IMPROVING ENTITY-CONFIGURATION GENERALIZATION IN REAL-TIME STRATEGY GAMES WITH TASK HYPERGRAPH REPRESENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

This paper investigates compositional generalization across entity configurations in real-time strategy (RTS) games, which remains challenging for existing methods. To address this problem, we propose a task-hypergraph-based policy architecture that represents game entities as nodes and task instances involving multiple entities, such as resource-gathering and attacking, as hyperedges. The architecture comprises a task hypergraph constructor and a task hypergraph network. We encode RTS domain knowledge as cross-scenario task schemas, feasibility constraints, and an objective function over task-instance sets, thereby formulating hypergraph construction as a constrained optimization problem over sets of task instances. Given the current state, the constructor instantiates schema-valid task instances and selects a jointly feasible subset using the set-level objective. The selected instances then induce the task hypergraph. The task hypergraph network aggregates information among entities participating in the task instance through node-to-hyperedge-to-node message passing and generates entity-level actions using multiple action heads, relation encodings, and hyperedge-type-dependent soft biases. By representing changes in entity configurations as recombinations of task instances and their participating entities under shared task schemas, our approach enables consistent task-level processing across scenarios and thereby supports compositional generalization.

1 INTRODUCTION

Deep reinforcement learning has achieved strong performance in RTS games under fixed scenarios (Vinyals et al., 2019; Wang et al., 2021). However, policy generalization in RTS environments, particularly under changes in initial entity configurations, remains comparatively underexplored, and high performance in training scenarios does not necessarily transfer to unseen environments. Our experiments confirm this limitation, showing that policies trained on specific maps can suffer substantial performance degradation when the initial entity configurations change. Accordingly, this paper studies compositional generalization across entity configurations. The training and test scenarios share the same entity types, action space, interaction rules, and task objective, but differ in their initial entity-type compositions and spatial layouts. For example, a training map may contain only bases and workers, whereas a test map may additionally include barracks or combat units and alter the relative positions of entities.

Existing generalization approaches, including domain randomization, data augmentation, regularization, and invariant representation learning, primarily improve data coverage or reduce dependence on environment-specific features (Kirk et al., 2023; Korkmaz, 2024). Although effective against visual variations, environmental perturbations, and observation noise, they struggle with distribution shifts caused by different initial entity-type compositions and spatial layouts.

Battaglia et al. (2018) argue that compositional generalization depends on structured representations of entities, relations, and their composition rules, and that manually designed structural priors can complement end-to-end learning: the former provide inductive biases aligned with the problem structure, while the latter learn concrete relational representations and decision rules from interaction data. Inspired by this perspective, we represent an RTS state as a structured object composed of

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.

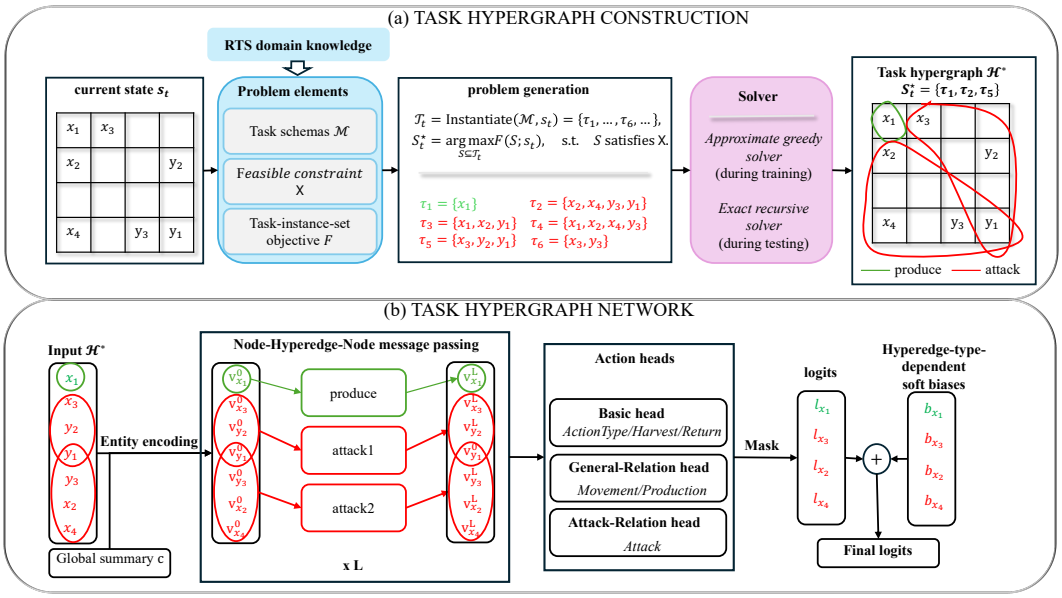

Figure 1: Schematic overview of the proposed architecture. (a) The current state s_t, where x and y denote controlled and opponent entities, is transformed into a task hypergraph. (b) The hypergraph network generates entity-level actions through task-structured message passing. Opponent entities contribute to representation but are masked from action output.

entities and their tactical relations, enabling our policy network to learn reusable local relational patterns rather than directly fitting the global state of an entire scenario.

