TEMPORAL HYPERGRAPH JOINT-EMBEDDING PREDICTIVE ARCHITECTURE FOR MULTI-AGENT PATH FINDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-Agent Path Finding (MAPF) becomes difficult in dense environments where multiple agents must coordinate through shared aisles, intersections, and bottlenecks. Existing learned methods typically predict actions from the current interaction state, while pairwise representations can lose information about conflicts involving several agents. We introduce **TH-JEPA**, a Temporal Hypergraph Joint-Embedding Predictive Architecture that models MAPF as the evolution of higher-order agent–resource interactions. TH-JEPA first constructs a dynamic conflict hypergraph from anticipated shared-resource usage, with persistent resource nodes preserving conflict identity over time. Training proceeds in two stages: masked temporal pretraining learns the structure of conflict evolution without actions, followed by action-conditioned post-training that predicts future agent and resource representations over multiple horizons. To prevent collapse in the shared latent space, we introduce **Conflict-Spectral Dispersion Regularization (CSDR)**, which preserves variation over the conflict hypergraph while discouraging redundant latent directions. At inference, candidate joint-action sequences are rolled forward in latent space and evaluated using receding-horizon planning. We show that an unweighted pairwise co-membership projection cannot in general preserve higher-order incidence structure, and derive sufficient conditions for non-collapse within the CSDR-regularized structural subspace. Experiments on dense and out-of-distribution MAPF settings show improved scaling to larger agent populations, stronger performance under higher-cardinality conflicts, and accurate multi-horizon prediction.

1 INTRODUCTION

Multi-Agent Path Finding (MAPF) asks multiple agents to reach individual goals in a shared environment while avoiding vertex and edge-swap collisions Stern et al. (2019). It is a basic coordination problem in warehouse automation, transportation, and multi-robot systems, where many agents may need to traverse the same aisles, intersections, or bottlenecks. Classical MAPF solvers such as PIBT and LaCAM3 provide strong solutions Okumura et al. (2022); Okumura (2024), but repeated online search becomes increasingly costly as the number of agents and interactions grows. This has motivated learned approaches that replace part of the search process with policies trained from reinforcement learning or expert trajectories Li et al. (2021); Ma et al. (2022); Wang et al. (2023); Andreychuk et al. (2025).

Most learned MAPF policies reason from the current coordination state. Graph-based methods such as MAGAT communicate through pairwise agent interactions Li et al. (2021), while MAPF-GPT learns actions from large collections of expert trajectories Andreychuk et al. (2025). These approaches can be effective, but they do not explicitly model how alternative joint actions change the future interaction state. This matters in dense environments because agents that are currently separated may soon compete for the same constrained region. Hypergraph-based MAPF methods such as HMAGAT capture higher-order interactions among groups of agents Jain et al. (2026), but use this structure primarily for current-state action prediction. MAPF-World instead learns action-conditioned MAPF dynamics Yang et al. (2025), but does not explicitly organize the predicted state around persistent shared resources.

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.

We argue that these two aspects should be modeled together. A useful representation should identify agents that are expected to compete for the same physical resource, while a useful world model should predict how those relationships change under candidate actions. We therefore represent each MAPF state by a dynamic conflict hypergraph. Persistent resources such as intersections, narrow aisles, merge regions, and bottlenecks form the hyperedges, while their memberships change as agents move through the map. This preserves the identity of the same physical resource across time while allowing the interaction structure to evolve.

Based on this representation, we introduce **TH-JEPA**, a Temporal Hypergraph Joint-Embedding Predictive Architecture for MAPF. Training proceeds in two stages. Stage I performs masked temporal pretraining without actions. Agent and resource features are masked across a trajectory window, and a temporal predictor estimates their latent representations from the remaining temporal context and observed interaction structure. Stage II initializes from this representation and learns action-conditioned latent dynamics. Starting from the current agent and resource representations, the model recursively predicts how the coordination state changes under a sequence of joint actions. The future hypergraphs are used only to construct latent targets and are never provided to the prediction rollout. This separation follows the broader idea of first learning predictive structure from observation and then learning controllable dynamics, as explored in joint-embedding predictive models Assran et al. (2023; 2025).

Using the same encoder for context and targets makes collapse an important consideration. We introduce **Conflict-Spectral Dispersion Regularization (CSDR)**, which measures latent variation through the hypergraph Laplacian. CSDR regularizes a projected agent subspace by maintaining nonzero variation over the conflict structure while discouraging redundant structural directions. At inference time, TH-JEPA uses the Stage-II model as a latent world model. Candidate joint-action sequences are rolled forward, scored using predicted collision risk, congestion, waiting, and goal progress, and executed in a receding-horizon manner.

Our contributions are fourfold. **(i)** We introduce a dynamic resource-based conflict hypergraph that represents anticipated higher-order interactions while preserving resource identity across time. **(ii)** We develop a two-stage temporal JEPA for MAPF that combines action-free masked temporal pretraining with action-conditioned multi-horizon latent prediction. **(iii)** We introduce CSDR and establish sufficient conditions under which the regularized structural subspace cannot collapse into the hypergraph-Laplacian nullspace or a lower-dimensional structural representation. **(iv)** We use the learned dynamics for latent receding-horizon planning and evaluate the model under population shift, topology shift, higher-order conflicts, and multi-horizon prediction.

2 PRELIMINARIES

Hypergraphs and Hypergraph Neural Networks. A hypergraph is denoted by $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where each hyperedge $e \in \mathcal{E}$ may connect an arbitrary subset of nodes. Its structure is represented by an incidence matrix $B \in \{0,1\}^{N \times M}$, where $B_{ie} = 1$ if node i belongs to hyperedge e. Hypergraph neural networks propagate information through alternating node-to-hyperedge and hyperedge-to-node aggregation. Given node representations $Z \in \mathbb{R}^{N \times d}$ and hyperedge representations $U \in \mathbb{R}^{M \times d}$, a generic layer can be written as $U' = \Phi_E(Z, B)$ and $Z' = \Phi_V(U', B)$. This allows a shared interaction among multiple agents to be represented directly rather than decomposed into pairwise edges.

Joint-embedding prediction. Joint-Embedding Predictive Architectures learn by predicting representations of unobserved or future content rather than reconstructing the original observation Assran et al. (2023). Given an encoder E and context x, a predictor estimates the representation of a target x', $\widehat{z}' = P(E(x)), \qquad z' = E(x')$. The objective compares $\widehat{z}'$ and z' directly in latent space. Temporal versions extend this idea to sequences by predicting latent states across time. Action-conditioned variants additionally model how interventions change future representations, which makes latent prediction suitable for planning Assran et al. (2025). TH-JEPA applies these ideas to dynamic agent–resource hypergraphs rather than images or video frames.

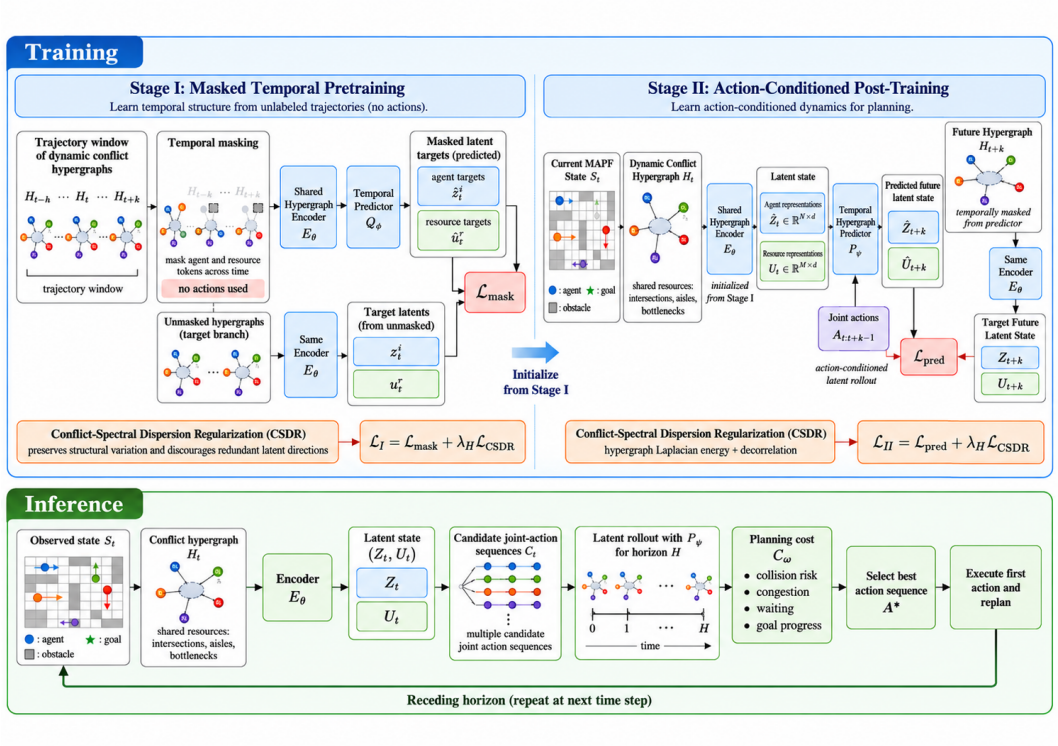

Figure 1: Overview of TH-JEPA

3 Related Work

Learning-based and predictive MAPF. Classical solvers such as PIBT and LaCAM3 explicitly search for collision-free paths Okumura et al. (2022); Okumura (2024), while learned methods amortize part of this computation through policies trained from interaction or expert demonstrations. MAGAT uses graph attention for decentralized communication Li et al. (2021), DCC and SCRIMP learn coordination through communication and reinforcement learning Ma et al. (2022); Wang et al. (2023), and MAPF-GPT scales imitation learning over expert trajectories Andreychuk et al. (2025). HMAGAT introduces hypergraph interactions for higher-order coordination Jain et al. (2026), while MAPF-World learns action-conditioned temporal dynamics Yang et al. (2025). TH-JEPA combines these directions by representing anticipated conflicts through persistent navigation resources and predicting how the resulting agent–resource state evolves under joint actions.

Joint-embedding predictive learning. I-JEPA introduced latent target prediction for images Assran et al. (2023), V-JEPA extended this idea to temporal visual representation learning Bardes et al. (2024), and V-JEPA 2 combined action-free predictive pretraining with action-conditioned world-model learning for planning Assran et al. (2025). LeJEPA studies shared-encoder joint-embedding learning with explicit regularization against collapse Balestriero & LeCun (2025). TH-JEPA adapts latent predictive learning to structured multi-agent coordination, where the latent state contains coupled agent and persistent resource representations.

