table 2: **Comparison with C-SWM on Multi-Step Latent-State Ranking Tasks.** Higher H@1 and MRR are better.

Env.	Model	1 Step		5 Steps		10 Steps	
		H@1	MRR	H@1	MRR	H@1	MRR
Pong	C-SWM	35.00	51.45	11.33	27.24	6.67	19.03
	+ WorldGraph	**40.33**	**57.68**	**26.33**	**45.85**	**16.00**	**34.62**
SI	C-SWM	63.67	75.49	41.33	56.66	29.67	46.16
	+ WorldGraph	**68.00**	**79.68**	**62.67**	**76.04**	**61.00**	**75.71**

Table 3: **Comparison with GWM-E on Node Classification (NC), Link Prediction (LP), and Graph Classification (GC) tasks (Accuracy/%).** Higher accuracy is better.

Model	**Cora**		**PubMed**		**HIV**
	NC	LP	NC	LP	GC
GWM-E	83.03	94.31	84.22	94.01	93.86
WorldGraph	**88.93**	**95.17**	**89.34**	**97.04**	**96.49**

For comparison with GWM-E, we use its traditional graph prediction setting on Cora (McCallum et al., 2000), PubMed (Namata et al., 2012), and HIV (Wu et al., 2018), covering node classification, link prediction, and graph classification. We independently train WorldGraph on these tasks with its multi-granularity graph encoder and task-specific prediction heads. WorldGraph achieves higher accuracy in all five columns, with absolute gains of 5.90 and 0.86 on Cora node classification and link prediction, 5.12 and 3.03 on the corresponding PubMed tasks, and 2.63 on HIV graph classification.

5 Conclusion

In this paper, we study *graph world modeling* by formalizing latent world states and heterogeneous graph transitions from evolving graph observations. We introduce *GWM-Zero* to cover node-, edge-, and graph-level transition prediction across eight temporal graph datasets, and propose *WorldGraph*, which uses a state-aware graph transformer for multi-granularity state modeling and a transition-aware GRPO with dynamic grouping and structure-aware verifiable rewards for learning under frequency imbalance. Extensive experiments show that WorldGraph consistently achieves stronger performance across diverse *multi-granularity graph transition prediction tasks*. WorldGraph provides a general and effective framework for graph-native world modeling and opens up promising directions for learning richer latent dynamics and more controllable graph evolution in future work.

AI USE STATEMENT

In this work, we used generative AI tools for polishing writing. We have not used generative AI tools for other tasks with required disclosure, and the rest of the required disclosure tasks are not applicable to this work. All AI-polished text was reviewed by all authors for accuracy and clarity. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects and is based on publicly available datasets.

REPRODUCIBILITY STATEMENT

We ensure reproducibility by providing references to the exact experimental setup, dataset and model details in the main paper, appendix, and supplementary materials, and by releasing source code.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. 2016. URL https://arxiv.org/abs/1607.06450.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. 2016. URL https://arxiv.org/abs/1606.01540.

Maxime Burchi and Radu Timofte. Learning transformer-based world models with contrastive predictive coding. In *International Conference on Learning Representations*, volume 2025, pp. 6949–6970, 2025.

Weilin Cong, Si Zhang, Jian Kang, Baichuan Yuan, Hao Wu, Xin Zhou, Hanghang Tong, and Mehrdad Mahdavi. Do we really need complicated model architectures for temporal networks? In *International Conference on Learning Representations*, 2023.

Tal Daniel, Carl Qi, Dan Haramati, Amir Zadeh, Chuan Li, Aviv Tamar, Deepak Pathak, and David Held. Latent particle world models: Self-supervised object-centric stochastic dynamics modeling. 2026. URL https://arxiv.org/abs/2603.04553.

Elizabeth Dinella, Hanjun Dai, Ziyang Li, Mayur Naik, Le Song, and Ke Wang. Hoppity: Learning graph transformations to detect and fix bugs in programs. In *International conference on learning representations*, 2020.

Jingtao Ding, Yunke Zhang, Yu Shang, Yuheng Zhang, Zefang Zong, Jie Feng, Yuan Yuan, Hongyuan Su, Nian Li, Nicholas Sukiennik, Fengli Xu, and Yong Li. Understanding world or predicting future? A comprehensive survey of world models. *ACM Comput. Surv.*, 58(3): 57:1–57:38, 2026.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pp. 1329–1338. PMLR, 2016.

Tao Feng, Yexin Wu, Guanyu Lin, and Jiaxuan You. Graph world model. In *International Conference on Machine Learning*, volume 267, 2025.

David Ha and Jürgen Schmidhuber. World models. 2018. doi: 10.5281/ZENODO.1207631. URL https://zenodo.org/record/1207631.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Nick Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations*, volume 2024, pp. 47376–47405, 2024.

Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael M. Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. Temporal graph benchmark for machine learning on temporal graphs. *Advances in Neural Information Processing Systems*, 36, 2023.

Yinan Huang, William Lu, Joshua Robinson, Yu Yang, Muhan Zhang, Stefanie Jegelka, and Pan Li. On the stability of expressive positional encodings for graphs. In *International Conference on Learning Representations*, 2024.

Yaning Jia, Chunhui Zhang, and Soroush Vosoughi. Aligning relational learning with lipschitz fairness. In *International Conference on Learning Representations*, 2024.

Taehyeon Kim and Kyung-Taek Lee. Efficiency-aware group size optimization for grpo via multi-fidelity bayesian optimization. *AI*, 7(7):234, 2026.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Thomas N. Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. In *8th International Conference on Learning Representations*, 2020.

Srijan Kumar, Xikun Zhang, and Jure Leskovec. Predicting dynamic embedding trajectory in temporal interaction networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1269–1278, 2019.

Andrey Kurenkov, Michael Lingelbach, Tanmay Agarwal, Emily Jin, Chengshu Li, Ruohan Zhang, Li Fei-Fei, Jiajun Wu, Silvio Savarese, and Roberto Martín-Martín. Modeling dynamic environments with scene graph memory. In *International Conference on Machine Learning*, volume 202, pp. 17976–17993. PMLR, 2023.

Jiawei Liu, Senqiao Yang, Mingjun Wang, Yu Wang, and Bei Yu. Graph world models: Concepts, taxonomy, and future directions, 2026. URL `https://arxiv.org/abs/2604.27895`.

Junfeng Liu, Min Zhou, Shuai Ma, and Lujia Pan. Mata*: Combining learnable node matching with a* algorithm for approximate graph edit distance computation. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pp. 1503–1512, 2023.

Xiaodong Lu, Leilei Sun, Tongyu Zhu, and Weifeng Lv. Improving temporal link prediction via temporal walk matrix projection. *Advances in Neural Information Processing Systems*, 37:141153–141182, 2024.

Sohir Maskey, Raffaele Paolino, Fabian Jogl, Gitta Kutyniok, and Johannes F. Lutzeyer. Graph representational learning: When does more expressivity hurt generalization? In *International Conference on Learning Representations*, 2026.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

Galileo Namata, Ben London, Lise Getoor, Bert Huang, and U Edu. Query-driven active surveying for collective classification. In *10th international workshop on mining and learning with graphs*, volume 8, pp. 1, 2012.

Jie Peng, Zhewei Wei, and Yuhang Ye. TIDFormer: Exploiting temporal and interactive dynamics makes a great dynamic graph transformer. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 2245–2256, 2025.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, Vikash Kumar, and Wojciech Zaremba. Multi-goal reinforcement learning: Challenging robotics environments and request for research. 2018. URL https://arxiv.org/abs/1802.09464.

Farimah Poursafaei, Shenyang Huang, Kellin Pelrine, and Reihaneh Rabbany. Towards better evaluation for dynamic link prediction. *Advances in Neural Information Processing Systems*, 35: 32928–32941, 2022.

Ladislav Rampásek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, 2022.

Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael M. Bronstein. Temporal graph networks for deep learning on dynamic graphs, 2020. URL https://arxiv.org/abs/2006.10637.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Weining Shi, Zhisen Wen, Qinggang Zhang, Chentao Zhang, and Zhihong Zhang. Temporal graph thumbnail: Robust representation learning with global evolutionary skeleton. In *International Conference on Learning Representations*, volume 2026, pp. 137964–137999, 2026.

Geri Skenderi, Hang Li, Jiliang Tang, and Marco Cristani. Graph-level representation learning with joint-embedding predictive architectures, 2023. URL https://arxiv.org/abs/2309.16014.

Yuxing Tian, Yiyan Qi, and Fan Guo. Freedyg: Frequency enhanced continuous-time dynamic graph model for link prediction. In *International Conference on Learning Representations*, volume 2024, pp. 8431–8450, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Cheng Wan, Youjie Li, Ang Li, Nam Sung Kim, and Yingyan Lin. Bns-gcn: Efficient full-graph training of graph convolutional networks with partition-parallelism and random boundary node sampling. *Proceedings of Machine Learning and Systems*, 4:673–693, 2022.