Graph representations introduce relational inductive biases by modeling entities as nodes and relations as edges. However, binary graphs are limited to pairwise relations and cannot naturally represent multi-entity RTS tasks, such as resource gathering involving a worker, a resource, and a base. We therefore propose a task-hypergraph-based policy architecture that represents game entities as nodes and multi-entity tasks as hyperedges. The architecture comprises a task hypergraph constructor, which reconstructs each state as a task hypergraph, and a task hypergraph network, which aggregates intra-task information and generates actions for controlled entities, as shown in Figure 1.

To transform an environment state into a task hypergraph, we use RTS domain knowledge to define task schemas, feasibility constraints, and a task-instance-set objective shared across scenarios, thereby formulating hypergraph construction as a constrained optimization problem. We define five task categories, each associated with a schema specifying its entity roles, admissible entity types, role capacities, and state-dependent validity conditions. A task instance is a state-specific instantiation of a schema and records its category, participating entities, resource cost, and other attributes. The constraints ensure joint executability and compliance with the available resource budget, while the objective evaluates feasible task-instance sets according to task and combat utility, distance cost, and resource cost. Accordingly, the optimization problem is to find a task-instance set that maximizes the task-instance-set objective subject to the above constraints. The resulting set induces a task hypergraph, with entities as nodes and selected task instances as hyperedges. To solve this optimization problem, we develop an exact recursive solver that identifies the globally optimal feasible task-instance set and an efficient greedy solver that incrementally constructs a feasible set using objective-informed heuristics. We use the greedy solver during training for computational efficiency and replace it with the recursive solver during testing to improve hypergraph construction quality without retraining the policy network.

Regarding the network architecture, the task hypergraph network performs message passing within each selected task hyperedge. It first aggregates the states and relative features of participating entities into a hyperedge representation and then propagates this representation back to the associated nodes for entity-level action selection. To handle the composite action space, which consists of

action type, movement direction, harvesting direction, resource-return direction, production direction, production type, and attack location, the network employs multiple action heads and relation encodings tailored to different action components. For example, the attack-relation head encodes the relative distance between a controlled entity and its target and whether the target is within attack range. The network also applies hyperedge-type-dependent soft biases to encourage task-relevant actions. For example, it increases the preference for attack actions for a controlled entity participating in an attacking task hyperedge.

Our method promotes compositional generalization by converting entity-configuration changes into variations in the number, membership, and organization of task instances. Shared task schemas, feasibility constraints, and the task-instance-set objective provide a consistent state decomposition across training and test scenarios. The task hypergraph network processes each selected task instance as a local information-exchange unit and generates actions based on task types, entity roles, and relative relations. Consequently, learned task-level decision patterns can be reused when familiar task structures are newly instantiated and recombined. Experiments in full-game μRTS demonstrate improved generalization to unseen initial entity-type compositions and spatial layouts.

Our main contributions are as follows:

- We propose a task hypergraph representation for entity-configuration generalization in RTS games. It represents game entities as nodes and multi-entity task instances as hyperedges, allowing different entity configurations to be expressed through recombinations of task instances.

- We formulate task hypergraph construction as a constrained optimization problem over task-instance sets, based on shared task schemas, feasibility constraints, and a set-level objective. We develop an exact recursive solver and an efficient greedy solver for this problem.

- We design a task hypergraph network. It combines node-to-hyperedge-to-node message passing with multiple action heads, relation encodings, and hyperedge-type-dependent soft biases to generate entity-level actions.

2 Related Work

Compositional Generalization in Reinforcement Learning. Compositional generalization concerns reusing learned components in unseen combinations. Existing work mainly studies two forms of composition. The first composes subtasks through parameterized skills and hierarchical control (Oh et al., 2017), reusable subtask policies (Jothimurugan et al., 2023), or automatically generated curricula of interdependent subtasks (Gur et al., 2021). The second considers entity and structural recombination, including new combinations of robots, objects, goals, and obstacles (Mendez et al., 2022), object-oriented or MDP-based compositional representations (Zhao et al., 2022; Bakirtzis et al., 2024), and entity-based policies for varying entity sets and goals (Zhou et al., 2024). These methods primarily operate on predefined compositions specified at the task, skill, or model level. In contrast, our method derives the compositional structure directly from each state by instantiating and selecting task instances to construct a task hypergraph.