4 Methodology

Problem Formulation: We consider multi-agent path finding (MAPF) on an environment graph $\mathcal{G}_{\text{env}} = (\mathcal{V}_{\text{env}}, \mathcal{E}_{\text{env}})$ with N agents $\mathcal{A} = \{1, \dots, N\}$. Agent i starts at $s_i \in \mathcal{V}_{\text{env}}$ and must reach goal $g_i \in \mathcal{V}_{\text{env}}$. At time t, its state is $x_i^t = [p_i^t, g_i, v_i^t, d_i^t, \ell_i^t]$, containing its current position, goal, recent motion, orientation, and local environment information. Each agent selects $a_i^t \in \mathcal{U}$ from the feasible movements and WAIT, forming the joint action $A_t = (a_1^t, \dots, a_N^t)$. We denote the global state by $S_t = \{x_i^t\}_{i=1}^N$. A valid solution requires all agents to reach their goals without vertex or

edge-swap collisions, i.e., $p_i^{t+1} \neq p_j^{t+1}$ and $(p_i^t, p_i^{t+1}) \neq (p_j^{t+1}, p_j^t)$ for all $i \neq j$. Rather than directly learning a policy $S_t \mapsto A_t$, our objective is to learn how candidate joint actions change the future coordination state of the system. To make this interaction structure explicit, we first transform each MAPF state into a dynamic conflict hypergraph. Figure 1 summarizes the two-stage training and inference pipeline.

4.1 Dynamic Conflict Hypergraph

Multi-agent conflicts often arise because several agents are expected to use the same constrained region. We therefore define a fixed set of persistent resources $\mathcal{R} = \{r_1, \ldots, r_M\}$ for each environment, including intersections, narrow aisles, merge regions, and bottlenecks. For each agent i, we compute a short nominal route $\tau_i^t = (p_i^t, \ldots, p_i^{t+H_c})$ toward g_i while temporarily ignoring other agents. The agents associated with resource r are

$$e_r^t = \left\{ i \in \mathcal{A} : \tau_i^t \cap r \neq \emptyset \right\}, \qquad (B_t)_{ir} = \mathbf{1}\left[i \in e_r^t \ \wedge \ |e_r^t| \geq 2 \right]. \tag{1}$$

Here $B_t \in \{0,1\}^{N \times M}$ is the dynamic incidence matrix. A resource becomes an active conflict hyperedge when at least two agents are expected to use it within the look-ahead horizon H_c; otherwise its incidence column is zero. We denote the resulting state by $\mathcal{H}_t = (\mathcal{A}, \mathcal{R}, B_t, X_t, R_t)$, where $X_t \in \mathbb{R}^{N \times d_x}$ contains agent features and $R_t \in \mathbb{R}^{M \times d_r}$ contains resource features. The resource set $\mathcal{R}$ remains fixed while the memberships encoded by B_t evolve with agent motion. This preserves resource identity across time and gives all hypergraphs a common $N \times M$ incidence structure, allowing the model to track how the same bottleneck or intersection changes from one coordination state to the next. We next encode $\mathcal{H}_t$ into coupled agent and resource representations that serve as the state of the learned temporal dynamics.

4.2 Hypergraph State Encoder

The encoder $(Z_t, U_t) = E_\theta(\mathcal{H}_t)$ maps the observed conflict hypergraph to agent representations $Z_t \in \mathbb{R}^{N \times d}$ and persistent-resource representations $U_t \in \mathbb{R}^{M \times d}$. We initialize $h_i^{(0)} = \phi_x(x_i^t)$, $u_r^{(0)} = \phi_r(r_r^t)$. Each encoder layer first aggregates the agents incident to a resource,

$$u_r^{(\ell+1)} = \phi_e\left(u_r^{(\ell)}, \sum_{i:(B_t)_{ir}=1} \alpha_{ir}^{(\ell)} W_v h_i^{(\ell)} \right), \tag{2}$$

where $\alpha_{ir}^{(\ell)}$ is obtained by a softmax over the agents incident to resource r. The updated resource states are then propagated back to the agents,

$$h_i^{(\ell+1)} = \phi_v\left(h_i^{(\ell)}, \sum_{r:(B_t)_{ir}=1} \beta_{ri}^{(\ell)} W_e u_r^{(\ell+1)} \right), \tag{3}$$

where $\beta_{ri}^{(\ell)}$ is normalized over the resources incident to agent i. After L layers, $Z_t = [h_1^{(L)}, \ldots, h_N^{(L)}]^\top$, $U_t = [u_1^{(L)}, \ldots, u_M^{(L)}]^\top$. The agent states encode local motion together with current shared-resource interactions, while the resource states summarize the coordination state of persistent map regions.

4.3 Temporal Hypergraph JEPA

TH-JEPA is trained in two stages. Stage I learns temporal regularities of agent and resource representations without using actions. Stage II then learns how joint actions change this latent coordination state. Both stages use the same hypergraph encoder E_θ for context and target representations.

Stage I: Masked temporal pretraining. We first learn temporal regularities from hypergraph trajectories alone. Given a temporal window $\mathcal{W}_t = \{\mathcal{H}_{t-h}, \ldots, \mathcal{H}_{t+k}\}$, we sample agent-token masks $\mathcal{M}_A$ and resource-token masks $\mathcal{M}_R$ across multiple time steps. For a masked agent token $(\tau, i) \in \mathcal{M}_A$, the corresponding agent feature is replaced by a learned mask embedding before

encoding. Similarly, a masked resource token $(\tau, r) \in \mathcal{M}_R$ is replaced by a learned resource-mask embedding.We retain the observed incidence matrix B_τ, so masking removes agent and resource attributes rather than the interaction scaffold itself. Stage I therefore learns temporal regularities of agent and resource representations conditioned on the observed conflict structure. Let $(\widetilde{Z}_\tau, \widetilde{U}_\tau) = E_\theta(\widetilde{\mathcal{H}}_\tau)$ denote the representations obtained from the masked hypergraph, and let $(Z_\tau, U_\tau) = E_\theta(\mathcal{H}_\tau)$ denote the corresponding targets obtained from the unmasked hypergraph. The temporal predictor Q_ϕ receives the masked-context representations together with temporal and token-position embeddings and predicts only the latent targets indexed by $\mathcal{M}_A$ and $\mathcal{M}_R$. The masked prediction objective is

$$\mathcal{L}_{\text{mask}} = \frac{1}{|\mathcal{M}_A|} \sum_{(\tau,i)\in\mathcal{M}_A} \|\widehat{z}_i^\tau - z_i^\tau\|_2^2 + \frac{\beta}{|\mathcal{M}_R|} \sum_{(\tau,r)\in\mathcal{M}_R} \|\widehat{u}_r^\tau - u_r^\tau\|_2^2. \tag{4}$$

Both branches use the same encoder parameters. Gradients propagate through the context and target encodings. With a shared context–target encoder, latent prediction alone can admit collapsed solutions. We therefore regularize the agent representations using the structure of the current conflict hypergraph.

Regularization. For incidence matrix B_τ, let $W_\tau = \operatorname{diag}(w_1^\tau, \ldots, w_M^\tau), \quad w_r^\tau \geq 0$, denote the resource-weight matrix. The hyperedge and agent degree matrices are defined by $(D_e^\tau)_{rr} = \sum_i (B_\tau)_{ir}, \quad (D_v^\tau)_{ii} = \sum_r w_r^\tau (B_\tau)_{ir}$. We use the unnormalized hypergraph Laplacian $L_\tau = D_v^\tau - B_\tau W_\tau (D_e^\tau)^\dagger B_\tau^\top$, where $(D_e^\tau)^\dagger$ handles persistent resources that are inactive at time τ. For nonnegative resource weights, L_τ is symmetric positive semidefinite and satisfies $L_\tau \mathbf{1} = 0$.

To measure structural variation in a compact latent subspace, let $R \in \mathbb{R}^{d\times q}$ have orthonormal columns, with $q \ll d$, and define $Y_\tau = Z_\tau R$. Each projected direction is normalized as $\bar{y}_j^\tau = \frac{y_j^\tau}{\|y_j^\tau\|_2+\epsilon}$, and we form $C_\tau = \bar{Y}_\tau^\top L_\tau \bar{Y}_\tau$. The diagonal entries of C_τ measure variation over the conflict structure, while the off-diagonal entries measure redundancy between structural directions. We define Conflict-Spectral Dispersion Regularization (CSDR) as

$$\mathcal{L}_{\text{CSDR}} = \frac{1}{q} \sum_{j=1}^{q} [\gamma - (C_\tau)_{jj}]_+^2 + \frac{\lambda_o}{q(q-1)} \sum_{j\neq l} (C_\tau)_{jl}^2. \tag{5}$$

The first term penalizes projected directions with insufficient variation over the current conflict structure. In particular, directions lying in the Laplacian nullspace have zero structural energy. The second term discourages different projected directions from encoding the same structural variation. We apply CSDR to sampled context and target representations across the temporal window. Stage I therefore minimizes $\mathcal{L}_{\text{I}} = \mathcal{L}_{\text{mask}} + \lambda_H \mathcal{L}_{\text{CSDR}}$.

Stage II: Action-conditioned post-training. We initialize E_θ from the Stage-I checkpoint and learn controllable latent dynamics from the same trajectories augmented with their executed joint actions. Starting from $(\widehat{Z}_t, \widehat{U}_t) = (Z_t, U_t)$, the predictor recursively applies

$$(\widehat{Z}_{t+j+1}, \widehat{U}_{t+j+1}) = P_\psi(\widehat{Z}_{t+j}, \widehat{U}_{t+j}, A_{t+j}). \tag{6}$$

Each action a_i^{t+j} is embedded and fused with the corresponding predicted agent state before updating the next agent and resource representations. The rollout does not observe the future hypergraph $\mathcal{H}_{t+j+1}$ or its incidence matrix B_{t+j+1}. Future agent–resource interactions must therefore be inferred from the current latent state and the intervening actions.

For each horizon $k \in \mathcal{K}$, the target is independently obtained as $(Z_{t+k}, U_{t+k}) = E_\theta(\mathcal{H}_{t+k})$. We minimize

$$\mathcal{L}_{\text{pred}} = \sum_{k\in\mathcal{K}} w_k \left[\frac{\|\widehat{Z}_{t+k} - Z_{t+k}\|_F^2}{N} + \frac{\beta \|\widehat{U}_{t+k} - U_{t+k}\|_F^2}{M} \right]. \tag{7}$$

The rollout is recursive, so ground-truth intermediate latent states are not inserted between prediction horizons. We retain the same structural regularizer during post-training, giving $\mathcal{L}_{\mathrm{II}} = \mathcal{L}_{\mathrm{pred}} + \lambda_H \mathcal{L}_{\mathrm{CSDR}}$. Stage I therefore learns the temporal organization of the conflict state, while Stage II learns how that state changes under joint actions.

4.4 Latent-Space Planning

At inference time, the Stage-II encoder and action-conditioned predictor form the latent world model. Given the observed state, we construct $\mathcal{H}_t$ and compute $(Z_t, U_t) = E_\theta(\mathcal{H}_t)$ once. Enumerating all joint actions is infeasible because the action space grows exponentially with the number of agents. We therefore use a bounded beam search to construct $\mathcal{C}_t$. Individually infeasible actions and partial joint assignments that produce immediate vertex or edge-swap conflicts are discarded before latent rollout. At each expansion step, only the highest-ranked partial assignments are retained until the candidate budget is reached. The beam width and candidate budget are reported in Appendix B.8. All surviving candidates share the same initial latent state and are evaluated in parallel.