Peihao Wang, Ruisi Cai, Yuehao Wang, Jiajun Zhu, Pragya Srivastava, Zhangyang Wang, and Pan Li. Understanding and mitigating bottlenecks of state space models through the lens of recency and over-smoothing. In *International Conference on Learning Representations*, volume 2025, pp. 82314–82342, 2025a.

Shuo Wang, Bokui Wang, Zhixiang Shen, Boyan Deng, and Zhao Kang. Multi-domain graph foundation models: Robust knowledge transfer via topology alignment. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025b.

Yanbang Wang, Yen-Yu Chang, Yunyu Liu, Jure Leskovec, and Pan Li. Inductive representation learning in temporal networks via causal anonymous walks. In *International Conference on Learning Representations*, 2021.

Qitian Wu, Wentao Zhao, Zenan Li, David P. Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Qitian Wu, Kai Yang, Hengrui Zhang, David Wipf, and Junchi Yan. Sgformer: Simplifying and scaling graph transformers with single-layer attention and approximation-free linear complexity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–16, 2026.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Da Xu, Chuanwei Ruan, Evren Korpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *International Conference on Learning Representations*, 2020.

Le Yu, Leilei Sun, Bowen Du, and Weifeng Lv. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems*, 36:67686–67700, 2023.

Lunjun Zhang, Ge Yang, and Bradly C. Stadie. World model as a graph: Learning latent landmarks for planning. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 12611–12620. PMLR, 2021.

Shaobin Zhuang, Zhipeng Huang, Ying Zhang, Fangyikang Wang, Canmiao Fu, Binxin Yang, Chong Sun, Chen Li, and Yali Wang. Video-gpt via next clip diffusion. In *International Conference on Learning Representations*, volume 2026, pp. 156161–156191, 2026.

Tao Zou, Yuhao Mao, Junchen Ye, and Bowen Du. Repeat-aware neighbor sampling for dynamic graph learning. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 4722–4733, 2024.

CONTENTS

1 Introduction **1**

2 Problem Formulation **2**

3 WorldGraph **3**

3.1 State-aware Graph Transformer 3

3.1.1 Stage 1: Multi-granularity Graph Encoder 4

3.1.2 Stage 2: History-aware State Encoder 4

3.1.3 Theoretical Analysis of State-Aware Graph Transformer 5

3.2 Transition-Aware Reinforcement Learning 5

3.2.1 Transition-Aware Dynamic Group Sampling 5

3.2.2 Structure-aware Reward Design 6

3.2.3 Theoretical Analysis of Transition-Aware RL Algorithm 6

4 Experiments **6**

4.1 Experimental Setup 7

4.2 Overall Performance (**RQ1**) 8

4.3 Ablation Study (**RQ2**) 8

4.4 Sensitivity Analysis (**RQ3**) 8

4.5 Comparison on Existing GWM Problem Settings (**RQ4**) 9

5 Conclusion **9**

A Notations **16**

B Additional Technical Details **17**

B.1 Detailed Design of State-aware Graph Transformer 17

B.1.1 Details of Multi-granularity Graph Encoder 17

B.1.2 Details of History-aware State Encoder 18

B.2 Detailed Design of Transition-aware RL Algorithm 18

B.2.1 Details of Transition-Aware Dynamic Group Sampling 18

B.2.2 Structure-aware Reward Design 18

B.3 Details of Task-specific Structure-aware Reward Calculation 18

B.4 Pseudo-code of Transition-aware RL 19

C Proof of Theorems **21**

C.1 Proof of Theorem 1 21

C.2 Proof of Theorem 2 21

D Experimental Validation of Theoretical Results **22**

D.1 Validation of the Graph-Encoder Bound 22
D.2 Validation of the RL Variance Bound 22

E Time and Space Complexity Analysis of WorldGraph. 23
E.1 Multi-Granularity Graph Encoding. 23
E.2 History-Aware State Encoding. 23
E.3 Transition-Aware RL. 23

F Additional Datasets and Baselines Information 24
F.1 Statistics of Datasets 24
F.2 Statistics of Baselines 24
F.3 Why We Choose These Datasets and Baselines 25

G Experimental details 27
G.1 Details of WorldGraph Pretraining 27
G.2 Details of Task Information 27
G.3 Details of Plug-in Integration and Traditional Graph Prediction 27
G.4 Details of Hardware Information 27
G.5 Details of Parameter setup. 28

H More Experiments 29
H.1 Efficiency Analysis of WorldGraph. 29
H.2 Multi-step Prediction Analysis. 29
H.3 Additional Sensitivity Analysis 29
H.4 Image-to-Graph Visual Rollout. 31

I More Related Works 32
I.1 World Models. 32
I.2 Temporal Graph Learning. 32

J Reproducibility and Code Availability 32

K Limitations and Broader Impacts 32
K.1 Limitations 32
K.2 Broader Impacts 32

A NOTATIONS

We provide the detailed notation in Table 4.

Table 4: Core notations used in WorldGraph.

Symbol	Description
Graph World	
$t,\ \tau,\ T$	Current time, historical time, and sequence length.
$g_t = (V_t, E_t, X_t) \in \mathcal{G}$	Observed graph at time t, its node, edge, and property sets, and the graph space.
Δg_t	Graph change associated with the transition from g_t to g_{t+1}.
$\mathbf{s}_t \in \mathcal{S},\ f_s$	Latent world state, world-state space, and state model.
f_Δ	Transition model that predicts Δg_t from the current graph, the preceding graph change, and $\mathbf{s}_t$.
$\mathbf{z}_{v,t},\ \mathbf{z}_t,\ \mathbf{s}_{v,t}$	Structure-aware node embedding, graph embedding, and node-level latent state at time t.
State-Aware Graph Transformer	
$L_{\max},\ \ell$	Maximum hop number and hop index.
$\mathbf{x}_v^{(\ell)},\ \mathbf{x}_v^{\mathrm{path}_i}$	Hop-level and i-th random-walk-path embeddings of node v.
$M,\ \mathrm{path}_i$	Number of sampled random-walk paths and the i-th path.
$\mathbf{x}_{\Delta g_\tau},\ \mathrm{Type}(\Delta g_\tau)$	Transition representation and learnable transition-type embedding for Δg_τ.
$\mathbf{m}_{v,\tau},\ \mathrm{Dist}(t,\tau),\ t_{\max}$	Historical memory of node v, relative temporal-distance embedding, and maximum historical time span.
$\mathbf{W}_x,\ \mathbf{W}_z,\ \mathbf{W}_a$	Learnable projections of node history, graph history, and transition-type information in the node memory.
$\mathbf{q}_{v,t},\ \beta_{v,t,\tau},\ \tilde{\beta}_{v,t,\tau},\ d$	History-attention query, attention score, normalized attention weight, and projected embedding dimension.
$\mathbf{p}_{t-\tau},\ \Pi(\Delta g_\tau, \Delta g_{t-1})$	Relative-time attention bias and similarity between a historical transition and the latest observed transition.
$\mathbf{W}_Q,\ \mathbf{W}_K,\ \mathbf{W}_V$	Learnable query, key, and value projections for history attention.
$\mathbf{h}_{v,t},\ \tilde{\mathbf{h}}_{v,t}$	Single-head history embedding and the final multi-head history embedding of node v.
$g^{(\mathrm{SG})},\ g_i^{(\mathrm{OURS})}$	An SGFormer subgraph and a WorldGraph sampled subgraph.
$g_i^{(\mathrm{IDEAL})},\ \zeta$	An ideal full-coverage subgraph and the ideal coverage count.
$\mathbf{s}_{v,t}^{(\mathrm{SG})},\ \mathbf{s}_{v,t}^{(\mathrm{OURS})}$	Final latent states induced by SGFormer and WorldGraph.
$\mathbf{s}_{v,t}^{(\mathrm{IDEAL})}$	Final latent state induced by the ideal full-coverage encoder.
$\Delta_{\mathrm{SG}},\ \Delta_{\mathrm{OURS}}$	Deviations of SGFormer and WorldGraph from the ideal latent state.
Transition-Aware RL Algorithm	
$\mathcal{J}_t,\ J_i,\ \lvert J_i\rvert$	Set of transition-conditioned groups at time t, its i-th group, and the group size.
$\widehat{p}(J_i),\ \omega(J_i),\ \gamma$	Historical transition frequency, rarity weight, and rarity-weight exponent of group J_i.
$\mathrm{Prob}(J_i),\ \mathcal{B},\ B_t(J_i)$	Selection probability, total rollout budget, and budget allocated to group J_i.
$\rho^{\mathrm{GRPO}},\ \rho^{\mathrm{OURS}},\ \rho_i,\ C_a$	Group sizes under standard GRPO and Transition-aware RL, a generic group size, and the base group-size constant.
$\mathrm{EVR}(\rho_i),\ M_{\mathrm{EVR}},\ \delta$	Effective variance ratio, its residual term, and reward-distribution kurtosis.
$\mathcal{I}_{\mathbf{s}}(v),\ \deg(v),\ \theta$	Degree-based structural importance, node degree, and degree-scaling exponent.
$f_{<t}(v),\ \mathcal{I}_{\mathbf{v}}(v),\ \eta$	Historical change frequency, variability importance, and variability-scaling exponent.
$\mathcal{I}(v),\ V_t$	Normalized importance of node v and the current node set over which it is normalized.
$\Delta\hat{Q},\ \Delta Q$	Predicted and ground-truth change sets.
$\mathcal{P},\ \mathcal{R},\ R_{\mathrm{structure}}$	Importance-weighted precision, recall, and their harmonic-mean structure reward.
$\lambda,\ R_{\mathrm{property}},\ R$	Structural/property reward balancing coefficient, property-change reward, and unified task reward.