Hypergraph-Based Reinforcement Learning. Existing studies mainly follow two directions. The first models higher-order coordination among multiple agents by representing agent groups as hyperedges, with hypergraph structures constructed through learned coordination rewards, neighborhood interactions, dynamic clustering, or stochastic sampling (Bai et al., 2022; Zhang et al., 2022; Liu & Li, 2025; Zhao et al., 2025). The second applies hypergraphs to domain-specific relations, including spatio-temporal dependencies among traffic intersections, interactions among robots, humans, and points of interest, and user–attribute–item preferences in conversational recommendation (Wang et al., 2025; Wang et al., 2024; Zhao et al., 2023). These studies use hypergraphs mainly for higher-order coordination or state representation, without exploiting cross-scenario semantic consistency to promote generalization. In contrast, our approach represents entities as nodes and selected task instances as hyperedges, using shared task schemas and task-set optimization to preserve task semantics across scenarios and facilitate the reuse of task-level decision patterns.

Deep Reinforcement Learning for Full-Game RTS. Reinforcement learning in full-game RTS environments has mainly addressed long decision horizons, large state–action spaces, and the coupling of strategic planning with unit control. SC2LE established a standard interface for StarCraft

II research (Vinyals et al., 2017), while AlphaStar achieved Grandmaster-level performance through imitation learning and league-based training (Vinyals et al., 2019). TStarBot-X (Han et al., 2020), SCC (Wang et al., 2021), and hierarchical approaches (Pang et al., 2019) improved training efficiency and long-horizon decision-making through rule-guided search, lightweight architectures, macro-actions, and curriculum transfer. Gym-μRTS enabled affordable full-game research with invalid-action masking (Huang et al., 2021), while later work addressed varying map sizes through spatial pyramid pooling (Lemos et al., 2025). However, these studies devote limited attention to generalization across initial entity-type compositions and spatial layouts, and do not explicitly model dynamic higher-order task relations among RTS entities.

3 METHOD

3.1 PROBLEM FORMULATION

We formulate RTS decision-making as a single-agent reinforcement learning problem in which a centralized policy controls all player-owned entities. A scenario ξ is an MDP $\mathcal{M}_\xi = (\mathcal{S}, \mathcal{A}, P, R, s_0^\xi, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and joint action spaces, P is the transition dynamics, R is the reward function, s_0^ξ is the initial state, and $\gamma \in [0, 1)$ is the discount factor. At time t, the game objects in $s_t \in \mathcal{S}$ form a variable-size entity set $\mathcal{V}_t = \{v_{t,1}, \ldots, v_{t,N_t}\}$, with attributes $\mathbf{X}_t = \{\mathbf{x}_{t,i}\}_{i=1}^{N_t}$ encoding entity type, owner, position, health, carried resources, and action status. Let $\mathcal{V}_t^{\text{ctrl}} \subseteq \mathcal{V}_t$ denote the controlled entities. The policy outputs their joint action $\mathbf{a}_t = (a_{t,1}, \ldots, a_{t,N_t^{\text{ctrl}}})$, where $a_{t,i} = (a_{t,i}^{\text{type}}, a_{t,i}^{\text{move}}, a_{t,i}^{\text{harvest}}, a_{t,i}^{\text{return}}, a_{t,i}^{\text{produce-dir}}, a_{t,i}^{\text{produce-type}}, a_{t,i}^{\text{attack}})$. These components specify the action type and its movement, harvesting, return, production, and attack parameters, with invalid values excluded by legal-action masks. The environment then produces $s_{t+1} \sim P(\cdot \mid s_t, \mathbf{a}_t)$ and $r_t = R(s_t, \mathbf{a}_t, s_{t+1})$. We study compositional generalization across initial entity configurations. Training scenarios Ξ_{tr} and test scenarios Ξ_{te} share the same entity types, action definitions, interaction rules, task objectives and opponent settings, but differ in their initial entity-type compositions and spatial layouts. Test scenarios therefore contain unseen configurations of familiar entities rather than unseen entity types or game mechanics. A policy is trained only on Ξ_{tr} and evaluated without further training on Ξ_{te}, with generalization performance measured by its win rate across the test scenarios.