For a candidate sequence $\mathbf{A} = (A_t, \ldots, A_{t+H-1}) \in \mathcal{C}_t$, we recursively apply Eq. equation 6 for H steps to obtain the predicted latent trajectory $(\widehat{Z}_{t:t+H}, \widehat{U}_{t:t+H})$. A lightweight readout C_ω predicts four trajectory-level quantities: collision risk $\widehat{C}_{\mathrm{col}}$, resource congestion $\widehat{C}_{\mathrm{cong}}$, waiting $\widehat{C}_{\mathrm{wait}}$, and goal progress $\widehat{C}_{\mathrm{goal}}$. These outputs are supervised using the corresponding quantities computed from training trajectory segments. The planner scores each candidate using

$$J_\omega(\mathbf{A}) = \lambda_{\mathrm{col}}\widehat{C}_{\mathrm{col}} + \lambda_{\mathrm{cong}}\widehat{C}_{\mathrm{cong}} + \lambda_{\mathrm{wait}}\widehat{C}_{\mathrm{wait}} - \lambda_{\mathrm{goal}}\widehat{C}_{\mathrm{goal}}. \tag{8}$$

We select $\mathbf{A}^\star = \arg\min_{\mathbf{A}\in\mathcal{C}_t} J_\omega(\mathbf{A})$ and execute only its first joint action $A_t^\star$. The next state is then observed, its conflict hypergraph is reconstructed, and planning is repeated.

4.5 Theoretical Properties

Theorem 1 (non-identifiability under unweighted pairwise projection) *Let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a hypergraph over $\mathcal{V} = \{1, \ldots, n\}$, and let $\Pi(\mathcal{H})$ denote its unweighted pairwise co-membership projection, where (i, j) is an edge whenever i and j belong to at least one common hyperedge, while hyperedge identity and multiplicity are discarded. Then Π is not injective. In particular, for every $n \geq 3$, there exist distinct hypergraphs $\mathcal{H}_1 \neq \mathcal{H}_2$ such that $\Pi(\mathcal{H}_1) = \Pi(\mathcal{H}_2)$. Consequently, any deterministic graph encoder whose input depends on $\mathcal{H}$ only through $\Pi(\mathcal{H})$ cannot distinguish these higher-order structures. In contrast, an incidence-aware permutation-equivariant hypergraph map can separate them. (proof in Appendix A)*

Theorem 1 characterizes the information lost by the unweighted projection used in our Pairwise-JEPA ablation. Distinct higher-order incidence structures can induce the same projected graph, so this information cannot be recovered from that projection alone. The result does not rule out pairwise models augmented with additional higher-order edge attributes. TH-JEPA instead retains the incidence structure explicitly through B_t.

Theorem 2 (structural non-collapse under CSDR) *Let $L_{\mathcal{H}} \in \mathbb{R}^{N\times N}$ be a symmetric positive semidefinite hypergraph Laplacian and let $\bar{Y} = [\bar{\mathbf{y}}_1, \ldots, \bar{\mathbf{y}}_q] \in \mathbb{R}^{N\times q}$ contain the normalized projected latent directions regularized by CSDR. Define $C = \bar{Y}^\top L_{\mathcal{H}} \bar{Y}$ and $C_{\mathrm{off}} = C - \mathrm{diag}(\mathrm{diag}(C))$. Assume that $C_{jj} \geq \gamma > 0$ for every j and that $\|C_{\mathrm{off}}\|_F \leq \eta$, where $\eta < \gamma/\sqrt{q-1}$ for $q \geq 2$. Then $\lambda_{\min}(C) \geq \gamma - \sqrt{q-1}\eta > 0$. Consequently, C is positive definite, $\mathrm{rank}(L_{\mathcal{H}}^{1/2}\bar{Y}) = q$, and no projected direction $\bar{\mathbf{y}}_j$ lies in $\ker(L_{\mathcal{H}})$. Thus CSDR excludes collapse into the hypergraph-Laplacian nullspace and dimensional collapse within the regularized q-dimensional structural subspace. (proof in Appendix A.1)*

Theorem 2 applies to every context or target representation on which CSDR is imposed in either training stage. The diagonal term enforces nonzero structural energy, while the off-diagonal term prevents the projected directions from becoming structurally redundant. For a connected conflict hypergraph, $\ker(L_{\mathcal{H}}) = \mathrm{span}\{\mathbf{1}\}$, so constant collapse is excluded directly. For a disconnected

Table 1: Zero-shot scaling on small-map environments. Success rate (%, mean ± std.) is reported at $N = 64/90/128$. TH-JEPA is trained only with $N \in \{16, 24, 32\}$.

	Random Obstacles			Dense Maze			Dense Room			Dense Warehouse		
Method	64	90	128	64	90	128	64	90	128	64	90	128
MAGAT	94.5±0.8	86.7±1.3	78.1±1.9	88.3±1.2	80.5±1.7	72.7±2.1	84.4±1.4	75.0±2.0	66.4±2.5	41.4±2.7	14.8±2.2	2.3±0.9
HMAGAT	97.7±0.5	94.5±0.8	90.6±1.1	93.0±0.9	84.4±1.5	73.4±1.9	89.1±1.1	75.8±1.8	60.9±2.4	98.4±0.4	94.5±0.9	89.1±1.3
MAPF-GPT-85M	98.4±0.4	95.3±0.7	92.2±1.0	91.4±1.0	82.8±1.5	70.3±2.0	92.2±0.9	82.0±1.6	70.3±2.1	83.6±1.5	41.4±2.8	9.4±1.7
MAPF-World	99.2±0.3	97.7±0.5	94.5±0.8	95.3±0.7	90.6±1.1	82.8±1.6	94.5±0.8	88.3±1.2	78.9±1.8	96.9±0.6	91.4±1.0	84.4±1.5
TH-JEPA	**99.2±0.2**	**98.4±0.3**	**96.1±0.6**	**98.4±0.3**	**94.5±0.7**	**88.3±1.1**	**97.7±0.4**	**92.2±0.8**	**84.4±1.3**	**99.2±0.2**	**97.7±0.4**	**94.5±0.7**

hypergraph, the nullspace contains signals that are constant on each connected component, and the same result excludes collapse entirely into this component-wise constant subspace.

5 Experiments

Data and training. We generate expert trajectories in POGEMA Skrynnik et al. (2025) using LaCAM3 Okumura (2024). Following the matched-data setting of HMAGAT Jain et al. (2026), the training set contains 21K instances: 20% random-obstacle layouts and 80% maze layouts, with map sizes 17×17–21×21 and $N \in \{16, 24, 32\}$ agents. We retain complete trajectories $(S_0, A_0, \ldots, S_T)$. Stage I uses these trajectories for masked temporal pretraining without actions, while Stage II uses the same data with joint actions for multi-horizon prediction at $\mathcal{K} = \{1, 2, 4, 8\}$. Room and Warehouse layouts are not used during training and are reserved for topology-shift evaluation. Full optimization, masking, and implementation details are given in Appendix B.

Baselines and metrics. Our primary learned baselines are MAGAT Li et al. (2021), HMAGAT Jain et al. (2026), MAPF-GPT-85M Andreychuk et al. (2025), and MAPF-World Yang et al. (2025). PIBT Okumura et al. (2022) and LaCAM3 Okumura (2024) are included as classical solver references in the supplementary evaluation. All methods are evaluated on the same maps, start-goal assignments, episode limits, and random seeds for each reported comparison. We additionally report relative Sum of Costs (Rel. SoC), makespan, blocking events, prediction quality, and inference latency. TH-JEPA uses latent dimension $d = 128$, three hypergraph encoder layers, three predictor layers, and AdamW with learning rate 3×10^{-4}. Additional settings are reported in Appendix B.

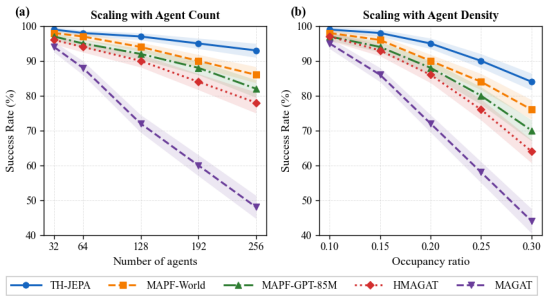

Figure 2: Scaling with agent count and density. Shaded regions denote 95% confidence intervals.

5.1 MAPF Performance and Qualitative Analysis

Table 1 evaluates zero-shot generalization beyond the training range of at most 32 agents. TH-JEPA remains competitive on Random Obstacles and shows larger gains as the environments become more structured. At $N = 128$, its success rates on Dense Maze, Dense Room, and Dense Warehouse are 88.3%, 84.4%, and 94.5%, respectively. The largest separation is observed in Dense Warehouse, where agents repeatedly compete for shared aisles and bottlenecks. Figure 2 further examines population size and occupancy. Increasing either quantity also increases the frequency and cardinality of shared-resource conflicts. TH-JEPA degrades more gradually under both shifts. Results on larger MovingAI maps and additional density settings are provided in Appendix E.

Table 2: Action-conditioned latent prediction on held-out trajectories.

k	Node MSE ↓	Res. MSE ↓	Cos. ↑	Conflict F1 ↑
1	0.018	0.014	0.982	0.956
2	0.026	0.021	0.968	0.941
4	0.041	0.034	0.944	0.918
8	0.067	0.056	0.911	0.881

Predictive Dynamics. Table 2 evaluates the Stage-II predictor independently of the planner. Starting from (Z_t, U_t), we recursively roll out P_ψ using the ground-truth joint actions and compare its

Method	Dense Maze			Dense Room			Dense Warehouse		
	MS ↓	B ↓	W ↓	MS ↓	B ↓	W ↓	MS ↓	B ↓	W ↓
PIBT	55.0±3.2	22.0±2.1	71.0±5.6	61.0±3.8	27.0±2.5	84.0±6.4	52.0±3.0	21.0±2.0	69.0±5.1
MAGAT	51.0±2.9	20.0±1.9	66.0±5.0	57.0±3.4	24.0±2.3	77.0±5.8	49.0±2.8	19.0±1.8	63.0±4.7
HMAGAT	47.0±2.6	18.0±1.7	58.0±4.4	52.0±3.0	21.0±2.0	68.0±5.0	47.0±2.5	18.0±1.6	58.0±4.2
MAPF-GPT-85M	41.0±2.3	13.0±1.5	43.0±3.8	46.0±2.7	16.0±1.7	54.0±4.3	40.0±2.2	12.0±1.4	41.0±3.5
MAPF-World	36.0±2.0	9.0±1.2	29.0±2.9	40.0±2.2	12.0±1.4	38.0±3.3	36.0±1.9	9.0±1.1	27.0±2.7
TH-JEPA	**30.0±1.6**	**4.0±0.8**	**15.0±2.0**	**33.0±1.8**	**6.0±0.9**	**20.0±2.3**	**28.0±1.5**	**3.0±0.7**	**12.0±1.8**

Table 3: Coordination statistics on held-out $N = 128$ instances, reported as mean ± standard deviation across five evaluation seeds. MS denotes makespan, B the number of blocking events, and W cumulative agent waiting steps. All methods use identical maps, start–goal assignments, episode limits, and evaluation seeds.

predictions with independently encoded future hypergraphs. Future incidence matrices are never given to the predictor. We report Node MSE and Resource MSE for agent and resource latents, cosine similarity between predicted and target agent representations, and Conflict F1, which measures how accurately a fixed probe recovers future active agent-resource relations. Prediction quality decreases gradually with rollout length. At the longest planning horizon used by TH-JEPA, $k = 8$, cosine similarity remains 0.911 and Conflict F1 is 0.881.