B Additional Technical Details

B.1 Detailed Design of State-aware Graph Transformer

B.1.1 Details of Multi-granularity Graph Encoder

Details of Hop-Level Message Passing. Given the maximum hop number $L_{\max}$, for a given graph g_t, we perform message passing for $\ell = \{1, 2, \cdots, L_{\max}\}$ over the neighbors of each node to obtain the initial node embeddings. Let $\mathcal{N}(v)$ denote the neighbor set of node v. We compute the node embedding of v:

$$\mathbf{x}_v^{(\ell)} = \mathrm{ReLU}\left(\mathbf{W}_{\text{self}}^{(\ell)} \mathbf{x}_v^{(\ell-1)} + \frac{\sum_{u \in \mathcal{N}(v)} w_{uv}\, \mathbf{W}_{\text{neighbor}}^{(\ell)} \mathbf{x}_u^{(\ell-1)}}{\max\left(1, \sum_{u \in \mathcal{N}(v)} w_{uv}\right)} \right) \tag{5}$$

, where w_{uv} is set to the currently observed edge weight for a weighted graph, and to 1 for an unweighted graph, in which case the neighborhood term reduces to mean aggregation. $\mathbf{W}_{\text{self}}^{(\ell)}$ and $\mathbf{W}_{\text{neighbor}}^{(\ell)}$ denote the learnable weight matrices at layer ℓ. Thus, we obtain node embeddings computed through multi-layer message passing based on neighbors at different hop distances and, when available, their observed relation strengths.

Details of Random-Walk Path Sampling. We use projected scaled dot-product attention to aggregate the node embeddings along path_i. Specifically, for each node $v_j \in \text{path}_i$, we compute the attention score with respect to the starting node v_0 as

$$\text{Score}_{i,j} = \frac{\left(\mathbf{W}_{\text{path}} \mathbf{x}_{v_j}^{(L_{\max})}\right)^\top \left(\mathbf{W}_{\text{path}} \mathbf{x}_{v_0}^{(L_{\max})}\right)}{\sqrt{d}}, \tag{6}$$

where $\mathbf{W}_{\text{path}}$ is a learnable projection for path attention and d is the dimension of the projected node embedding. We normalize the scores over the nodes on the same path:

$$\alpha_{i,j} = \frac{\exp(\text{Score}_{i,j})}{\sum_{k=0}^{T_i} \exp(\text{Score}_{i,k})}. \tag{7}$$

The embedding corresponding to the i-th path is then obtained by weighted aggregation:

$$\mathbf{x}_v^{\text{path}_i} = \sum_{j=0}^{T_i} \alpha_{i,j}\, \mathbf{x}_{v_j}^{(L_{\max})}. \tag{8}$$

Thus, we derive $\mathbf{x}_v^{\text{path}_i}$ for different sampled paths path_i, capturing multi-step structural information around node v from different random-walk trajectories.

After obtaining the hop-level embeddings $\{\mathbf{x}_v^{(\ell)}\}_{\ell=1}^{L_{\max}}$ and the path-level embeddings $\{\mathbf{x}_v^{\text{path}_i}\}_{i=1}^{M}$, we fuse all structural representations of node v via an attention-based aggregation to obtain a structure-aware embedding $\mathbf{z}_{v,t}$.

Details of Multi-Granularity Attention Fusion. Then, we collect all candidate representations into a unified set $\mathcal{S}_v = \left\{\mathbf{x}_v^{(1)}, \ldots, \mathbf{x}_v^{(L_{\max})}, \mathbf{x}_v^{\text{path}_1}, \ldots, \mathbf{x}_v^{\text{path}_M}\right\}$. Let $\{\mathbf{u}_{v,k}\}_{k=1}^{L_{\max}+M}$ denote the candidates in $\mathcal{S}_v$, where $\mathbf{u}_{v,k} = \mathbf{x}_v^{(k)}$ for $1 \le k \le L_{\max}$ and $\mathbf{u}_{v,k} = \mathbf{x}_v^{\text{path}_{k-L_{\max}}}$ for $L_{\max} < k \le L_{\max} + M$. We compute an unnormalized attention score between each candidate $\mathbf{u}_{v,k}$ and the last-hop embedding of node v as $\alpha_{v,k} = \frac{(\mathbf{W}_{\text{fusion}} \mathbf{u}_{v,k})^\top \left(\mathbf{W}_{\text{fusion}} \mathbf{x}_v^{(L_{\max})}\right)}{\sqrt{d}}$, where $\mathbf{W}_{\text{fusion}}$ is a learnable projection for fusing the hop- and path-level representations. We then normalize the scores over all candidates via a softmax: $\tilde{\alpha}_{v,k} = \frac{\exp(\alpha_{v,k})}{\sum_{k'=1}^{L_{\max}+M} \exp(\alpha_{v,k'})}$. The final structure-aware embedding is retained as the node-level world state:

$$\mathbf{z}_{v,t} = \sum_{k=1}^{L_{\max}+M} \tilde{\alpha}_{v,k}\, \mathbf{u}_{v,k}. \tag{9}$$

B.1.2 Details of History-aware State Encoder

Details of Query Embedding. For each node v, we construct a query embedding from its latest node-level world state, the global graph summary, the latest transition, and its previous latent state: $\mathbf{q}_{v,t} = [\mathbf{z}_{v,t};\, \mathbf{z}_t;\, \mathbf{x}_{\Delta g_{t-1}};\, \mathbf{s}_{v,t-1}]$, and aggregate its historical memory $\{\mathbf{m}_{v,\tau}\}$ with multi-head attention.

Details of Multi-Head Attention The same operation is performed in parallel by all attention heads. For one attention head, the history embedding is obtained by weighted aggregation of the value projections:

$$\mathbf{h}_{v,t} = \sum_{\tau} \widetilde{\beta}_{v,t,\tau} \left(\mathbf{m}_{v,\tau}\mathbf{W}_V\right). \tag{10}$$

The outputs from all heads are concatenated and mixed by an MLP to obtain the final history embedding:

$$\widetilde{\mathbf{h}}_{v,t} = \mathrm{MLP}\Big(\left[\mathbf{h}_{v,t}^{(1)}; \mathbf{h}_{v,t}^{(2)}; \ldots; \mathbf{h}_{v,t}^{(H)}\right]\Big). \tag{11}$$

B.2 Detailed Design of Transition-aware RL Algorithm

B.2.1 Details of Transition-Aware Dynamic Group Sampling

Details of Rarity-based Rollout Budget Allocation Finally, to make earlier rare transitions receive a larger effective group size, we allocate the total rollout budget $\mathcal{B}$ across groups proportionally. We reserve the same minimum number of rollouts[9] for every group and use largest-remainder rounding for the remaining budget, such that $\sum_{j\in\mathcal{J}_t} B_t(j) = \mathcal{B}$, where $B_t(j)$ is the allocated number of rollouts for transition group j at time step t. Thus, groups corresponding to transitions that are rarer in the prefix $\{\Delta g_\tau\}_{\tau<t}$ are assigned more rollout samples at time t, increasing the relative training signal for these frequency-imbalanced modification patterns.

B.2.2 Structure-aware Reward Design

Details of Node/Edge/Graph Importance Calculation. The three graph-world tasks produce outputs at different granularities. We therefore retain the same structural importance $\mathcal{I}(v)$, while defining a verifiable reward consistent with the output and evaluation metric of each task. For a changed edge $e = (u, v)$, its importance is determined by its two nodes: $\mathcal{I}(e) = \frac{\mathcal{I}(u)+\mathcal{I}(v)}{2}$. For a changed local graph centered at node v, its importance is $\mathcal{I}(v)$.

Although the three graph-world tasks operate at different granularities, their rewards share the same principle: a prediction should identify the correct structural changes and estimate the corresponding representation changes. We therefore use the unified reward

$$R = \lambda R_{\text{structure}} + (1-\lambda) R_{\text{property}}, \tag{12}$$

where R_{property} is calculated based on the ground-truth updated property and the predicted property. λ balances the two terms.

B.3 Details of Task-specific Structure-aware Reward Calculation

Node-level tasks. Let V_{t+1}^{add}, V_{t+1}^{delete}, and $\Delta V_{t+1}^{\text{property}}$ denote the ground-truth sets of added, deleted, and property-changed nodes, respectively; their hatted versions denote the corresponding predictions. For a property-changed node v, $\Delta\mathbf{x}_{v,t+1}$ and $\widehat{\Delta\mathbf{x}}_{v,t+1}$ denote its ground-truth and predicted property changes. The importance-weighted property reward is

$$R_{\text{property}}^{\text{node}} = \exp\left(-\frac{\sum_{v\in\Delta V_{t+1}^{\text{property}}} \mathcal{I}(v)\ \mathrm{MAE}\Big(\widehat{\Delta\mathbf{x}}_{v,t+1}, \Delta\mathbf{x}_{v,t+1}\Big)}{\sum_{v\in\Delta V_{t+1}^{\text{property}}} \mathcal{I}(v)}\right). \tag{13}$$

[9] In practice, we set the minimum number of rollouts as 2. We do this to ensure every transition group receives at least a minimal number of rollouts, preventing overly noisy return estimates from groups with too few samples.