3.2 METHOD OVERVIEW

As shown in Figure 1, our policy architecture consists of a task hypergraph constructor and a task-hypergraph-guided hypergraph network, forming a state-to-hypergraph-to-action pipeline. Using cross-scenario task schemas, feasibility constraints, and a task-instance-set objective designed from RTS knowledge, the constructor first generates state-specific task instances and formulates their selection as a constrained optimization problem. Only the task instances returned by solving this problem induce the hyperedges of the final task hypergraph, $\mathcal{H}_t^\star = \mathcal{C}(s_t) = (\mathcal{V}_t, \mathcal{E}_t^\star)$. The network encodes $\mathcal{H}_t^\star$ through node-to-hyperedge-to-node message passing and generates entity actions using three decision heads, head-specific relation encodings, and hyperedge-type-dependent soft biases. Let π_θ denote the policy network parameterized by θ, which maps the constructed task hypergraph to entity-level actions. The complete decision process is $s_t \xrightarrow{\mathcal{C}} \mathcal{H}_t^\star \xrightarrow{\pi_\theta} \mathbf{a}_t$.

We train the actor and value network with PPO, using the greedy solver to construct hypergraphs efficiently at every interaction step. The trained network coupled with the greedy solver forms the Task Hypergraph-Greedy policy. At test time, the greedy solver can be replaced by the exact recursive solver without retraining the network, yielding the Task Hypergraph-Recursive policy. The two policies therefore share the same learned network parameters and differ only in the solver used for hypergraph construction.

3.3 TASK HYPERGRAPH CONSTRUCTION

At time t, the constructor operates on the current entity set $\mathcal{V}_t$ and the available resource budget B_t. Among the controlled entities $\mathcal{V}_t^{\text{ctrl}} \subseteq \mathcal{V}_t$, let $\mathcal{U}_t$ denote the executors that must each be assigned to exactly one selected task instance. Entities outside $\mathcal{U}_t$ may still participate in

Algorithm 1: EXACTSOLVER($\mathcal{T}_t, \mathcal{U}_t, B_t, s_t, F$)

function RECURSIVE(S, C, b)
  $U(S) \leftarrow \bigcup_{\tau \in S} U_\tau$
  $C \leftarrow \{\tau \in C : c_\tau \leq b,\ U_\tau \cap U(S) = \emptyset\}$
  if $U(S) = \mathcal{U}_t$ **then**
    return $(F(S; s_t), S)$
  end if
  $C_u \leftarrow \{\tau \in C : u \in U_\tau\}$ for each $u \in \mathcal{U}_t \setminus U(S)$
  if $C_u = \emptyset$ for some $u \in \mathcal{U}_t \setminus U(S)$ **then**
    return $(-\infty, \emptyset)$
  end if
  if $C_u = \{\tau\}$ for some $u \in \mathcal{U}_t \setminus U(S)$ **then**
    return RECURSIVE($S \cup \{\tau\}, C \setminus \{\sigma : U_\sigma \cap U_\tau \neq \emptyset\}, b - c_\tau$)
  end if
  $u \leftarrow \arg\min_{u \in \mathcal{U}_t \setminus U(S)} |C_u|$; select $\tau \in C_u$
  $(z_{\text{in}}, Q_{\text{in}}) \leftarrow$ RECURSIVE($S \cup \{\tau\}, C \setminus \{\sigma : U_\sigma \cap U_\tau \neq \emptyset\}, b - c_\tau$)

Algorithm 1 (continued)

  $(z_{\text{out}}, Q_{\text{out}}) \leftarrow$ RECURSIVE($S, C \setminus \{\tau\}, b$)
  return $(z_{\text{in}}, Q_{\text{in}})$ if $z_{\text{in}} \geq z_{\text{out}}$; otherwise $(z_{\text{out}}, Q_{\text{out}})$
end function
$(z^\star, S_t^\star) \leftarrow$ RECURSIVE($\emptyset, \mathcal{T}_t, B_t$)
return $S_t^\star$

Algorithm 2: GREEDYSOLVER($\mathcal{T}_t, \mathcal{U}_t, B_t, s_t, \Phi_F$)