We also test whether the predictor actually uses the conditioning actions. Shuffling the action sequences while preserving the initial state increases Node MSE at $k = 8$ from 0.067 to 0.184 and reduces Conflict F1 from 0.881 to 0.641. On counterfactual action pairs, TH-JEPA ranks the lower-cost future correctly in 87.6% of cases, compared with 51.2% under shuffled actions. The degradation under action shuffling indicates that accurate prediction depends substantially on the intervening joint actions.

Coordination under shared-resource contention.

We further examine how the methods coordinate when several agents compete for constrained regions. We use the same held-out $N = 128$ instances from Dense Maze, Dense Room, and Dense Warehouse as in the main evaluation. For each method, we report the mean over evaluation episodes for five random seeds, with the standard deviation computed across seed-level means. Makespan (MS) measures the time until all successful agents complete the episode, blocking events (B) count persistent movement conflicts, and waiting (W) is the cumulative number of agent wait steps.

Table 3 shows that TH-JEPA reduces contention consistently across all three environments. The difference is most pronounced in Dense Warehouse, where repeated use of shared aisles and intersections creates persistent multi-agent conflicts. Relative to MAPF-World, TH-JEPA reduces the average makespan from 36.0 to 28.0 steps, blocking events from 9.0 to 3.0, and cumulative waiting from 27.0 to 12.0 steps. Similar reductions appear in Dense Maze and Dense Room. These results indicate that the improvement is not limited to a higher episode success rate. TH-JEPA also spends less time in blocked or waiting configurations, consistent with using predicted resource-level interactions to coordinate access to constrained regions before conflicts occur.

Higher-order conflicts. We group evaluation states by the size of the largest active conflict hyperedge, defined as the maximum number of agents simultaneously associated with the same shared resource.

Figure 3 shows little difference between Pairwise-JEPA and TH-JEPA when conflicts are predominantly pairwise. As the conflict cardinality increases, however, the performance gap widens: TH-JEPA improves success rate by 3.1, 8.8, and 16.5 percentage points for cardinality groups 3–4, 5–6, and ≥ 7, respectively. This trend suggests that the advantage of TH-JEPA is concentrated in settings with genuine higher-order contention, where several agents compete for the same resource simultaneously. Pairwise-JEPA models these interactions through separate agent pairs,

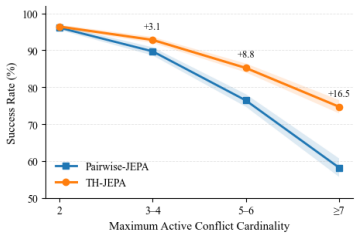

Figure 3: Success rate versus maximum active conflict cardinality.

whereas TH-JEPA represents the complete resource-level interaction in a single hyperedge. The observed gap is therefore consistent with Theorem 1, which shows that certain higher-order conflict structures cannot be uniquely recovered from pairwise relations alone.

5.2 Ablation Study

We evaluate the main design choices on Dense Warehouse using the same data, latent dimension, and optimization budget. *Direct Hypergraph Policy* replaces latent prediction and planning with an action head on the pretrained hypergraph encoder. *Pairwise-JEPA* replaces resource hyperedges with pairwise interactions, while *Random-HG* preserves hyperedge counts and cardinalities but randomizes membership. *w/o Stage-I* removes masked pretraining, $k = 1$ restricts Stage II to one-step prediction, and *w/o CSDR* removes the structural regularizer. The direct policy reaches 85.2% SR, compared with 95.3% for the full model, indicating that the hypergraph encoder alone does not account for the gain. Replacing resource hyperedges with pairwise or randomized interactions reduces SR to 88.3% and 87.5%, respectively. Removing Stage-I pretraining reduces SR from 95.3% to 92.2% and future-conflict F1 from 0.881 to 0.857. Restricting Stage II to one-step prediction reduces SR further to 90.6%, indicating that both temporal pretraining and multi-horizon supervision contribute to downstream planning. Removing CSDR has the largest effect on representation rank, reducing it from 31.8 to 12.7.

Table 4: Ablation on Dense Warehouse.

Variant	SR ↑	Rel. SoC ↓	F1 ↑	Eff. Rank ↑
Direct Hypergraph Policy	85.2	1.31	–	27.5
Pairwise-JEPA	88.3	1.25	0.824	28.6
Random-HG	87.5	1.28	0.803	26.9
TH-JEPA, $k = 1$	90.6	1.20	0.838	30.1
TH-JEPA, w/o Stage-I	92.2	1.17	0.857	27.9
TH-JEPA, w/o CSDR	91.4	1.18	0.868	12.7
Full TH-JEPA	**95.3**	**1.11**	**0.881**	**31.8**

Learned vs. oracle dynamics. To isolate world-model error, we replace P_ψ during candidate rollouts with ground-truth simulator transitions. For each candidate action, the resulting simulator state is re-encoded with the same encoder E_θ; the candidate generator, planning horizon, cost function, and receding-horizon decision rule remain unchanged. Replacing learned rollouts with exact transitions improves SR from 95.3% to 97.7% and further reduces SoC, makespan, blocking, and waiting. The relatively small learned-to-oracle gap indicates that TH-JEPA captures most of the dynamics needed for planning, while accumulated rollout error remains one source of residual failure.

Table 5: Learned versus oracle dynamics on Dense Warehouse. B: blocking events, W: cumulative waiting, GP: goal progress.

Method	SR ↑	SoC ↓	MS ↓	B ↓	W ↓	GP ↑
Direct Policy	85.2	1.31	47	18	58	0.84
TH-JEPA ($H = 1$)	90.6	1.20	39	12	39	0.90
TH-JEPA	95.3	1.11	28	3	12	0.96
Oracle ($H = 1$)	91.8	1.17	36	10	32	0.92
Oracle MPC	**97.7**	**1.06**	**25**	**2**	**8**	**0.98**

Limitations TH-JEPA relies on a predefined set of persistent navigation resources and short nominal routes to construct conflict hyperedges. Our current evaluation is limited to grid-based MAPF, leaving continuous and partially observed multi-robot settings for future work.

6 Conclusion

We introduced TH-JEPA, a temporal hypergraph world model for multi-agent path finding. Persistent navigation resources provide a higher-order representation of anticipated shared conflicts. Masked temporal pretraining learns regularities of agent and resource states, after which action-conditioned post-training learns their controllable latent dynamics. CSDR regularizes the structural dispersion of the agent representation, and the resulting world model supports receding-horizon planning in latent space. The largest empirical gains occur under larger populations and higher-cardinality conflicts. Predictive evaluations further show that the learned future representations depend on the conditioning actions and remain accurate over the horizons used for planning. These results support modeling not only the current coordination state but also how higher-order agent–resource interactions evolve under candidate actions.

Reproducibility Statement. We provide the data-generation procedure, model architecture, training configuration, masking strategy, optimization settings, planning parameters, baseline configurations, and hardware details in the appendix. All reported comparisons use matched test instances, episode limits, and evaluation seeds.

Impact Statement. TH-JEPA is intended to improve coordination in applications such as warehouse automation and multi-robot systems. However, our evaluation is simulation-based and should not be interpreted as evidence of safety in physical deployments. Real-world use would require additional validation under sensing uncertainty, dynamic obstacles, and system failures.

AI Assistance Statement. Generative AI tools were used to assist with language editing, grammar, and presentation.

REFERENCES

Anton Andreychuk, Konstantin Yakovlev, Aleksandr Panov, and Alexey Skrynnik. Mapf-gpt: Imitation learning for multi-agent pathfinding at scale. 2025. URL https://arxiv.org/abs/2409.00134.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. 2023. URL https://arxiv.org/abs/2301.08243.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. 2025.

Randall Balestriero and Yann LeCun. Lejepa: Provable and scalable self-supervised learning without the heuristics, 2025. URL https://arxiv.org/abs/2511.08544.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. Revisiting Feature Prediction for Learning Visual Representations from Video. *Transactions on Machine Learning Research*, 2024. URL https://mlanthology.org/tmlr/2024/bardes2024tmlr-revisiting/.

Rishabh Jain, Keisuke Okumura, Michael Amir, Pietro Lio, and Amanda Prorok. Pairwise is not enough: Hypergraph neural networks for multi-agent pathfinding. 2026. URL https://arxiv.org/abs/2602.06733.

Qingbiao Li, Weizhe Lin, Zhe Liu, and Amanda Prorok. Message-aware graph attention networks for large-scale multi-robot path planning. *IEEE Robotics and Automation Letters*, 6(3):5533–5540, 2021. doi: 10.1109/LRA.2021.3077863.

Ziyuan Ma, Yudong Luo, and Jia Pan. Learning selective communication for multi-agent path finding. *IEEE Robotics and Automation Letters*, 7(2):1455–1462, 2022. doi: 10.1109/LRA.2021.3139145.

Keisuke Okumura. Engineering lacam*: Towards real-time, large-scale, and near-optimal multi-agent pathfinding. 2024. URL https://arxiv.org/abs/2308.04292.

Keisuke Okumura, Manao Machida, Xavier Défago, and Yasumasa Tamura. Priority inheritance with backtracking for iterative multi-agent path finding. *Artificial Intelligence*, pp. 103752, 2022. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2022.103752.

Alexey Skrynnik, Anton Andreychuk, Anatolii Borzilov, Alexander Chernyavskiy, Konstantin Yakovlev, and Aleksandr Panov. Pogema: A benchmark platform for cooperative multi-agent pathfinding. 2025. URL https://arxiv.org/abs/2407.14931.

Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Roman Barták, and Eli Boyarski. Multi-agent pathfinding: Definitions, variants, and benchmarks. *Proceedings of the International Symposium on Combinatorial Search*, 10(1):151–158, 2019. doi: 10.1609/socs.v10i1.18510.

Yutong Wang, Bairan Xiang, Shinan Huang, and Guillaume Sartoretti. Scrimp: Scalable communication for reinforcement- and imitation-learning-based multi-agent pathfinding. 2023. URL https://arxiv.org/abs/2303.00605.