Here, $R^{(\text{node})}_{\text{structure}}$ averages the set rewards for node addition, node deletion, and property-change localization. Setting $\lambda_{\text{node}} = 3/4$ in the unified reward gives equal weight to the four constituent terms:

$$R_{\text{node}} = \frac{1}{4}\big[R_{\text{structure}}\Big(\widehat{V}^{\text{add}}_{t+1}, V^{\text{add}}_{t+1}\Big) + R_{\text{structure}}\Big(\widehat{V}^{\text{delete}}_{t+1}, V^{\text{delete}}_{t+1}\Big) + R_{\text{structure}}\Big(\widehat{\Delta V}^{\text{property}}_{t+1}, \Delta V^{\text{property}}_{t+1}\Big) + R^{\text{node}}_{\text{property}}\big]. \tag{14}$$

Edge-level tasks. Let E^{add}_{t+1}, E^{delete}_{t+1}, and $\Delta E^{\text{property}}_{t+1}$ denote the ground-truth sets of added, deleted, and property-changed edges, respectively; their hatted versions denote the corresponding predictions. For a property-changed edge e, $\Delta \mathbf{x}_{e,t+1}$ and $\widehat{\Delta \mathbf{x}}_{e,t+1}$ denote its ground-truth and predicted property changes. The importance-weighted property reward is

$$R^{\text{edge}}_{\text{property}} = \exp\left(-\frac{\sum_{e\in\Delta E^{\text{property}}_{t+1}} \mathcal{I}(e)\ \text{MAE}\Big(\widehat{\Delta \mathbf{x}}_{e,t+1}, \Delta \mathbf{x}_{e,t+1}\Big)}{\sum_{e\in\Delta E^{\text{property}}_{t+1}} \mathcal{I}(e)}\right). \tag{15}$$

Here, $R^{(\text{edge})}_{\text{structure}}$ averages the set rewards for edge addition, edge deletion, and property-change localization. With $\lambda_{\text{edge}} = 3/4$, the edge-level reward is

$$R_{\text{edge}} = \frac{1}{4}\big[R_{\text{structure}}\Big(\widehat{E}^{\text{add}}_{t+1}, E^{\text{add}}_{t+1}\Big) + R_{\text{structure}}\Big(\widehat{E}^{\text{delete}}_{t+1}, E^{\text{delete}}_{t+1}\Big) + R_{\text{structure}}\Big(\widehat{\Delta E}^{\text{property}}_{t+1}, \Delta E^{\text{property}}_{t+1}\Big) + R^{\text{edge}}_{\text{property}}\big]. \tag{16}$$

Graph-level tasks. Let $\Delta V^{\text{graph}}_{t+1}$ and $\widehat{\Delta V}^{\text{graph}}_{t+1}$ denote the ground-truth and predicted sets of nodes whose local graphs change. The local-graph representation $\boldsymbol{\phi}_{v,t}$ contains node count, edge count, density, number of connected components, clustering coefficient, and triangle count. We use $\Delta\boldsymbol{\phi}_{v,t+1} = \boldsymbol{\phi}_{v,t+1} - \boldsymbol{\phi}_{v,t}$ and $\widehat{\Delta\boldsymbol{\phi}}_{v,t+1} = \widehat{\boldsymbol{\phi}}_{v,t+1} - \boldsymbol{\phi}_{v,t}$ for the ground-truth and predicted representation changes. When evaluating a sampled transition, the property-change magnitude is computed over its selected local-graph centres that also exhibit a ground-truth structural change. Their property reward is

$$R^{\text{graph}}_{\text{property}} = \exp\left(-\frac{\sum_{v\in\Delta V^{\text{graph}}_{t+1}} \mathcal{I}(v)\ \text{MAE}\Big(\widehat{\Delta\boldsymbol{\phi}}_{v,t+1}, \Delta\boldsymbol{\phi}_{v,t+1}\Big)}{\sum_{v\in\Delta V^{\text{graph}}_{t+1}} \mathcal{I}(v)}\right). \tag{17}$$

With $\lambda_{\text{graph}} = 1/2$, the graph-level reward equally combines change localization and representation-change magnitude:

$$R_{\text{graph}} = \frac{1}{2}\big[R_{\text{structure}}\Big(\widehat{\Delta V}^{\text{graph}}_{t+1}, \Delta V^{\text{graph}}_{t+1}\Big) + R^{\text{graph}}_{\text{property}}\big]. \tag{18}$$

In practice, we add a small number (e.g., 10^{-8}) when computing the reward to avoid the numerical instability caused by a zero denominator.

B.4 Pseudo-code of Transition-aware RL

Algorithm 1 summarizes the training and inference procedure of the proposed transition-aware reinforcement learning algorithm. At each training step, the input is the current graph g_t, its observed historical transitions and latent states, and the next graph g_{t+1} used only as supervision. The state model first encodes g_t and its history into $\mathbf{s}_t$. From transitions observed before t, we then construct transition-conditioned groups $\mathcal{J}_t$, with one group corresponding to each transition type observed in the history. We estimate the historical frequency $\widehat{p}_t(J_i)$ of each group, convert it to the rarity weight $\omega_t(J_i) = \widehat{p}_t(J_i)^{-\gamma}$, and normalize these weights to obtain the group allocation probabilities. Every group receives the same base number C_a of rollouts, while the remaining budget is allocated according to these probabilities, giving relatively more samples to transition types that are less frequently observed.

For each allocated rollout, the first operation is constrained to the corresponding group; for example, in a node-level task, one group may start with node addition, another with node deletion, and another

Algorithm 1 Transition-aware RL Training and Inference

Require: Chronological graph sequence $\{g_t\}_{t=1}^{T}$, state model f_s, transition Controller f_Δ, rollout budget $\mathcal{B}$, base group size C_a, exponents γ, θ, η, and reward coefficient λ
Ensure: Trained WorldGraph model

```
Initialize the state model, transition Controller, and task-specific prediction heads
for t = 1, 2, ..., T − 1 do
    Encode the current graph and its history to obtain s_t = f_s(g_t, Δg_{t−1}, s_{t−1})
    Compute the degree-based and historical-variability importance terms and obtain I(v) using θ and η
    Construct the transition-conditioned groups J_t from transitions observed before time t
    for each group J_i ∈ J_t do
        Estimate its historical frequency p̂(J_i) and compute ω(J_i) = p̂(J_i)^{−γ}
    end for
    Compute Prob(J_i) by normalizing the rarity weights over J_t
    Allocate B_t(J_i) rollouts to each group, reserving C_a rollouts per group and distributing the remaining budget according to Prob(J_i)
    for each group J_i ∈ J_t do
        for b = 1, 2, ..., B_t(J_i) do
            Force the first operation to belong to J_i
            Sample subsequent valid operations and target nodes or edges from f_Δ until the stopping decision
            Decode the sampled operation sequence as a predicted change set ΔQ̂
            Compare ΔQ̂ with the ground-truth change set ΔQ and compute R_structure
            Compute the task-specific property reward R_property
            Compute R = λR_structure + (1 − λ)R_property
        end for
    end for
    Normalize the rewards within each transition-conditioned group to obtain group-relative advantages
    Update f_Δ with the clipped GRPO objective using all sampled rollouts and their group-relative advantages
end for
At inference time, use the trained Controller to provide the transition conditioning, and use the task-specific prediction heads to decode the final task output from the predicted future latent state
return the trained WorldGraph model
```

with a node feature change. The Controller then samples subsequent valid operations and target nodes or edges until it stops. The resulting operation sequence is decoded as a predicted change set $\Delta\hat{Q}$ (together with the task-specific representation change). Comparing it with the ground-truth change set ΔQ and representation change from g_{t+1} gives the structure reward $R_{\text{structure}}$ and property reward R_{property}, which are combined into the scalar return $R = \lambda R_{\text{structure}} + (1-\lambda) R_{\text{property}}$. We use GRPO to compare these returns only among rollouts that began with the same edit type. Its core idea is to avoid fitting a separate value or critic network: rewards are normalized within each transition-conditioned group to form group-relative advantages. A rollout with an above-group-average return receives a positive advantage, whereas a below-average rollout receives a negative one; the clipped policy-ratio objective then increases the probability of the former and decreases the probability of the latter. At inference time, the trained Controller supplies the transition conditioning and the task-specific prediction heads decode the final output from the predicted future latent state.