$S \leftarrow \emptyset, C \leftarrow \mathcal{T}_t, b \leftarrow B_t$
while $\bigcup_{\tau \in S} U_\tau \neq \mathcal{U}_t$ **do**
  $U(S) \leftarrow \bigcup_{\tau \in S} U_\tau$
  $C \leftarrow \{\tau \in C : c_\tau \leq b,\ U_\tau \cap U(S) = \emptyset\}$
  $\tau^{\text{g}} \leftarrow \operatorname{lex\,max}_{\tau \in C} \Phi_F(\tau \mid S, s_t)$
  $S \leftarrow S \cup \{\tau^{\text{g}}\}$
  $b \leftarrow b - c_{\tau^{\text{g}}}$
  $C \leftarrow C \setminus \{\sigma : U_\sigma \cap U_{\tau^{\text{g}}} \neq \emptyset\}$
end while
return $S_t^{\text{g}} \leftarrow S$

Figure 2: The exact recursive solver (Algorithm 1) and approximate greedy solver (Algorithm 2) for the task-instance-set optimization problem.

task instances but are not subject to exclusive assignment. We consider five task categories, $\mathcal{K} = \{\text{noop}, \text{produce}, \text{harvest}, \text{attack}, \text{retreat}\}$, and define one task schema for each category. For $q \in \mathcal{K}$, schema $\mathsf{M}_q = (\mathcal{A}_q, \Gamma_q, \Psi_q)$ uses $\mathcal{A}_q$ to specify its roles and admissible entity types, Γ_q to specify role capacities, and Ψ_q to determine whether a concrete role assignment is valid in the current state (More schema details are provided in A.2). A schema-valid task instance is $\tau = (q_\tau, e_\tau, U_\tau, \eta_\tau)$, where $e_\tau \subseteq \mathcal{V}_t$ contains all participants, $U_\tau \subseteq e_\tau \cap \mathcal{U}_t$ contains its exclusively assigned executors, and η_τ includes attributes such as resource cost c_τ. All task instances form $\mathcal{T}_t = \{\tau_1, \ldots, \tau_n\}$, and each instance uniquely determines a hyperedge e_τ. Two distinct task instances conflict if they share an executor. This defines the conflict graph $G_t = (\mathcal{T}_t, \mathcal{C}_t)$, where $\mathcal{C}_t = \{\{\tau, \sigma\} : \tau \neq \sigma,\ U_\tau \cap U_\sigma \neq \emptyset\}$, and its independent-set family $\mathcal{I}(G_t) = \{S \subseteq \mathcal{T}_t : U_\tau \cap U_\sigma = \emptyset,\ \forall \tau \neq \sigma \in S\}$.

The feasible family is

$$\mathfrak{F}_t = \left\{ S \in \mathcal{I}(G_t) : \bigcup_{\tau \in S} U_\tau = \mathcal{U}_t, \quad \sum_{\tau \in S} c_\tau \leq B_t \right\}.$$

The objective $F(S; s_t) = V_{\text{local}}(S; s_t) + V_{\text{combat}}(S; s_t) - C_{\text{dist}}(S; s_t) - C_{\text{res}}(S; s_t)$ balances instance-level and joint combat utility against execution-distance and resource-opportunity costs (More objective details are provided in A.3). Hypergraph construction follows

$$S_t^\star \in \arg\max_{S \in \mathfrak{F}_t} F(S; s_t) \quad \Longrightarrow \quad \mathcal{E}_t^\star = \{e_\tau : \tau \in S_t^\star\} \quad \Longrightarrow \quad \mathcal{H}_t^\star = (\mathcal{V}_t, \mathcal{E}_t^\star).$$

3.3.1 EXACT RECURSIVE SOLVER

The exact solver performs a recursive search over states (S, C, b), where S is the selected task-instance set, C is the set of undecided instances, and b is the remaining budget. Given (S, C, b), $\mathcal{R}(S, C, b) = (z, Q)$ returns the highest-valued complete feasible extension $Q \supseteq S$, with $z = F(Q; s_t)$. The search begins with the root call $\mathcal{R}(\emptyset, \mathcal{T}_t, B_t)$. The complete procedure is summarized in Algorithm 1 of Figure 2.

To track assignment coverage, let $U(S) = \bigcup_{\tau \in S} U_\tau$ denote the executors assigned by S. At each recursive call, instances with $c_\tau > b$ are first removed from C. If $U(S) = \mathcal{U}_t$, then S is complete and the solver returns $(F(S; s_t), S)$. If an executor in $\mathcal{U}_t \setminus U(S)$ belongs to no remaining instance, no feasible completion exists and the solver returns $(-\infty, \emptyset)$. If an unassigned executor belongs to exactly one remaining instance, that instance must occur in every feasible completion and is therefore accepted directly. Budget filtering and forced selection are repeated until the state terminates or no forced choice remains. For a remaining nonterminal state, the solver chooses an unassigned