Zhanjiang Yang, Yang Shen, Yueming Li, Meng Li, and Lijun Sun. Mapf-world: Action world model for multi-agent path finding. 2025. URL https://arxiv.org/abs/2508.12087.

A Proof of Theorem 1

Let $\mathcal{V} = \{1, \dots, n\}$ with $n \geq 3$, and define

$$\mathcal{H}_1 = (\mathcal{V}, \{\mathcal{V}\}), \qquad \mathcal{H}_2 = (\mathcal{V}, \{\{i, j\} : 1 \leq i < j \leq n\}). \tag{9}$$

Clearly, $\mathcal{H}_1 \neq \mathcal{H}_2$, since $\mathcal{H}_1$ contains a single hyperedge of cardinality n, whereas $\mathcal{H}_2$ contains $\binom{n}{2}$ hyperedges of cardinality two. **Equality of the pairwise projections.** For any distinct $i, j \in \mathcal{V}$, the unique hyperedge $\mathcal{V}$ in $\mathcal{H}_1$ contains both i and j. Therefore $(i, j) \in \Pi(\mathcal{H}_1)$. Similarly, by construction, $\{i, j\} \in \mathcal{E}_2$ for every distinct pair i, j, so $(i, j) \in \Pi(\mathcal{H}_2)$. Hence both projected graphs contain every possible pairwise edge and

$$\Pi(\mathcal{H}_1) = \Pi(\mathcal{H}_2) = K_n. \tag{10}$$

Therefore the map Π is not injective. **Indistinguishability of pairwise graph encoders.** Let F be any deterministic graph encoder operating on the projected graph and node features $X \in \mathbb{R}^{n \times d_x}$. Since Eq. equation 10 gives identical pairwise structures and the node features are unchanged,

$$F(\Pi(\mathcal{H}_1), X) = F(\Pi(\mathcal{H}_2), X). \tag{11}$$

Thus the ambiguity is introduced before message passing is applied. Increasing the depth or expressive capacity of F cannot recover the higher-order incidence information removed by Π. **Separation by the incidence representation.** Let $B_k \in \{0, 1\}^{n \times m_k}$ denote the incidence matrix of $\mathcal{H}_k$, and define the incidence-aware map

$$\Psi(B) = BB^\top \mathbf{1}_n. \tag{12}$$

For $\mathcal{H}_1$, the incidence matrix is $B_1 = \mathbf{1}_n$, since all nodes belong to the single hyperedge. Therefore

$$\Psi(B_1) = B_1 B_1^\top \mathbf{1}_n = n\mathbf{1}_n. \tag{13}$$

For $\mathcal{H}_2$, every node belongs to exactly $n - 1$ pairwise hyperedges, and every such hyperedge has cardinality two. Thus

$$\Psi(B_2) = B_2 B_2^\top \mathbf{1}_n = 2(n-1)\mathbf{1}_n. \tag{14}$$

Since $n \geq 3$,

$$n \neq 2(n-1), \tag{15}$$

which implies $\Psi(B_1) \neq \Psi(B_2)$. **Permutation equivariance.** Let P be any permutation matrix acting on the node ordering. The incidence matrix transforms as $B \mapsto PB$. Using $P^\top \mathbf{1}_n = \mathbf{1}_n$,

$$\Psi(PB) = PBB^\top P^\top \mathbf{1}_n = PBB^\top \mathbf{1}_n = P\Psi(B). \tag{16}$$

Therefore Ψ is permutation equivariant and can be realized by a node–hyperedge–node message-passing architecture with sum aggregation. Hence distinct higher-order conflict structures can collapse to the same pairwise graph under Π, while the incidence representation preserves sufficient information to distinguish them. This proves the claim.

A.1 Proof of Theorem 2

Since $L_{\mathcal{H}}$ is symmetric positive semidefinite, it admits the factorization $L_{\mathcal{H}} = L_{\mathcal{H}}^{1/2} L_{\mathcal{H}}^{1/2}$. Therefore

$$C = \bar{Y}^\top L_{\mathcal{H}} \bar{Y} = \left(L_{\mathcal{H}}^{1/2} \bar{Y}\right)^\top \left(L_{\mathcal{H}}^{1/2} \bar{Y}\right), \tag{17}$$

so C is the Gram matrix of the structurally transformed projected directions.

Positive structural energy. For every projected direction $\bar{\mathbf{y}}_j$,

$$C_{jj} = \bar{\mathbf{y}}_j^\top L_{\mathcal{H}} \bar{\mathbf{y}}_j = \left\| L_{\mathcal{H}}^{1/2} \bar{\mathbf{y}}_j \right\|_2^2 \geq \gamma > 0. \tag{18}$$

If $\bar{\mathbf{y}}_j \in \ker(L_{\mathcal{H}})$, then $L_{\mathcal{H}}^{1/2} \bar{\mathbf{y}}_j = \mathbf{0}$ and Eq. equation 18 would give $C_{jj} = 0$, contradicting $C_{jj} \geq \gamma$. Hence

$$\bar{\mathbf{y}}_j \notin \ker(L_{\mathcal{H}}) \qquad \text{for all } j. \tag{19}$$

Lower bound on the smallest eigenvalue. For row j of C, define the off-diagonal absolute row sum $r_j = \sum_{k \neq j} |C_{jk}|$. By the Cauchy–Schwarz inequality,

$$r_j \leq \sqrt{q-1} \left(\sum_{k \neq j} C_{jk}^2 \right)^{1/2} \leq \sqrt{q-1}\, \|C_{\text{off}}\|_F \leq \sqrt{q-1}\eta. \tag{20}$$

Since C is symmetric, all of its eigenvalues are real. By the Gershgorin circle theorem, every eigenvalue λ of C satisfies $\lambda \geq C_{jj} - r_j$ for at least one row j. Therefore

$$\lambda_{\min}(C) \geq \min_j (C_{jj} - r_j) \geq \gamma - \sqrt{q-1}\eta. \tag{21}$$

By assumption, $\eta < \gamma / \sqrt{q-1}$, and hence

$$\lambda_{\min}(C) > 0. \tag{22}$$

Thus C is positive definite.

Preservation of structural rank. Let $M = L_{\mathcal{H}}^{1/2} \bar{Y}$. From Eq. equation 17, $C = M^\top M$. Since C is positive definite,

$$q = \operatorname{rank}(C) = \operatorname{rank}(M^\top M) = \operatorname{rank}(M). \tag{23}$$

Therefore

$$\operatorname{rank}\left(L_{\mathcal{H}}^{1/2} \bar{Y} \right) = q. \tag{24}$$

Hence the q projected directions remain linearly independent after removing the nullspace of $L_{\mathcal{H}}$. In particular, they cannot collapse onto fewer than q structurally independent directions.

Relation to the CSDR objective. The CSDR regularizer is $\mathcal{L}_{\text{CSDR}} = \frac{1}{q} \sum_{j=1}^{q} [\gamma - C_{jj}]_+^2 + \frac{\lambda_o}{q(q-1)} \sum_{j \neq k} C_{jk}^2$. Vanishing diagonal penalty implies $C_{jj} \geq \gamma$ for every j, while the second term directly controls $\|C_{\text{off}}\|_F^2$. Therefore, whenever the off-diagonal component satisfies

$$\|C_{\text{off}}\|_F < \frac{\gamma}{\sqrt{q-1}}, \tag{25}$$

the conditions of Theorem 2 hold. Thus CSDR prevents the regularized projected representations from collapsing into the hypergraph Laplacian nullspace or into a lower-dimensional structural subspace. This proves the claim.

CSDR and representation collapse. We compare CSDR with no regularization, feature variance–covariance regularization, and SIGReg Balestriero & LeCun (2025). CSDR yields the highest effective rank and structural energy and the lowest pairwise cosine similarity (Table 6), consistent with Theorem 2.

Additional out-of-distribution results, masking analyses, planner sensitivity, and computational-efficiency results are reported in Appendices E, F, G, and H.

Table 6: Representation-collapse analysis.

Regularizer	Eff. Rank ↑	Pair. Cos. ↓	Struct. Energy ↑	SR ↑
None	12.7	0.91	0.011	91.4
Variance–Cov.	26.4	0.58	0.029	92.2
SIGReg Balestriero & LeCun (2025)	29.8	0.51	0.033	93.0
CSDR	**31.8**	**0.43**	**0.061**	**95.3**

B Implementation Details

B.1 Dataset Generation

We generate expert trajectories in POGEMA Skrynnik et al. (2025) using LaCAM3 Okumura (2024). Following the matched-data setting used in the main paper, the training set contains 21K instances. Random-obstacle layouts account for 20% of the data and maze layouts account for the remaining 80%. Training maps range from 17×17 to 21×21, with $N \in \{16, 24, 32\}$ agents.

For each instance, we retain the complete expert trajectory

$$(S_0, A_0, S_1, A_1, \ldots, S_T),$$

including the static map, agent positions and goals, executed joint actions, and terminal state. We split the data at the problem-instance level so that trajectories generated from the same start–goal configuration never appear in more than one split. We use 80% of the instances for training, 10% for validation, and 10% for held-out in-distribution evaluation.

Room and Warehouse layouts are excluded from training and are used only for topology-shift evaluation. The learned models are trained with at most 32 agents and evaluated zero-shot with 64, 90, and 128 agents on the small-map suite. Larger population shifts are considered separately in Appendix E.

Trajectory preprocessing. At every time step, we retain the static occupancy map together with the dynamic state of each agent. Agent features include current position, goal offset, recent displacement, motion direction, and local obstacle information. All continuous features are normalized using statistics computed from the training split. Discrete actions are represented by five values corresponding to Wait, Up, Down, Left, and Right.

The persistent resource set is extracted once for each map and reused throughout the trajectory. Only the agent–resource memberships change over time.

B.2 Persistent Resource Extraction

We first convert each grid map into a four-neighbor free-space graph. Persistent navigation resources are then extracted using only the map geometry. We identify four types of structures: intersections, narrow aisles, merge regions, and bottlenecks.

A free cell with at least three traversable neighbors is treated as an intersection candidate. Maximal degree-two chains are treated as corridor segments, and segments with local free-space width of at most two cells are marked as narrow aisles. Merge regions are formed around transitions from a narrow corridor to a wider region. Bottleneck candidates are narrow corridor segments whose removal locally separates otherwise connected free-space regions. Adjacent candidates of the same type are merged into a single resource, and overlapping resource regions are merged to avoid duplicate hyperedges.

This preprocessing produces a fixed resource set $\mathcal{R} = \{r_1, \ldots, r_M\}$ for every map. Each resource retains the same index throughout an episode, which is important for predicting the temporal evolution of its latent state.

At state S_t, we compute a short nominal route for each agent using A* with Manhattan distance while temporarily ignoring other agents. We use a conflict look-ahead horizon of

$$H_c = 8.$$

Only the first H_c steps of the nominal route are used when constructing the conflict hypergraph. Agent i is associated with resource r if this route intersects the corresponding resource region. A resource becomes an active hyperedge only when at least two agents are associated with it:

$$(B_t)_{ir} = \mathbf{1}\left[\tau_i^t \cap r \neq \emptyset \ \wedge \ |e_r^t| \geq 2\right].$$

Inactive resources remain in the representation with zero incidence. Shortest path ties are broken deterministically so that resource memberships do not change because of arbitrary routing decisions.