C Proof of Theorems

C.1 Proof of Theorem 1

Proof. From Equation 3, the final latent state can be written as

$$\mathbf{s}_{v,t}^{(\text{method})} = \mathcal{F}\left(\mathbf{z}_{v,t}^{(\text{method})}\right). \tag{19}$$

$\mathcal{F}$ is a deterministic function induced by the structure and history information with the same Δg_{t-1}, $\mathbf{s}_{v,t-1}$[10], and model parameters.

Following existing theoretical analysis of graph learning methods (Huang et al., 2024; Maskey et al., 2026; Jia et al., 2024), we consider that the message function $\mathcal{F}(\cdot)$ is C-Lipschitz. Then the deviation between a method and the ideal model is bounded by the induced deviation of the aggregated inputs:

$$\Delta_{\text{method}} = \left\|\mathbf{s}_{v,t}^{(\text{method})} - \mathbf{s}_{v,t}^{(\text{IDEAL})}\right\|_2 \leq C_L \cdot \left\|\mathbf{z}_{v,t}^{(\text{method})} - \mathbf{z}_{v,t}^{(\text{IDEAL})}\right\|_2$$

where C_L is a constant.

Based on Equation 9, we can model $\mathbf{z}_{v,t}^{(\text{method})} = \sum_i \psi(g_i)$. In our state-aware graph transformer (SGT), $\psi(g_i) = \tilde{\alpha}_i \mathbf{u}_{v,k}$, as defined in Equation 9. To avoid representation drift of the same subgraph instance induced by different hyperparameters in attention designs (e.g., the number of attention heads), we use a consistent attention encoder across SGFormer, SGT (ours), and the ideal model. Consequently, for any training subgraph g_i, $\psi_{\text{SGFormer}}(g_i) = \psi_{\text{OURS}}(g_i) = \psi_{\text{IDEAL}}(g_i) = \psi(g_i)$. Following (Wan et al., 2022), we can bound $\psi(g_i)$ by a constant C_p. $\tilde{\alpha}_i$ is bounded by 1 because it is exponentially normalized.

Therefore, $\left\|\mathbf{z}^{(\text{method})} - \mathbf{z}^{(\text{IDEAL})}\right\|_2 = \|\sum_{g_i \notin g^{(\text{method})}} \psi(g_i)\|_2 \leq \|\sum_{g_i \notin g^{(\text{method})}} C_p\|_2$. Then, the upper bound of $\Delta_{\text{OURS}} = \left\|\mathbf{z}^{(\text{OURS})} - \mathbf{z}^{(\text{IDEAL})}\right\|_2$ is $(\zeta - (L_{\max} + M))\|C_p\|_2$. The upper bound of $\Delta_{\text{SGFormer}} = \left\|\mathbf{z}^{(\text{SGFormer})} - \mathbf{z}^{(\text{IDEAL})}\right\|_2$ is $(\zeta - 1)\|C_p\|_2 > (\zeta - (L_{\max} + M))\|C_p\|_2$. Therefore, the upper bound of Δ_{OURS} is smaller than that of Δ_{SG}.

□

C.2 Proof of Theorem 2

Proof. Because GRPO uses a fixed group size, the group size corresponding to each transition type is a constant C_a. Hence,

$$\rho^{\text{GRPO}} = C_a. \tag{20}$$

Our RL algorithm uses a dynamic group size based on the transition frequencies. As discussed in Footnote 6, we assign a minimum number of rollouts to different transition types. In this case, each transition type corresponds to the same group size as in GRPO. Then, based on the transition frequency, we allocate additional rollout budget to transitions with larger/minimum counts (i.e., larger group sizes) for different transition types, which leads to

$$\rho^{\text{OURS}} = C_a + \text{Prob}(J_i) \cdot \mathcal{B} > \rho^{\text{GRPO}} = C_a. \tag{21}$$

From (Kim & Lee, 2026), $\text{EVR}(\rho) = 1 + \frac{\delta - 1}{2\rho} + M_{\text{EVR}}$. Following (Kim & Lee, 2026), we consider $\delta > 1$. Since $\text{EVR}(\rho)$ is monotonically decreasing with respect to ρ,

$$\text{EVR}(\rho^{\text{OURS}}) < \text{EVR}(\rho^{\text{GRPO}}). \tag{22}$$

This completes the proof.

□

[10] Since SGFormer is designed for static graph scenarios, we use the same $\mathbf{s}_{t-1}$ to mitigate the effects brought by the quality of temporal encodings, thereby allowing the model to focus more on graph-structure encoding.

Table 5: Experimental validation of Theorem 1 on Graph-level tasks/Flights. Values are mean ± standard deviation over five checkpoints; lower is better.

Encoder setting	$\|\mathbf{z}_{v,t} - \mathbf{z}_{v,t}^{(\mathrm{IDEAL})}\|_2$	$\|\mathbf{s}_{v,t} - \mathbf{s}_{v,t}^{(\mathrm{IDEAL})}\|_2$
SGFormer (Single-view)	11.2743 ± 0.3400	6.2460 ± 0.9255
WorldGraph (Multi-view SGT)	**3.8210 ± 0.5362**	**1.5568 ± 0.3691**

Table 6: Experimental validation of Theorem 2 on Node-level tasks/TGBN-Trade. Mean EVR is the arithmetic mean over the three transition groups; lower is better.

Method	Mean EVR
Standard GRPO	1.5550
Transition-aware RL	**1.3526**

D Experimental Validation of Theoretical Results

D.1 Validation of the Graph-Encoder Bound

To empirically verify the ordering predicted by Theorem 1, we use five frozen WorldGraph checkpoints on Graph-level tasks/Flights and evaluate the same test transitions without further training. We construct a full-view reference representation using an expanded deterministic structural-view set, and compare it with (i) SGFormer, which retains only one structural view, and (ii) the default multi-view SGT encoder. We report the mean node-wise Euclidean distances to the reference representation and the corresponding hidden-state distances. As shown in Table 5, the default multi-view SGT has substantially smaller deviations than SGFormer for both $\mathbf{z}_{v,t}$ and $\mathbf{s}_{v,t}$, consistent with the smaller deviation bound established in Appendix C.1.

D.2 Validation of the RL Variance Bound

To empirically validate the ordering predicted by Theorem 2, we conduct a controlled experiment on Node-level tasks/TGBN-Trade. Standard GRPO uses a fixed base group size C_a for each transition type. Transition-aware RL retains the same base group size, then allocates the remaining rollout slots according to the estimated transition rarity until the configured total budget is exhausted. We report the first-order EVR approximation $1 + (\delta - 1)/(2\rho)$. As shown in Table 6, Transition-aware RL has a 13.02% lower mean EVR than standard GRPO, consistent with the ordering established in Appendix C.2.

E TIME AND SPACE COMPLEXITY ANALYSIS OF WORLDGRAPH.

Let $N = |V_t|$ and $E = |E_t|$ denote the numbers of nodes and edges in the current graph, respectively, and let d be the maximum hidden dimension. We further use $L_{\max}$ for the number of message-passing hops, M for the number of random walks per node, T_{rw} for their maximum length, L_{hist} for the history-window length, C_t for the number of legal transition candidates at time t, and $\mathcal{B}$ for the total rollout budget. The time complexity of WorldGraph consists of the following three parts.

E.1 MULTI-GRANULARITY GRAPH ENCODING.

The $L_{\max}$ hop-level message-passing layers require $O\big(L_{\max}(Ed + Nd^2)\big)$ time. Sampling M walks of length at most T_{rw} from every node costs $O(NMT_{\mathrm{rw}})$, while path encoding and attention fusion require $O\big(N(MT_{\mathrm{rw}} + L_{\max} + M)d^2\big)$. Therefore, the total time complexity of this component is

$$O\big(L_{\max}Ed + N(L_{\max} + MT_{\mathrm{rw}} + M)d^2\big)\,. \tag{23}$$

E.2 HISTORY-AWARE STATE ENCODING.

For each node, the state encoder uses one current query to attend to at most L_{hist} historical memories. The query, key, value, and feed-forward projections require $O(NL_{\mathrm{hist}}d^2)$ time, while computing and aggregating the attention weights requires $O(NL_{\mathrm{hist}}d)$ time. Hence, this component has complexity $O(NL_{\mathrm{hist}}d^2)$.

E.3 TRANSITION-AWARE RL.

Encoding the node states and scoring the legal node or edge candidates costs $O\big((N + C_t)d^2\big)$. Dynamic group statistics and rollout allocation depend only on the small set of transition groups and are negligible compared with candidate scoring. The encoded graph state and Controller logits are shared across the $\mathcal{B}$ rollouts; batched transition-conditioned state updates and structure-aware reward computation require at most $O\big(\mathcal{B}(Nd^2 + C_t + N + E)\big)$ time. For node- and graph-level tasks, C_t is typically linear in N. For edge-level tasks, C_t may reach $O(N^2)$ when all legal node pairs are considered, in which case candidate construction and scoring dominate the computation.