executor with the fewest available task instances and selects one associated instance τ for branching. Accepting τ produces $(S_{\text{in}}, C_{\text{in}}, b_{\text{in}}) = (S \cup \{\tau\}, \{\sigma \in C \setminus \{\tau\} : U_\sigma \cap U_\tau = \emptyset\}, b - c_\tau)$, whereas rejecting it produces $(S_{\text{out}}, C_{\text{out}}, b_{\text{out}}) = (S, C \setminus \{\tau\}, b)$. Let $(z_{\text{in}}, Q_{\text{in}}) = \mathcal{R}(S_{\text{in}}, C_{\text{in}}, b_{\text{in}})$ and $(z_{\text{out}}, Q_{\text{out}}) = \mathcal{R}(S_{\text{out}}, C_{\text{out}}, b_{\text{out}})$. The recursion returns

$$\mathcal{R}(S, C, b) = \begin{cases} (z_{\text{in}}, Q_{\text{in}}), & z_{\text{in}} \geq z_{\text{out}}, \\ (z_{\text{out}}, Q_{\text{out}}), & z_{\text{in}} < z_{\text{out}}. \end{cases}$$

Consequently, the root call returns the optimal solution: $\mathcal{R}(\emptyset, \mathcal{T}_t, B_t) = (F(S_t^\star; s_t), S_t^\star)$.

3.3.2 Approximate Greedy Solver

Because exact search has exponential worst-case complexity, training uses a greedy solver with the same state (S, C, b) and feasibility constraints. At each step, it removes conflicting or unaffordable instances, selects one remaining instance, and retains only its acceptance branch. Each remaining instance is assigned an ordered rule-based score $\Phi_F(\tau \mid S, s_t) = (\phi_1, \ldots, \phi_m)$. Informed by F, its components express task and combat utility, distance cost, and resource opportunity cost through local criteria concerning schema priority, threat, combat strength, distance, resources, and production conditions. The solver selects $\tau^{\text{g}} = \operatorname{lex\,max}_{\tau \in C} \Phi_F(\tau \mid S, s_t)$, where $\operatorname{lex\,max}$ denotes lexicographic maximization: ϕ_1 is maximized first, followed by ϕ_2 among ties, and so forth. It then adds τ^{g} to S, removes it and its conflicting instances from C, and updates $b \leftarrow b - c_{\tau^{\text{g}}}$.

The procedure terminates when $U(S) = \mathcal{U}_t$, yielding $S_t^{\text{g}} \in \mathfrak{F}_t$. Because rejected alternatives are not reconsidered, S_t^{g} need not maximize F. The complete procedure is summarized in Algorithm 2 of Figure 2.

3.4 Task Hypergraph Network

Let $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathbf{H}, \mathbf{X})$ denote the constructed hypergraph, where $\mathcal{V} = \{v_i\}_{i=1}^N$ and $\mathcal{E} = \{e_j\}_{j=1}^M$ are the node and hyperedge sets, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^N$ contains node attributes, and $\mathbf{H} \in \{0,1\}^{N \times M}$ satisfies $H_{ij} = 1$ iff $v_i \in e_j$. The network computes

$$\mathbf{Z}_t = f_{\text{enc}}(\mathcal{H}_t^\star, \mathbf{c}_t), \qquad \boldsymbol{\ell}_t = f_{\text{head}}(\mathbf{Z}_t), \qquad \pi_\theta(\cdot \mid \mathcal{H}_t^\star) = \text{MaskedSoftmax}(\boldsymbol{\ell}_t + \lambda \mathbf{B}_t).$$

Here, $\mathbf{c}_t = \text{Enc}_G(\text{Summary}(\mathcal{H}_t^\star))$ is the global context, $\mathbf{Z}_t = (\mathbf{V}_t^{(L)}, \mathbf{E}_t^{(L)})$ contains the encoded node and hyperedge representations, and $\boldsymbol{\ell}_t$ contains the controlled-entity logits. The matrix $\mathbf{B}_t$ provides soft action preferences, with strength $\lambda \geq 0$, while MaskedSoftmax masks illegal choices and normalizes each action component. The time index is omitted below.