B.3 Hypergraph Encoder

The shared encoder E_θ contains three alternating agent-to-resource and resource-to-agent message-passing layers. Agent and resource features are first projected to the common latent dimension

$$d = 128.$$

Each message-passing layer uses four attention heads and a hidden dimension of 128. The feed-forward block uses hidden dimension 256, GELU activations, residual connections, and dropout 0.1.

The encoder returns

$$Z_t \in \mathbb{R}^{N\times 128}, \qquad U_t \in \mathbb{R}^{M\times 128}.$$

The same encoder parameters are used for both context and target representations. We do not maintain a separate EMA target encoder. Context and target representations are produced by the same encoder, while CSDR regularizes the structural dispersion of the projected agent representation.

Resource embeddings are preserved even when a resource is inactive at a particular time step. This allows the predictor to track the same aisle, intersection, or bottleneck across the complete trajectory.

B.4 Stage I: Masked Temporal Pretraining

Stage I learns temporal regularities of the dynamic conflict hypergraph without using actions. We sample windows containing nine consecutive hypergraph states,

$$\{\mathcal{H}_{t-4}, \ldots, \mathcal{H}_t, \ldots, \mathcal{H}_{t+4}\}.$$

Thus, $h = k = 4$ during masked temporal pretraining.

For every window, we independently mask 40% of the agent tokens and 40% of the resource tokens. Masks are sampled across multiple time steps rather than only at the final state.This requires the model to recover masked agent and resource representations from temporal context while conditioning on the observed conflict structure.

The visible tokens are processed by the prediction branch, while the corresponding unmasked hypergraphs are independently encoded with the same E_θ to provide latent targets. The temporal predictor Q_ϕ contains three transformer-style temporal blocks with latent dimension 128, four attention heads, and dropout 0.1.

Stage I minimizes

$$\mathcal{L}_{\mathrm{I}} = \mathcal{L}_{\mathrm{mask}} + \lambda_H \mathcal{L}_{\mathrm{CSDR}}.$$

We train Stage I for 80 epochs with a global batch size of 64 distributed across the two GPUs. No actions are provided during this stage.

B.5 Stage II: Action-Conditioned Post-Training

Stage II initializes the encoder from the Stage-I checkpoint and learns how joint actions modify the latent coordination state. The action-conditioned predictor P_ψ contains three layers with latent dimension 128 and four attention heads.

Each action is first mapped to a 32-dimensional embedding and concatenated with the corresponding agent representation before projection back to the 128-dimensional latent space. Resource updates are then inferred through learned agent–resource attention over the persistent resource embeddings.

Table 7: Default TH-JEPA training configuration used in the reported experiments.

Setting	Value
Latent dimension d	128
Encoder layers	3
Attention heads	4
Predictor layers	3
Feed-forward dimension	256
Dropout	0.1
Prediction horizons $\mathcal{K}$	$\{1, 2, 4, 8\}$
Horizon weights	$(0.4, 0.3, 0.2, 0.1)$
Agent/resource loss weight β	1.0
Stage-I temporal window	$t-4:t+4$
Agent masking ratio	0.40
Resource masking ratio	0.40
Stage-I global batch size	64
Stage-I epochs	80
Stage-II global batch size	32
Stage-II epochs	60
Optimizer	AdamW
Learning rate	3×10^{-4}
Weight decay	10^{-4}
Gradient clipping	1.0
Warm-up	5% of updates
LR schedule	cosine decay
Precision	mixed precision
Hardware	2× RTX4090
Conflict horizon H_c	8
CSDR projection dimension q	32
Energy threshold γ	0.05
Decorrelation weight λ_o	0.1
CSDR loss weight λ_H	0.05

We supervise the prediction horizons

$$\mathcal{K} = \{1, 2, 4, 8\}.$$

The horizon weights are

$$(w_1, w_2, w_4, w_8) = (0.4, 0.3, 0.2, 0.1),$$

placing slightly greater emphasis on shorter horizons while still requiring accurate long-range prediction.

Starting from the observed latent state, the predictor is recursively rolled forward without teacher forcing. Intermediate ground-truth latent states and future incidence matrices are not provided. Targets are independently obtained from

$$(Z_{t+k}, U_{t+k}) = E_\theta(\mathcal{H}_{t+k}).$$

Because the agent and resource errors are already normalized by N and M, respectively, we use

$$\beta = 1.$$

Stage II is trained for 60 epochs with a global batch size of 32. The smaller batch size reflects the additional memory required by recursive multi-horizon prediction.

B.6 Conflict-Spectral Dispersion Regularization

CSDR is applied to sampled context and target representations in both training stages. Since the latent dimension is 128, we use a structural projection dimension

$$q = 32.$$

This retains enough directions to measure structural diversity while keeping the regularization cost small relative to the encoder.

The orthonormal projection $R \in \mathbb{R}^{128\times 32}$ is initialized once and kept fixed during training. For $Y_t = Z_t R$, the normalized projected representation $\bar{Y}_t$ is used to construct

$$C_t = \bar{Y}_t^\top L_t \bar{Y}_t.$$

We use

$$\gamma = 0.05, \qquad \lambda_o = 0.1, \qquad \lambda_H = 0.05.$$

The structural-energy threshold γ prevents projected directions from approaching the Laplacian nullspace, while λ_o controls redundancy between structural directions. The relatively small outer weight λ_H keeps CSDR auxiliary to the predictive objective rather than allowing it to dominate training.

Inactive resources have zero hyperedge degree. We therefore construct

$$L_t = D_v - B_t W_t D_e^\dagger B_t^\top$$

using the Moore–Penrose inverse of D_e. Unless otherwise stated, $W_t = I$ for active resources.

B.7 Optimization

Both training stages use AdamW with

$$\text{learning rate} = 3 \times 10^{-4}, \qquad \text{weight decay} = 10^{-4}.$$

We use the standard Adam coefficients $(\beta_1, \beta_2) = (0.9, 0.999)$ and clip the global gradient norm to 1.0. The learning rate is linearly warmed up during the first 5% of optimization steps and then decayed using a cosine schedule.

Training uses automatic mixed precision on two NVIDIA RTX4090 GPUs. The implementation uses distributed data parallelism, so each GPU stores a full copy of the model while processing a different portion of the mini-batch.

The complete default configuration is summarized in Table 7.

Tables 8 and 10 provide the complete TH-JEPA configuration, including all data, architectural, training, regularization, planning, and evaluation settings used in the experiments.

B.8 Latent-Space Planning

At inference time, the observed state is converted to $\mathcal{H}_t$ and encoded once to obtain (Z_t, U_t). We then construct candidate joint-action sequences using beam search rather than enumerating the full joint-action space.

At each planning step, individually infeasible actions are removed before forming joint candidates. We use a beam width of 32 and retain at most 128 candidate sequences. Each sequence is rolled forward for

$$H = 8$$

latent steps using P_ψ. Candidate rollouts are batched so that the additional planning computation can be evaluated efficiently on the GPU.

The readout C_ω predicts normalized collision, congestion, waiting, and goal-progress quantities from each latent rollout. The scalar planning score is

$$J_\omega = 5\widehat{C}_{\text{col}} + 2\widehat{C}_{\text{cong}} + 0.5\widehat{C}_{\text{wait}} - \widehat{C}_{\text{goal}}.$$

where C_{col} represents predicted collision risk, C_{cong} measures shared-resource congestion, C_{wait} penalizes unnecessary waiting, and C_{goal} rewards progress toward the assigned goals. Collision risk receives the largest coefficient because safety remains the primary constraint.

The lowest-cost sequence is selected, but only its first joint action is executed. The next observation is then encoded and planning is repeated, yielding the receding-horizon procedure described in the main paper.

Table 8: Data, hypergraph construction, architecture, and Stage-I pretraining configuration used for TH-JEPA.

Group	Setting	Stage I	Stage II	Planning / Evaluation	Notes
Data and environment					
Data	Training instances	21K		–	POGEMA trajectories generated with LaCAM3
Data	Training layouts	20% Random + 80% Maze		Room, Warehouse, MovingAI	Room/Warehouse excluded from training
Data	Training map size	17×17–21×21		larger held-out maps	–
Data	Agent population	$N \in \{16, 24, 32\}$		$N \in \{64, 90, 128, 192, 256\}$	larger populations evaluated zero-shot
Data	Train / validation / held-out split	80% / 10% / 10%		–	split at problem-instance level
Data	Expert solver	LaCAM3		LaCAM3 reference	complete trajectories retained
Data	Action space	not used	WAIT, UP, DOWN, LEFT, RIGHT	same	discrete joint actions
Conflict-hypergraph construction					
Hypergraph	Conflict look-ahead horizon	$H_c = 8$		8	nominal routes obtained using A*
Hypergraph	Resource types	intersections, narrow aisles, merge regions, bottlenecks			persistent across an episode
Hypergraph	Active hyperedge criterion	$\lvert e_r \rvert \geq 2$		same	otherwise incidence column is zero
Hypergraph	Resource weights	$W_t = I$ for active resources		–	default Laplacian construction
Hypergraph	Inactive-resource inverse	$D_e^{\dagger}$		–	Moore–Penrose inverse
Hypergraph	Resource identity	persistent		persistent	membership changes over time
Shared hypergraph encoder					
Encoder	Latent dimension	128	128	128	$d = 128$
Encoder	Encoder layers	3	3	3	agent–resource message passing
Encoder	Attention heads	4	4	–	–
Encoder	Feed-forward dimension	256	256	–	–
Encoder	Dropout	0.10	0.10	disabled	inference uses deterministic model
Encoder	Context encoder	E_θ	E_θ	E_θ	shared across stages
Encoder	Target encoder	shared E_θ	shared E_θ	–	no separate EMA encoder
Encoder	Agent latent shape	$N \times 128$	$N \times 128$	$N \times 128$	Z_t
Encoder	Resource latent shape	$M \times 128$	$M \times 128$	$M \times 128$	U_t
Stage I: masked temporal pretraining					
Stage I	Temporal window	$t-4 : t+4$	–	–	nine consecutive hypergraphs
Stage I	Temporal predictor	3 blocks	–	–	transformer-style predictor Q_ϕ
Stage I	Predictor latent dimension	128	–	–	–
Stage I	Predictor attention heads	4	–	–	–
Stage I	Predictor dropout	0.10	–	–	–
Stage I	Agent masking ratio	0.40	–	–	temporal token masking
Stage I	Resource masking ratio	0.40	–	–	sampled independently
Stage I	Action conditioning	none	joint actions	candidate actions	Stage I is action-free
Stage I	Agent/resource loss weight	$\beta = 1.0$	$\beta = 1.0$	–	–
Stage I	Objective	$\mathcal{L}_{\text{mask}} + \lambda_H \mathcal{L}_{\text{CSDR}}$	–	–	latent target prediction
Stage I	Global batch size	64	–	–	distributed across two GPUs
Stage I	Training epochs	80	–	–	–

Table 9: Default latent-planning configuration.