Combining the three components, the per-transition training complexity is

$$O\big(L_{\max}Ed + N(L_{\max} + MT_{\mathrm{rw}} + M + L_{\mathrm{hist}})d^2 + C_td^2 + \mathcal{B}(Nd^2 + C_t + N + E)\big)\,. \tag{24}$$

Since $L_{\max}$, M, T_{rw}, L_{hist}, and $\mathcal{B}$ are bounded hyperparameters, WorldGraph scales linearly with the observed graph size apart from the task-dependent transition-candidate set. Its space complexity is

$$O(E + N(L_{\max} + MT_{\mathrm{rw}} + L_{\mathrm{hist}})d + C_td + \mathcal{B}(Nd + C_t))\,, \tag{25}$$

accounting for graph storage, hop/path representations, historical memories, candidate states, and rollout tensors.

F Additional Datasets and Baselines Information

F.1 Statistics of Datasets

We evaluate GWM-Zero on eight real-world dynamic graph datasets. Their statistics are summarized in Table 7. Contact interactions are aggregated into causal six-hour snapshots, SocialEvo interactions into daily snapshots, and Enron interactions into weekly snapshots; the remaining datasets retain their released temporal granularity. We adopt the official TGB splits and chronological 70%/15%/15% splits otherwise. All preprocessing statistics and thresholds are fitted using the training split only, and predictions use only the observed graph history.

Table 7: Statistics of the eight real-world datasets used in GWM-Zero. The numbers of temporal events and snapshots are computed from the processed artifacts used by all methods.

Dataset	Domain	Nodes	Temporal Events	Snapshots	Edge Input	Snapshot Interval	Task(s)
TGBN-Trade	Economic interaction	255	468,245	31	Weighted	Annual	Node-Level Tasks, Edge-Level Tasks
TGBN-Genre	User preference	1,505	17,858,395	1,580	Weighted	Daily	Node-Level Tasks
TGBN-Reddit	Social interaction	11,766	27,174,118	1,090	Binary	Daily	Node-Level Tasks
UN Vote	Political interaction	201	1,035,742	72	Weighted	Annual	Edge-Level Tasks
Contact	Physical proximity	692	2,426,279	112	Binary	Six hours	Edge-Level Tasks, Graph-Level Tasks
SocialEvo	Social proximity	74	2,098,119	243	Binary	Daily	Edge-Level Tasks
Flights	Transportation	13,169	1,811,118	41	Binary	Monthly	Graph-Level Tasks
Enron	Communication	184	108,825	183	Binary	Weekly	Graph-Level Tasks

- **TGBN-Trade** (Huang et al., 2023) is a directed international trade network. Nodes denote countries and weighted temporal edges record trade relations. Its official dynamic country-property vectors support the node-level task, while its annual topology is also used for edge-level prediction.
- **TGBN-Genre** (Huang et al., 2023) is a weighted bipartite interaction graph between users and music-genre coordinates. The official daily genre-preference vectors are retained without target remapping or normalization and are used for node-level evolution prediction.
- **TGBN-Reddit** (Huang et al., 2023) is a large bipartite interaction graph between entities and subreddit coordinates. We use binary graph connectivity and retain the official 698-dimensional dynamic property vectors for the node-level task.
- **UN Vote** (Poursafaei et al., 2022) is an annual country interaction network. A directed weighted edge represents the number of co-yes voting relations between a country pair within an annual interval. It is used for edge addition and removal prediction.
- **Contact** (Poursafaei et al., 2022) records physical proximity among students. We aggregate the released events into causal six-hour binary snapshots and use the resulting sequence for both edge-level and graph-level evaluation.
- **SocialEvo** (Poursafaei et al., 2022) records temporal social-proximity interactions. After removing self-loops, we aggregate interactions into daily binary snapshots and use them for edge-level evaluation.
- **Flights** (Poursafaei et al., 2022) is a directed airport network in which an edge denotes an observed flight connection. For graph-level evaluation, we use the released native temporal snapshots and predict the next snapshot's local structural changes.
- **Enron** (Poursafaei et al., 2022) is a directed employee email network. We remove self-loops and aggregate the interactions into weekly binary snapshots for graph-level structural-evolution prediction.

F.2 Statistics of Baselines

We compare WorldGraph with eleven representative baselines covering graph encoders, graph Transformers, temporal graph models, pretrained graph models, and graph world models. Every method receives the same current graph, released history, data split, and task targets. For compatibility with the graph-world tasks, each baseline is equipped with supervised transition and count-prediction heads, while its original representation backbone is preserved.

- **MPNN.**

  - **GCN** (Kipf & Welling, 2017) performs normalized neighborhood aggregation through graph convolution.
  - **GAT** (Veličković et al., 2018) assigns learned attention weights to neighboring nodes during message passing.
  - **GraphSAGE** (Hamilton et al., 2017) constructs node representations using sampled-neighborhood aggregation.
- **Graph Transformers.**
  - **SGFormer** (Wu et al., 2026) combines simplified global self-attention with graph propagation for scalable graph representation learning.
  - **NodeFormer** (Wu et al., 2022) approximates all-pair node attention with kernelized message passing.
  - **GraphGPS** (Rampásek et al., 2022) combines a local message-passing module with a global Transformer module.
- **Temporal graph models.**
  - **TGN** (Rossi et al., 2020) maintains node-wise memories that are updated by time-stamped interactions and uses them for temporal prediction.
  - **TIDFormer** (Peng et al., 2025) models temporal intervals and evolving dependencies through interval-aware attention.
- **Pretrained graph models.**
  - **Graph-JEPA** (Skenderi et al., 2023) improves the structured representation of a graph world model through train-only self-supervised latent prediction on graph patches, followed by fine-tuning of its encoder and task adapter.
  - **MDGFM** (Wang et al., 2025b) transfers a leave-one-domain-out pretrained encoder that aligns topology across source domains before downstream adaptation.
- **Graph world model**[11].
  - **GWM-E** (Feng et al., 2025)[12] adopts the publicly released embedding-based GWM (GWM-E). It repeatedly propagates node properties over the graph to obtain multi-hop embeddings, maps the embedding at each hop into graph tokens through hop-specific projectors, and inserts these tokens into the task prompt so that a frozen LLaMA decoder can produce the final prediction.
  - **L^{3}P** (Zhang et al., 2021) identifies landmark states in the environment and constructs a reachability graph over them. It performs long-horizon planning by composing transitions between these landmarks, reducing the accumulation of errors from step-by-step prediction.
  - **C-SWM** (Kipf et al., 2020) encodes visual observations into object-centric latent nodes, uses a graph neural network to model their interactions and transitions, and learns the latent dynamics with a contrastive objective.

F.3 Why We Choose These Datasets and Baselines

The selected datasets cover economic, preference, social, political, proximity, transportation, and communication networks. They vary substantially in scale, temporal granularity, graph type, and edge semantics, ranging from 74 to 13,169 nodes and from approximately 10^5 to 2.7×10^7 temporal events. Moreover, the same collection supports complementary node-, edge-, and graph-level predictions, allowing us to evaluate whether a graph world model can consistently capture evolution at different structural granularities rather than overfitting to one task type.

The baselines provide complementary comparisons. GCN, GAT, and GraphSAGE test conventional local message passing; SGFormer, NodeFormer, and GraphGPS test global graph attention; TGN

[11] For L^{3}P and C-SWM, we preserve their original task-specific transition and decoder, and add World-Graph's multi-granularity graph encoder, history-aware state encoder, Controller, transition-aware dynamic group sampling, and structure-aware rewards. GWM-E does not model transition-conditioned temporal state transitions, so we insert only our multi-granularity graph encoder as a fusion adapter over its multi-hop graph tokens and retain its original LLM decoder.

[12] Feng et al. (2025) propose GWM-E and GWM-T. We compare our method with GWM-E, since it is publicly available and is reported as the stronger variant, whereas GWM-T is not publicly accessible.

and TIDFormer test dedicated temporal modeling; Graph-JEPA and MDGFM test whether graph pretraining alone is sufficient; L^3P and C-SWM test graph-structured planning and object-centric visual dynamics, respectively; and GWM-E tests graph-token conditioning of an LLM for traditional graph prediction tasks. This selection therefore separates the contributions of graph encoding, temporal state modeling, pretraining, graph-structured world modeling, graph-token task conditioning, and transition-aware reinforcement learning.

G EXPERIMENTAL DETAILS

G.1 DETAILS OF WORLDGRAPH PRETRAINING

WorldGraph-Pretrain adopts same-task leave-one-dataset-out pretraining: for each target dataset, we exclude it and use chronological transitions from the remaining datasets under the same node-, edge-, or graph-level task. Given a current graph g_t, we construct two views through property masking and independent random-walk sampling. Representations of the same node v in the two views form a positive pair, while representations of different sampled nodes serve as negatives; we optimize these pairs with a symmetric InfoNCE loss. We minimize the cosine distance between the two views' corresponding graph embeddings $\mathbf{z}_t$ and between their corresponding node-level latent states $\mathbf{s}_{v,t}$, and align the predicted next-step node representations with those encoded from g_{t+1}. The resulting multi-granularity graph encoder, history-aware state encoder, and latent predictor initialize the complete WorldGraph model for downstream training.