Each hyperedge e_j is induced by a selected task instance τ_j, with attributes $\mathbf{y}_j = \text{Attr}(\tau_j)$. The graph summary $\mathbf{g} = \text{Summary}(\mathcal{H})$ aggregates resource, force, and threat statistics, and $\mathbf{c} = \text{Enc}_G(\mathbf{g})$ encodes this information. To simplify notation, define the globally conditioned initializer $\text{Init}_q(\mathbf{z}, \mathbf{c}) = \text{LN}((1 + \boldsymbol{\gamma}_q(\mathbf{c})) \odot \text{Enc}_q(\mathbf{z}) + \boldsymbol{\beta}_q(\mathbf{c}))$ for $q \in \{V, E\}$, where $\boldsymbol{\gamma}_q$ and $\boldsymbol{\beta}_q$ are learned projections and $\odot$ denotes elementwise multiplication. The initial representations are then $\mathbf{v}_i^{(0)} = \text{Init}_V(\mathbf{x}_i, \mathbf{c})$ and $\mathbf{e}_j^{(0)} = \text{Init}_E(\mathbf{y}_j, \mathbf{c})$.

Let $\mathbf{r}_{ij}$ encode the normalized offsets, Manhattan distance, and local proximity between v_i and the anchor of e_j. At layer l, the node-to-hyperedge and hyperedge-to-node messages are $\mathbf{m}_{i \to j}^{(l)} = \text{MLP}_{V \to E}([\mathbf{v}_i^{(l)}; \mathbf{e}_j^{(l)}; \mathbf{r}_{ij}])$ and $\mathbf{m}_{j \to i}^{(l)} = \text{MLP}_{E \to V}([\mathbf{e}_j^{(l+1)}; \mathbf{v}_i^{(l)}; \mathbf{r}_{ij}])$. For $l = 0, \ldots, L-1$, the representations are updated as $\mathbf{e}_j^{(l+1)} = \text{LN}(\mathbf{e}_j^{(l)} + |e_j|^{-1} \sum_{i:H_{ij}=1} \mathbf{m}_{i \to j}^{(l)})$ and $\mathbf{v}_i^{(l+1)} = \text{LN}(\mathbf{v}_i^{(l)} + d_i^{-1} \sum_{j:H_{ij}=1} \mathbf{m}_{j \to i}^{(l)})$, where $|e_j| = \sum_i H_{ij}$ and $d_i = \sum_j H_{ij}$.

For each controlled entity v_i, the network uses three action heads. The basic head uses $\mathbf{v}_i^{(L)}$ directly. The general encoding $\mathbf{r}_i^g$ describes offsets, Manhattan distance, and proximity to incident-hyperedge anchors, while the attack encoding $\mathbf{r}_i^a$ additionally describes squared Euclidean distance, adjacency, and attack-range proximity. The heads are $\boldsymbol{\ell}_i^{\text{base}} = f_{\text{base}}(\mathbf{v}_i^{(L)})$, $\boldsymbol{\ell}_i^{\text{rel}} = f_{\text{rel}}([\mathbf{v}_i^{(L)}; \mathbf{r}_i^g])$,

and $\boldsymbol{\ell}_i^{\text{attack}} = f_{\text{attack}}([\mathbf{v}_i^{(L)}; \mathbf{r}_i^a])$. They respectively produce logits for action type and harvesting/return directions, movement/production directions and production type, and attack location. Their outputs form $\boldsymbol{\ell}_i \in \mathbb{R}^{d_\mathcal{A}}$. For each node–hyperedge incidence (i, j), let $\mathbf{b}_{ij} \in \{0, 1\}^{d_\mathcal{A}}$ denote the action preference determined by the task attributes, entity role, and entity attributes. Preferences from multiple incident hyperedges are combined as $\mathbf{B}_i = \max_{j:H_{ij}=1} \mathbf{b}_{ij}$, and $\mathbf{B}_t$ stacks the preferences of all controlled entities. The final logits are $\boldsymbol{\ell}_i^{\text{out}} = \boldsymbol{\ell}_i + \lambda \mathbf{B}_i$, which favor task-consistent actions without excluding other legal choices.