Setting	Value
Planning horizon H	8
Beam width	32
Maximum candidate sequences	128
Collision weight λ_{col}	5.0
Congestion weight λ_{cong}	2.0
Waiting weight λ_{wait}	0.5
Goal-progress weight λ_{goal}	1.0

C Evaluation Protocol

C.1 Evaluation Environments

The small-map evaluation contains Random Obstacles, Dense Maze, Dense Room, and Dense Warehouse environments. Random Obstacles is closest to the training distribution, while the remaining layouts introduce increasingly structured and repeated coordination constraints.

Dense Maze contains narrow passages and long corridor dependencies. Dense Room changes the global topology relative to the maze-dominated training set. Dense Warehouse contains repeated aisles, intersections, and merge points, creating the highest frequency of shared-resource conflicts.

The principal zero-shot evaluation uses

$$N \in \{64, 90, 128\},$$

although the model is trained only with $N \in \{16, 24, 32\}$. We evaluate 500 MAPF instances for every environment–population configuration. Each learned configuration is trained with five independent random seeds, and all methods are evaluated on exactly the same test instances.

Table 10: Action-conditioned training, structural regularization, optimization, planning, and evaluation configuration used for TH-JEPA.

Group	Setting	Stage I	Stage II	Planning / Evaluation	Notes
Stage II: action-conditioned latent dynamics					
Stage II	Initialization	–	Stage-I checkpoint	trained Stage-II model	encoder transferred from Stage I
Stage II	Predictor	–	P_ψ	P_ψ	recursive latent world model
Stage II	Predictor layers	–	3	3	–
Stage II	Predictor latent dimension	–	128	128	–
Stage II	Attention heads	–	4	–	–
Stage II	Action embedding dimension	–	32	32	fused with agent latent state
Stage II	Prediction horizons	–	$\mathcal{K} = \{1, 2, 4, 8\}$	$H = 8$	recursive rollout
Stage II	Horizon weights	–	$(0.4, 0.3, 0.2, 0.1)$	–	(w_1, w_2, w_4, w_8)
Stage II	Teacher forcing	–	none	none	intermediate targets not injected
Stage II	Future incidence input	–	not provided	not provided	inferred from latent dynamics
Stage II	Loss weight	–	$\beta = 1.0$	–	agent/resource latent losses
Stage II	Objective	–	$\mathcal{L}_{\text{pred}} + \lambda_H \mathcal{L}_{\text{CSDR}}$	–	multi-horizon prediction
Stage II	Global batch size	–	32	–	–
Stage II	Training epochs	–	60	–	–
Conflict-Spectral Dispersion Regularization					
CSDR	Projection dimension	$q = 32$	$q = 32$	–	–
CSDR	Projection matrix	fixed orthonormal R	fixed orthonormal R	–	$R \in \mathbb{R}^{128\times 32}$
CSDR	Energy threshold	$\gamma = 0.05$	$\gamma = 0.05$	–	prevents structural collapse
CSDR	Decorrelation weight	$\lambda_o = 0.10$	$\lambda_o = 0.10$	–	off-diagonal penalty
CSDR	CSDR loss weight	$\lambda_H = 0.05$	$\lambda_H = 0.05$	–	used in both stages
CSDR	Hyperedge weights	$W_t = I$	$W_t = I$	–	active resources
Optimization and compute					
Optimization	Optimizer	AdamW	AdamW	–	–
Optimization	Learning rate	3×10^{-4}	3×10^{-4}	–	–
Optimization	Weight decay	10^{-4}	10^{-4}	–	–
Optimization	Adam coefficients	$(0.9, 0.999)$	$(0.9, 0.999)$	–	(β_1, β_2)
Optimization	Gradient clipping	1.0	1.0	–	global norm
Optimization	Warm-up	5%	5%	–	fraction of updates
Optimization	LR schedule	cosine decay	cosine decay	–	–
Optimization	Precision	mixed	mixed	mixed	automatic mixed precision
Optimization	Parallelization	DDP	DDP	batched rollout	–
Optimization	Hardware	2× RTX4090	2× RTX4090	same	–
Latent-space planning					
Planner	Planning horizon	–	–	$H = 8$	receding horizon
Planner	Beam width	–	–	32	bounded search
Planner	Maximum candidates	–	–	128	$\lvert\mathcal{C}_t\rvert \le 128$
Planner	Collision weight	–	–	$\lambda_{\text{col}} = 5.0$	–
Planner	Congestion weight	–	–	$\lambda_{\text{cong}} = 2.0$	–
Planner	Waiting weight	–	–	$\lambda_{\text{wait}} = 0.5$	–
Planner	Goal-progress weight	–	–	$\lambda_{\text{goal}} = 1.0$	subtracted from cost
Planner	Candidate evaluation	–	–	parallel latent rollout	–
Planner	Execution	–	–	first joint action	replanned every environment step
Evaluation protocol					
Evaluation	Small-map populations	–	–	$\{64, 90, 128\}$	zero-shot population shift
Evaluation	Large populations	–	–	$\{128, 192, 256\}$	Warehouse / MovingAI
Evaluation	Instances per setting	–	–	500	matched across methods
Evaluation	Independent seeds	–	–	5	–
Evaluation	Small-map episode limit	–	–	256	environment steps
Evaluation	Large-map episode limit	–	–	512	environment steps
Evaluation	Primary metrics	–	–	SR, Rel. SoC, MS, B, W	plus prediction and latency metrics

We use a maximum episode length of 256 environment steps for the small-map experiments. Larger-map experiments use a limit of 512 steps.

C.2 Evaluation Metrics

Success rate. An instance is considered successful only when all agents reach their assigned goals within the episode limit without violating vertex or edge-swap collision constraints. Success rate is the percentage of successful test instances.

Sum of Costs. For agent i, let T_i denote the first time at which it reaches its goal. The Sum of Costs is

$$\mathrm{SoC} = \sum_{i=1}^{N} T_i.$$

Relative Sum of Costs. To make cost comparable across maps and population sizes, we normalize against the LaCAM3 expert solution on the same instance:

$$\mathrm{Rel.SoC} = \frac{\mathrm{SoC}_{\mathrm{method}}}{\mathrm{SoC}_{\mathrm{LaCAM3}}}.$$

A value close to one therefore indicates a solution whose total path cost is close to the expert reference.

Makespan. Makespan measures the time required for the final agent to reach its goal,

$$\mathrm{Makespan} = \max_i T_i.$$

Blocking events. A blocking event is counted when an unfinished agent cannot execute its intended motion because the target cell or corresponding transition is occupied by another agent. Consecutive blocking at different time steps is counted separately because it contributes directly to congestion persistence.

Cumulative waiting. Waiting is computed as

$$\mathrm{Wait} = \sum_t \sum_i \mathbf{1}[p_i^{t+1} = p_i^t,\, p_i^t \neq g_i].$$

Thus, waiting after an agent has reached its goal is not penalized.

Inference latency. Latency is measured with batch size one at the environment level and includes hypergraph construction, encoding, latent rollout, and candidate scoring, but excludes rendering and disk I/O. We discard the first 100 replanning steps as warm-up and report the mean over the next 1000 steps.

C.3 Predictive-Dynamics Evaluation

To evaluate the Stage-II world model independently of the planner, we start from the encoded ground-truth state (Z_t, U_t) and recursively roll the predictor using the recorded ground-truth joint actions. The future incidence matrices and future encoded states are not exposed to the predictor.

At horizon k, Node MSE is

$$\mathrm{NodeMSE}(k) = \frac{1}{N}\|\widehat{Z}_{t+k} - Z_{t+k}\|_F^2,$$

while Resource MSE is

$$\mathrm{ResourceMSE}(k) = \frac{1}{M}\|\widehat{U}_{t+k} - U_{t+k}\|_F^2.$$

Cosine similarity is computed per agent and then averaged across agents and evaluation states.

Conflict F1. We use a fixed two-layer MLP probe to predict future agent–resource incidence from the latent state. Given the pair (z_i^{t+k}, u_r^{t+k}), the probe predicts whether $(B_{t+k})_{ir} = 1$. The hidden dimension is 128, and the same probe architecture and decision threshold are used at every prediction horizon. Conflict F1 is computed over all evaluated agent–resource pairs.

Action-shuffling test. To verify that the predictor uses the supplied actions, we repeat the same rollouts after randomly permuting the action sequence across samples while keeping the initial latent state unchanged. A substantial degradation under this intervention indicates that the prediction cannot be explained by action-independent temporal extrapolation.

Counterfactual ranking. For each sampled state, we generate two feasible candidate joint-action sequences and execute both in the simulator to obtain their realized planning costs. The latent predictor is correct when it assigns lower predicted cost to the candidate with lower realized cost. We use 1000 such action pairs for the reported ranking experiment.

C.4 Representation-Collapse Metrics

We evaluate representation collapse using effective rank, average pairwise cosine similarity, and hypergraph structural energy.

Let $\sigma_1, \ldots, \sigma_q$ denote the singular values of the projected representation and define

$$p_j = \frac{\sigma_j}{\sum_k \sigma_k}.$$

The effective rank is

$$\mathrm{EffRank} = \exp\left(-\sum_{j=1}^{q} p_j \log p_j\right).$$

Since CSDR uses $q = 32$, a well-dispersed projected representation can attain an effective rank close to 32.

Structural energy is computed as

$$E_{\mathcal{H}} = \frac{1}{q}\,\mathrm{tr}\left(\bar{Y}^\top L_{\mathcal{H}} \bar{Y}\right).$$

Representations approaching the hypergraph-Laplacian nullspace have low structural energy.

Pairwise cosine similarity is computed between projected latent directions. High average similarity indicates redundant representation dimensions.

D Baseline Configuration

We compare TH-JEPA with search-based, pairwise learned, higher-order learned, and world-model approaches.

LaCAM3. LaCAM3 Okumura (2024) is used both as a strong search-based baseline and as the expert trajectory generator.

PIBT. PIBT Okumura et al. (2022) provides a lightweight classical coordination baseline based on dynamic agent priorities.

MAGAT. MAGAT Li et al. (2021) provides a learned pairwise graph-communication baseline. We use the same test maps, start states, goals, and episode limits as for TH-JEPA.

HMAGAT. HMAGAT Jain et al. (2026) is the closest higher-order representation baseline. It captures group interactions using hypergraphs but predicts actions from the current coordination state rather than learning an action-conditioned latent world model.

MAPF-GPT-85M. MAPF-GPT Andreychuk et al. (2025) represents large-scale imitation learning from expert demonstrations. We use the released 85M-parameter configuration and preserve its published inference procedure.

MAPF-World. MAPF-World Yang et al. (2025) is the closest temporal world-model baseline. It explicitly predicts future MAPF dynamics but does not maintain persistent resource states or higher-order conflict hyperedges.