G.2 DETAILS OF TASK INFORMATION

(1) Node Level Tasks. We evaluate node addition, node deletion, and node property change on TGBN-Trade, TGBN-Genre, and TGBN-Reddit. Each released snapshot defines one discrete time step. A node is naturally added when it has no incident edge at the current step but has one at the next, and naturally removed in the reverse case. Because natural changes are sparse, we supplement them with shared nodes whose historical interaction activity is rising (for addition) or declining (for deletion), masking the node, its incident edges, and its property at the corresponding time step. Properties are given by the official dynamic node-property labels. For the nodes observed at both adjacent steps, we compute their Jaccard distance between the Top-10 positive property dimensions. We then sort these distances in ascending order and use the value at the 60th percentile as the threshold. If a node's Jaccard distance across adjacent steps exceeds this threshold, we mark it as indicating a property change. We report F1 score for addition, deletion, and property change, and NDCG@10 for future property prediction on changed nodes.

(2) Edge Level Tasks. We evaluate edge addition and deletion on TGBN-Trade, UN Vote, Contact, and SocialEvo. Given the current graph, the model predicts which disconnected node pairs will form edges and which existing edges will disappear, and then reconstructs the next graph. We report F1 score for edge addition and deletion and their macro-F1 score.

(3) Graph Level Tasks. We evaluate subgraph-level structural property changes on Flights, Contact, and Enron datasets. We use all nodes incident to at least one edge in the current graph as center nodes. For each selected node, we construct its local subgraph in both the current and next graphs, and predict the change in its six-dimensional structural property vector, consisting of node count, edge count, density, number of connected components, clustering coefficient, and triangle count. We report MAE and RMSE for these changes and macro-F1 for classifying them as decreasing, unchanged, or increasing.

G.3 DETAILS OF PLUG-IN INTEGRATION AND TRADITIONAL GRAPH PREDICTION

As discussed in Section 4.5, we evaluate WorldGraph in the following two settings. For L^3P and C-SWM, we preserve each model's original backbone and task-specific decoder while adding the multi-granularity graph encoder, history-aware state encoder, Controller, transition-aware dynamic group sampling, and structure-aware reward. Separately, we compare standalone WorldGraph with GWM-E on the latter's traditional graph prediction tasks, where GWM-E values are taken from the original reported results.

G.4 DETAILS OF HARDWARE INFORMATION

The main experiments are conducted on a server equipped with an Intel Xeon E5-2698 v4 CPU at 2.20 GHz (40 logical CPU cores), 256 GB RAM, and four NVIDIA Tesla V100 GPUs with 16 GB memory each. Each training process uses one GPU.

Table 8: Principal hyperparameters used by WorldGraph.

Hyperparameter	Node-level tasks	Edge-level tasks	Graph-level tasks
Latent-state dimension	64	32	64
Hidden-state dimension	64	32	64
Transition-representation dimension	32	16	32
Maximum historical time span	8	8	8
Dropout rate	0.1	0.1	0.1
Learning rate	10^{-3}	10^{-3}	10^{-3}
Total rollout budget	12	16	12

G.5 Details of Parameter setup.

All methods use the same chronological splits, graph inputs, history, and task targets. WorldGraph is optimized with AdamW using an eight-state history window. Checkpoints are selected on the validation set, and results are averaged over five runs. The main settings are summarized in Table 8.

All graph-state transitions are processed chronologically without access to future information, and every method uses the same task construction and evaluation protocol.

H More Experiments

H.1 Efficiency Analysis of WorldGraph.

We compare the mean epoch time, computed by averaging the first five training epochs, for WorldGraph and the eight baseline models on TGBN-Genre. The last two rows in Table 9 report only the pure reinforcement-learning overhead for standard GRPO and Transition-aware RL, respectively. WorldGraph w/o RL is somewhat slower than most representation and temporal baselines, but it is not the slowest; given the substantially stronger predictive performance reported in Table 1, this moderate computational cost is acceptable. For the RL comparison, Transition-aware RL incurs only 1.99 s more per epoch than standard GRPO (27.54 s compared with 25.55 s), indicating that the two key components of our RL design—transition-aware dynamic group sampling and structure-aware reward—introduce little additional runtime. Overall, these results show that WorldGraph provides a reasonable overall efficiency while delivering stronger predictive performance.

Table 9: Mean training time per epoch (seconds) on TGBN-Genre, computed over the first five training epochs.

Model	Time (s/epoch)
GCN	47.26
GAT	51.47
GraphSAGE	43.49
SGFormer	53.45
NodeFormer	72.11
GraphGPS	64.04
TGN	44.50
TIDFormer	55.10
WorldGraph w/o RL	66.89
Standard GRPO	25.55
Transition-aware RL	27.54

H.2 Multi-step Prediction Analysis.

We evaluate multi-step prediction on the *Graph-level tasks*, where the model forecasts changes in the structural property vector of local subgraphs. Each model is first trained with the standard one-step Graph-level protocol. At evaluation time, it observes the graph at an initial time, predicts the next graph-state change, and then recursively feeds its own predicted graph state into the same state-transition and prediction modules for steps 2, 3, and 5; ground-truth future graphs are used only to construct the evaluation targets. Following the Graph-level evaluation protocol, we report MAE and RMSE for these structural-property changes and Macro-F1 for classifying them as decreasing, unchanged, or increasing. Table 10 summarizes the results. WorldGraph-Pretrain is the best method on all nine Contact and Enron metrics, and on eight of the nine Flights metrics. Its Macro-F1 improvement over the strongest external baseline is $1.78\%/3.90\%/4.12\%$ on Flights, $6.36\%/6.19\%/7.60\%$ on Contact, and $1.10\%/1.45\%/1.28\%$ on Enron for Steps $2/3/5$, respectively. At Step 5, it also reduces MAE/RMSE by $0.4075/0.5832$ on Flights, $0.3861/0.5506$ on Contact, and $0.3789/0.4997$ on Enron relative to the best external values. These results show that our models retain a clear advantage as the prediction step increases, demonstrating effective multi-step graph-state prediction.

H.3 Additional Sensitivity Analysis

Figure 6 shows the sensitivity of WorldGraph to four additional hyperparameters on Graph-Level Tasks/Flights (Macro-F1): the rarity weight exponent γ, the degree importance exponent θ, the volatility importance exponent η, and the historical time span $t_{\max}$ used by the history-aware state encoder. Overall, the four curves exhibit an inverted U-shaped trend, showing that moving either below or above the default configuration can degrade performance. The default values $\gamma = 1.00$, $\theta = 1.00$, $\eta = 1.00$, and $t_{\max} = 8$ reach the common peak of 58.40%; for example, the rarity exponent gives 57.99% at 0.50 and 58.07% at 1.25, while the historical span gives 58.13%, 58.35%,

Table 10: Multi-step prediction results on the Graph-level tasks. Models recursively use their predicted graph states for steps 2, 3, and 5. MAE and RMSE are lower-is-better, whereas Macro-F1 is higher-is-better. The best results are shown in **bold** and the second best results are underlined.

Dataset	Model	Step 2			Step 3			Step 5		
		MAE	RMSE	Macro-F1	MAE	RMSE	Macro-F1	MAE	RMSE	Macro-F1
Flights	GCN	0.9208	1.5297	0.4486	1.3097	2.0548	0.3970	2.2527	3.2212	0.3584
	GAT	0.9080	1.4992	0.4314	1.1381	1.7816	0.3950	1.7601	2.5047	0.3372
	GraphSAGE	0.8764	1.4712	0.4667	1.1325	1.8040	0.4418	1.6862	2.4935	0.3982
	SGFormer	0.8789	1.4686	0.4735	1.1885	1.8848	0.4207	1.8938	2.7597	0.3861
	NodeFormer	0.9694	1.5545	0.4004	1.3759	1.9958	0.3755	2.2803	2.9962	0.3449
	GraphGPS	0.8799	1.4658	0.4834	1.2033	1.8941	0.4433	1.9271	2.8334	0.4045
	TGN	0.9161	1.4559	0.4467	1.4293	2.0275	0.3755	3.0440	3.8768	0.3411
	TIDFormer	0.8942	1.4159	0.4682	1.2958	1.9056	0.4236	2.5503	3.5035	0.3537
	Graph-JEPA	0.9502	1.5329	0.4564	1.6205	2.3081	0.4047	4.1542	5.1787	0.3511
	MDGFM	0.9480	1.5371	0.3762	1.3147	1.9468	0.3512	2.1888	2.9197	0.3284
	WorldGraph	**0.7248**	1.2904	0.4889	0.8976	1.4458	0.4694	1.3700	2.0019	0.4376
	WorldGraph-Pretrain	0.7251	**1.2879**	**0.5012**	**0.8660**	**1.4123**	**0.4823**	**1.2787**	**1.9103**	**0.4457**
Contact	GCN	0.8891	1.1747	0.4889	1.3626	1.7485	0.4343	2.3580	2.8872	0.3735
	GAT	0.8579	1.1269	0.4862	1.2642	1.6818	0.4285	1.9412	2.7057	0.4048
	GraphSAGE	0.8653	1.1392	0.5057	1.2461	1.6131	0.4657	1.6471	2.1265	0.4286
	SGFormer	0.8160	1.1041	0.4935	1.1923	1.5899	0.4422	1.6425	2.2004	0.4057
	NodeFormer	1.0261	1.3081	0.4596	1.7076	2.1143	0.4068	2.8412	3.3949	0.3751
	GraphGPS	0.8513	1.1307	0.4971	1.3215	1.6957	0.4505	2.0084	2.5280	0.4244
	TGN	1.0661	1.3535	0.4750	1.8515	2.3242	0.4089	3.5733	4.3327	0.3650
	TIDFormer	0.9457	1.2559	0.4828	1.6557	2.1509	0.4324	3.3123	4.2178	0.3936
	Graph-JEPA	0.9747	1.2581	0.4884	1.6361	2.1350	0.4388	3.0525	4.0853	0.4112
	MDGFM	0.9567	1.2312	0.4642	1.5444	1.9985	0.4085	2.8350	3.6336	0.3985
	WorldGraph	0.6756	0.9754	0.5412	0.8290	1.1253	0.4984	1.3388	1.6578	0.4919
	WorldGraph-Pretrain	**0.6540**	**0.9542**	**0.5693**	**0.7796**	**1.0800**	**0.5276**	**1.2564**	**1.5759**	**0.5046**
Enron	GCN	1.3042	1.9075	0.3605	1.5877	2.1316	0.3126	2.2529	2.8909	0.2909
	GAT	1.2605	1.9077	0.3580	1.3940	1.9723	0.3290	1.6845	2.2817	0.2990
	GraphSAGE	1.2994	1.9115	0.3718	1.5150	2.0678	0.3369	1.9448	2.5415	0.3059
	SGFormer	1.2876	1.8868	0.3794	1.5019	2.0407	0.3455	1.8557	2.4450	0.3211
	NodeFormer	1.2769	1.9222	0.3948	1.4932	2.0900	0.3688	1.9648	2.6402	0.3424
	GraphGPS	1.2499	1.8518	0.3937	1.3978	1.9342	0.3546	1.7089	2.2872	0.3152
	TGN	1.2283	1.8633	0.4151	1.3613	1.9201	0.3968	1.6981	2.2845	0.3809
	TIDFormer	1.2737	1.8897	0.3761	1.4000	1.9510	0.3558	1.7766	2.3845	0.3299
	Graph-JEPA	1.3478	1.9117	0.3394	1.6037	2.1119	0.3145	2.2833	2.9209	0.2981
	MDGFM	1.3762	2.0157	0.3419	1.6272	2.1908	0.3015	2.2030	2.8013	0.2610
	WorldGraph	1.2231	1.7935	0.4172	1.2508	1.7171	0.4108	1.3441	1.7830	0.3901
	WorldGraph-Pretrain	**1.2071**	**1.7874**	**0.4261**	**1.2332**	**1.7136**	**0.4113**	**1.3192**	**1.7820**	**0.3937**

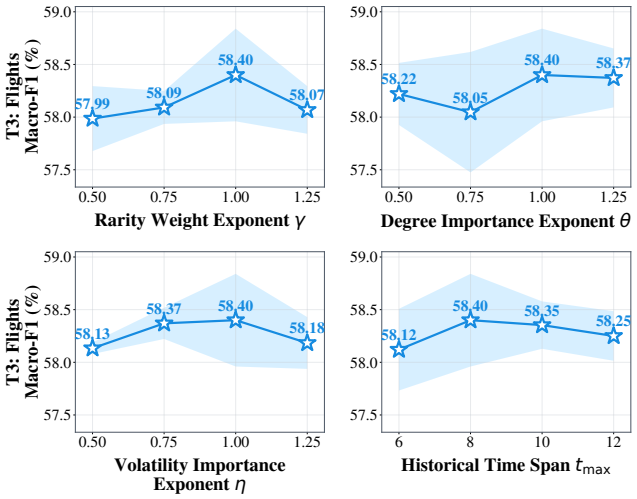


Figure 6: Sensitivity analysis on Graph-Level Tasks/Flights (Macro-F1) across the rarity weight exponent γ, degree importance exponent θ, volatility importance exponent η, and historical time span $t_{\max}$.

and 58.25% at $t_{\max} = 6$, 10, and 12, respectively. The intermediate settings of θ and η, together

with the default history span, provide a balance between emphasizing important nodes, preserving structural coverage, and retaining useful temporal context.

H.4 IMAGE-TO-GRAPH VISUAL ROLLOUT.

We further evaluate WorldGraph on a five-object Shapes transition-conditioned visual rollout task. Each input image is encoded by a CNN object extractor and an MLP into five object nodes forming a complete graph. Given the recorded environment transition, the multi-granularity graph encoder, history-aware state encoder, Controller, and transition module predict the next graph-state change; the Controller is trained with transition-aware dynamic group sampling and structure-aware rewards. A separately trained CNN decoder then maps the updated graph state back to an image. During multi-step prediction, only the initial image and the recorded transition sequence are provided, and the model recursively uses its predicted graph state rather than any ground-truth future image. As shown in Figure 7, the decoded WorldGraph rollout closely matches the ground-truth sequence, with the object positions and their temporal changes remaining well aligned across the prediction steps. This result demonstrates that WorldGraph can also support computer-vision tasks through graph-state prediction and reconstruction.

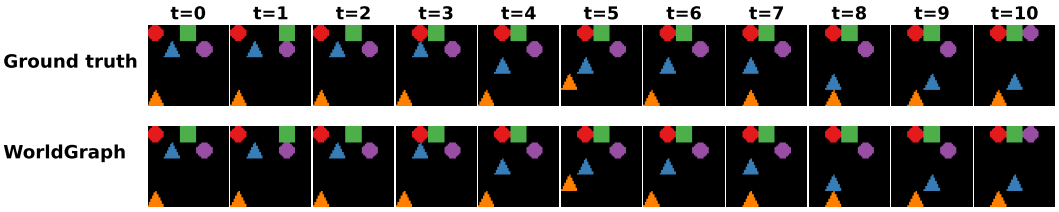


Figure 7: Transition-conditioned image-to-graph-to-image rollout. The first row shows the ground-truth sequence, and the second row shows the images decoded from the graph states predicted by WorldGraph.

I More Related Works

I.1 World Models.

World models learn compact internal representations of environments and model their temporal evolution to support prediction, planning, and decision-making. The seminal World Models framework (Ha & Schmidhuber, 2018) combines visual representation learning, recurrent memory, and a controller, while PlaNet (Hafner et al., 2019) learns stochastic latent dynamics for online planning and Dreamer (Hafner et al., 2020) learns long-horizon behaviors through latent imagination. Recent studies further improve the scalability and temporal abstraction of world models through decoder-free latent dynamics (Hansen et al., 2024), Transformer-based contrastive prediction (Burchi & Timofte, 2025), and autoregressive next-clip diffusion for visual prediction (Zhuang et al., 2026). Object-centric world models further decompose observations into explicit entities and model their stochastic dynamics (Daniel et al., 2026). However, although these representations improve prediction or make individual entities explicit, they generally leave evolving relations and structural dependencies implicit. Consequently, they struggle to model entity, relation, and attribute changes at different granularities, highlighting graph-structured representations and relational inductive biases as important foundations for modeling complex and evolving worlds (Liu et al., 2026).

I.2 Temporal Graph Learning.

Temporal graph learning models (Kumar et al., 2019; Rossi et al., 2020; Xu et al., 2020; Wang et al., 2021; Cong et al., 2023; Yu et al., 2023; Tian et al., 2024; Lu et al., 2024; Zou et al., 2024; Peng et al., 2025; Shi et al., 2026) time-varying interactions and connectivity, typically by encoding timestamped events or graph snapshots into temporal node representations. For example, TGN (Rossi et al., 2020) maintains node memories from timed events, TIDFormer (Peng et al., 2025) uses calendar-based temporal partitioning and interaction-level attention to capture temporal and interactive dynamics, and TGT (Shi et al., 2026) summarizes global evolutionary regularities with a temporal graph thumbnail. However, these methods primarily organize temporal information around interactions, sampled neighborhoods, or aggregate graph regularities and are mainly evaluated on node- or link-oriented prediction tasks. They are therefore not designed to explicitly represent coordinated changes to entities, relations, attributes, and local structures at multiple granularities. In contrast, WorldGraph models such structured graph changes as explicit transitions and jointly performs state modeling and node-, edge-, and graph-level transition prediction.

J Reproducibility and Code Availability

To ensure the reproducibility of our results, we provide the source code of WorldGraph at `https://anonymous.4open.science/r/WorldGraph-29C4`, and the benchmark datasets (GWM-Zero) in the *Supplementary Material*.

K Limitations and Broader Impacts

K.1 Limitations

In this work, we present an initial conceptualization of a graph-native world model. However, further work is still needed to better integrate information from diverse modalities, so as to more fully realize the potential of graph world models.

K.2 Broader Impacts

This work proposes a novel graph-native world modeling paradigm, with the potential to address existing limitations of conventional world models in learning structural information, thereby enabling more faithful modeling of relational dynamics in evolving graph worlds. Meanwhile, the core ideas of this work can be transferred to different modalities, enabling multimodal graph world modeling where diverse observations (e.g., images and text) are fused into an evolving graph representation.