All learned methods are evaluated on identical test instances. Whenever a released pretrained model is used, we report it as such rather than presenting the comparison as a matched-data retraining experiment.

Table 11: Zero-shot success rate (%) on larger unseen maps.

Method	$N = 128$	$N = 192$	$N = 256$
MAGAT	89.8	76.6	61.2
HMAGAT	91.5	82.4	70.1
MAPF-World	92.0	83.8	72.6
TH-JEPA	**93.1**	**88.0**	**80.4**

Table 12: TH-JEPA under increasing agent density. Active resources and maximum hyperedge cardinality are averaged over evaluation episodes.

Density	SR ↑	Blocking ↓	Active Res.	Max. $\lvert e_r \rvert$
0.15	99.1	1.8	2.1	3.2
0.20	98.4	3.7	3.6	4.1
0.25	96.9	6.4	5.2	5.3
0.30	94.5	10.2	7.3	6.6
0.35	89.8	15.8	9.8	8.1

E Additional Out-of-Distribution Evaluation

The main paper evaluates zero-shot scaling from at most 32 training agents to 128 agents on small maps. We further consider larger environments and more agents to test whether the learned coordination representation transfers beyond both the training topology and population range.

E.1 Larger Maps

We evaluate on held-out MovingAI-style maps that are not used during training. TH-JEPA is applied without additional fine-tuning at $N \in \{128, 192, 256\}$. Persistent resources are extracted from each unseen map using the same procedure described in Appendix B.2.

The performance gap is modest at $N = 128$ but increases as the population grows. At $N = 256$, TH-JEPA retains an SR of 80.4%, compared with 72.6% for MAPF-World and 70.1% for HMAGAT. This suggests that the resource-based latent state remains useful when substantially more agents interact than were observed during training.

E.2 Controlled Density Shift

Population size alone does not fully characterize MAPF difficulty. Higher occupancy also increases how frequently agents compete for the same resources and how many agents participate in each conflict. We therefore hold the map distribution fixed and vary agent occupancy from 0.15 to 0.35.

As density increases, both the number of active resources and the size of the largest conflict groups increase steadily. TH-JEPA degrades gradually under this shift, retaining 89.8% success at 0.35 occupancy despite an average maximum conflict cardinality above eight agents. This complements the cardinality analysis in the main paper by controlling the environment while systematically increasing interaction complexity.

F Masked Pretraining Analysis

The main ablation shows that removing Stage-I pretraining reduces downstream performance. We next examine how the masking configuration affects the learned temporal representation.

F.1 Masking Ratio

We vary the masking ratio over $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ while applying the same ratio to agent and resource tokens. Small ratios make the prediction task relatively easy, whereas aggressive masking removes useful temporal context. The full model uses a masking ratio of 0.4.

Table 13: Sensitivity to the Stage-I masking ratio. Downstream SR is measured after identical Stage-II post-training on Dense Warehouse.

Mask ratio	Masked latent MSE ↓	Downstream SR ↑
0.1	0.021	92.7
0.2	0.028	93.6
0.3	0.036	94.5
0.4	0.047	**95.3**
0.5	0.063	94.4

Table 14: Effect of the Stage-I masking target.

Masked tokens	Conflict F1 ↑	Downstream SR ↑
Agents only	0.861	93.8
Resources only	0.869	94.3
Agents + resources	**0.881**	**95.3**

Masked prediction error naturally increases with the masking ratio because less context is available. Downstream performance follows a different trend: moderate masking provides increasingly useful temporal supervision up to 0.4, while masking half of the tokens begins to remove too much information. We therefore use 0.4 in the full model.

F.2 Agent and Resource Masking

We next separate the two types of temporal masking. Agent-only masking mainly requires the model to recover motion and local interaction states, whereas resource-only masking emphasizes the evolution of shared conflicts. Joint masking requires both levels of the representation to be inferred from the remaining temporal context.

Resource masking provides a slightly larger gain than agent-only masking, consistent with the importance of predicting shared conflict evolution. Masking both token types performs best, indicating that agent motion and resource dynamics provide complementary temporal supervision.

G Planner Sensitivity

G.1 Planning Horizon

We vary the latent planning horizon over $H \in \{1, 2, 4, 8\}$. Short horizons reduce computation but cannot expose delayed congestion, while longer rollouts provide more information about future conflicts at the cost of additional prediction and inference time.

Performance improves consistently as the planning horizon increases. The gain from $H = 4$ to $H = 8$ is smaller than at shorter horizons, indicating diminishing returns as recursive prediction becomes less certain. We use $H = 8$ because it provides the strongest coordination performance while remaining within the horizons explicitly supervised during Stage II.

G.2 Candidate Budget

We next vary the number of candidate joint-action sequences considered by the planner. All other components, including the planning horizon and cost function, remain fixed.

A larger candidate set improves the probability of finding a coordinated low-cost sequence, but the benefit saturates. Increasing the budget from 128 to 256 nearly doubles planning latency while improving SR by only 0.3 percentage points. We therefore use 128 candidates in the default configuration.

Table 15: Sensitivity to the latent planning horizon on Dense Warehouse. Latency is measured per replanning step.

H	SR ↑	Rel. SoC ↓	Latency (ms) ↓
1	90.6	1.20	9.4
2	92.4	1.17	13.7
4	94.2	1.13	21.5
8	**95.3**	**1.11**	31.9

Table 16: Effect of candidate-sequence budget on Dense Warehouse.

Candidates	SR ↑	Rel. SoC ↓	Latency (ms) ↓
32	92.1	1.18	14.6
64	93.6	1.15	21.2
128	**95.3**	**1.11**	31.9
256	95.6	1.10	55.7

H Computational Efficiency

TH-JEPA introduces additional inference cost because candidate futures are evaluated before an action is executed. The current hypergraph is nevertheless encoded only once per replanning step, and candidate latent trajectories are evaluated in parallel on the GPU.

Table 17 reports approximate model size, mean replanning latency, peak GPU memory, and Dense Warehouse success rate under the same evaluation setting.

TH-JEPA is slower than direct action-prediction methods because it performs multi-step latent planning. However, its 31.9 ms replanning time remains well below one tenth of a second in this configuration. The additional computation is accompanied by substantially higher success in the dense setting. TH-JEPA also uses fewer parameters than the evaluated MAPF-World configuration while operating on compact 128-dimensional agent and resource latents.

I Qualitative Rollout Protocol

The qualitative rollout in Fig. 4 uses the same environment, initial agent positions, and goals for every method. Agent identities are preserved across rows using a fixed color assignment, and all displayed snapshots are drawn from the same complete episode.

Makespan, blocking events, and cumulative waiting are computed from the full trajectory rather than from the displayed frames. The shaded region denotes a persistent resource identified using the same map preprocessing used by TH-JEPA. The purpose of the visualization is therefore to show how different methods resolve the same shared-resource conflict over time rather than to compare different problem instances.

For the final reported experiment, all trajectories and episode-level metrics are generated directly by the corresponding evaluated models. No agent positions are manually modified for visualization.

Qualitative Coordination Analysis.

Figure 4 shows a representative Dense Warehouse instance with a shared intersection connecting opposing traffic streams. All methods use the same map, initial agent positions, and goals. We report makespan (MS), blocking events (B), and cumulative waiting over the complete episode.

HMAGAT sends several opposing flows toward the intersection at the same time, which produces repeated blocking near the bottleneck. MAPF-World reduces this congestion, although multiple flows still enter the constrained region at similar times. TH-JEPA instead delays some agents while another group clears the resource. This behavior trades short-term individual progress for lower congestion over the full episode. HMAGAT requires 47 steps with 18 blocking events and 58 cumulative wait steps, MAPF-World requires 36 steps with 9 blocking events and 27 wait steps, and

Table 17: Inference efficiency on Dense Warehouse at $N = 128$. Latency is measured per environment replanning step.

Method	Params	Latency (ms)	Peak Mem. (GB)	SR ↑
MAGAT	1.2M	6.4	0.72	2.3
HMAGAT	2.1M	9.8	0.93	89.1
MAPF-World	6.7M	18.6	1.48	84.4
TH-JEPA	4.8M	31.9	2.05	**94.5**

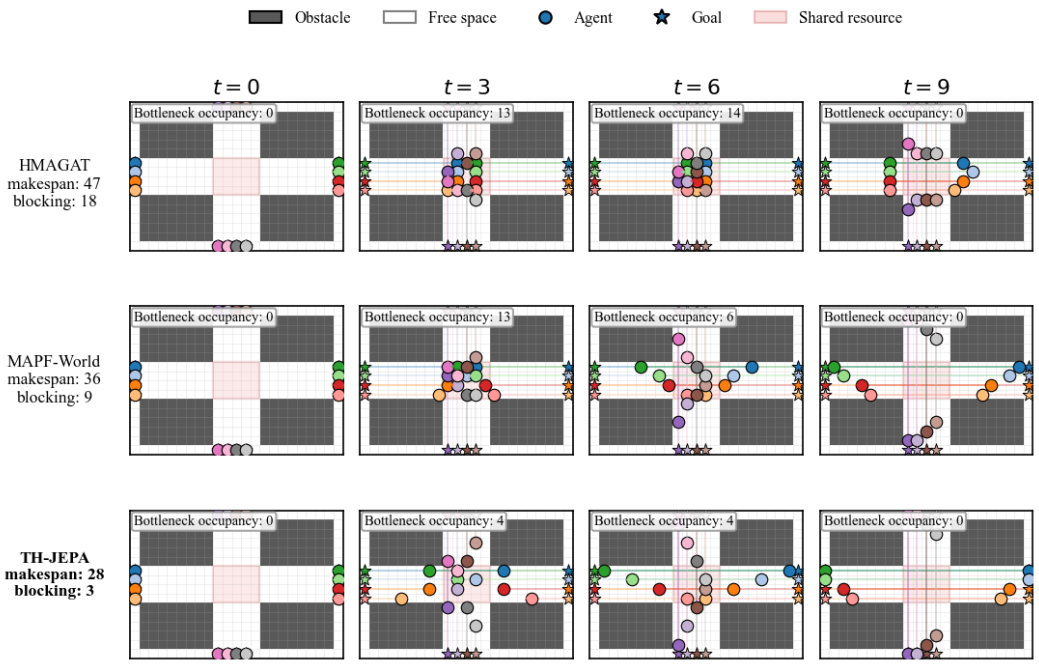

Figure 4: Qualitative rollouts on one held-out Dense Warehouse episode with identical map, start positions, and goals for all methods. Rows correspond to HMAGAT, MAPF-World, and TH-JEPA, while columns show common simulation times. Circles denote agents, stars denote goals, and the shaded region marks the persistent bottleneck resource. Episode-level makespan (MS) and blocking events (B) are reported beside each method.

TH-JEPA completes the instance in 28 steps with 3 blocking events and 12 wait steps. The example illustrates the behavior examined quantitatively in the conflict-cardinality analysis below.