3.5 Training

We optimize the policy defined in Section 3.4 together with a value network using PPO. The log-probability of a joint action is obtained by summing the log-probabilities of all action components selected by the controlled entities. The value network reuses $\mathbf{Z}_t = f_{\text{enc}}(\mathcal{H}_t^\star, \mathbf{c}_t)$. Let $\bar{\mathbf{v}}_t = |\mathcal{V}_t|^{-1} \sum_i \mathbf{v}_{t,i}^{(L)}$ and $\bar{\mathbf{e}}_t = |\mathcal{E}_t|^{-1} \sum_j \mathbf{e}_{t,j}^{(L)}$ denote the mean-pooled node and hyperedge representations. The state value is estimated as $V_\varphi(s_t) = f_V([\bar{\mathbf{v}}_t; \bar{\mathbf{e}}_t; \mathbf{c}_t])$, where f_V is a scalar value head. Trajectories collected under the behavior policy $\pi_{\theta_{\text{old}}}$ are used to compute generalized advantage estimates $\widehat{A}_t$ and return targets $\widehat{G}_t$. With probability ratio $\rho_t(\theta) = \pi_\theta(\mathbf{a}_t \mid \mathcal{H}_t^\star)/\pi_{\theta_{\text{old}}}(\mathbf{a}_t \mid \mathcal{H}_t^\star)$, the policy loss is $\mathcal{L}_{\text{policy}} = -\mathbb{E}_t[\min(\rho_t \widehat{A}_t, \text{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\widehat{A}_t)]$, and the value loss is $\mathcal{L}_{\text{value}} = \mathbb{E}_t[(V_\varphi(s_t) - \widehat{G}_t)^2]$. The total loss is $\mathcal{L}_{\text{PPO}} = \mathcal{L}_{\text{policy}} + c_V \mathcal{L}_{\text{value}} - c_H \mathbb{E}_t[\mathbb{H}(\pi_\theta)]$, where ϵ is the clipping parameter, $\mathbb{H}$ denotes policy entropy, and $c_V, c_H \geq 0$ are loss coefficients.

4 Experiment

4.1 Experiment Setting

We conduct experiments in Gym-μRTS, a computationally efficient reinforcement learning interface for full-game μRTS. μRTS is a two-player RTS environment in which both players issue actions simultaneously. All experiments use the fully observable GridNet interface on 16×16 maps. Each state is represented by a $16 \times 16 \times 29$ binary tensor encoding hit points, carried resources, ownership, entity type, current action, and terrain. Supported entities include resources, bases, barracks, workers, and light, heavy, and ranged combat units. At each decision step, the agent outputs a joint action for all currently controlled entities. Each entity action consists of an action type—no-op, move, harvest, return, produce, or attack—and the corresponding parameters, including cardinal directions for movement, harvesting, returning, and production, and a relative target position for attacking. Invalid choices are excluded using the environment-provided legal-action masks. Policies are trained on a single map initialized symmetrically with one base and one worker per player and two nearby resource deposits. We test each policy against four built-in AIs—RandomBiasedAI, NaiveMCTSAI, GuidedRojoA3N, and Rojo—on four held-out maps, denoted A–D. Compared with the training map, Maps A and B retain the initial entity types and counts but use different spatial layouts. Map C replaces each player's worker with a barracks, while Map D adds a barracks to one player's base–worker configuration. Thus, Maps A and B evaluate compositional generalization under spatial-layout changes, whereas Maps C and D evaluate compositional generalization under both spatial-layout and initial entity-type-composition changes (More map details are provided in B).

4.2 Baselines

We compare our approach with representative baselines covering convolutional architectures, reinforcement-learning generalization methods, and relational policy architectures. All methods are trained and evaluated under the same environment settings. The compared methods are described as follows. CNN (Huang et al., 2021) directly encodes the grid-based observation with a convolutional neural network and serves as the basic non-relational baseline. CNN-Reg (Liu et al., 2021) adds L_2 weight regularization to the same architecture. DCPG (Moon et al., 2022) adopts the delayed-critic optimization strategy proposed for reinforcement-learning generalization. Entity Transformer (Zwingenberger, 2023) represents non-empty entities as tokens and employs self-attention with relative-position features to model dense entity interactions. GTG (Jiang et al., 2021) converts grid cells into graph nodes and propagates information through predefined typed spatial relations.

Table 1: Win rates (%) on the training and held-out test maps. Each method plays 40 games on each map. Test WR aggregates all four held-out maps ($n = 160$); Spatial WR aggregates Maps A and B ($n = 80$); Compositional WR aggregates Maps C and D ($n = 80$); and Worst-map WR reports the lowest win rate on a single held-out map ($n = 40$).

METHOD	TRAIN WR	TEST WR	SPATIAL WR	COMP. WR	WORST-MAP WR
CNN	90.0%	43.1%	46.3%	40.0%	15.0%
CNN-Reg	80.0%	43.1%	47.5%	38.8%	25.0%
DCPG	80.0%	32.5%	28.8%	36.3%	25.0%
GTG	92.5%	80.6%	87.5%	73.8%	67.5%
Entity Transformer	87.5%	74.4%	73.8%	75.0%	65.0%
TaskHypergraph-Greedy	**95.0%**	86.9%	90.0%	83.8%	**77.5%**
TaskHypergraph-Recursive	92.5%	**89.4%**	**92.5%**	**86.3%**	75.0%

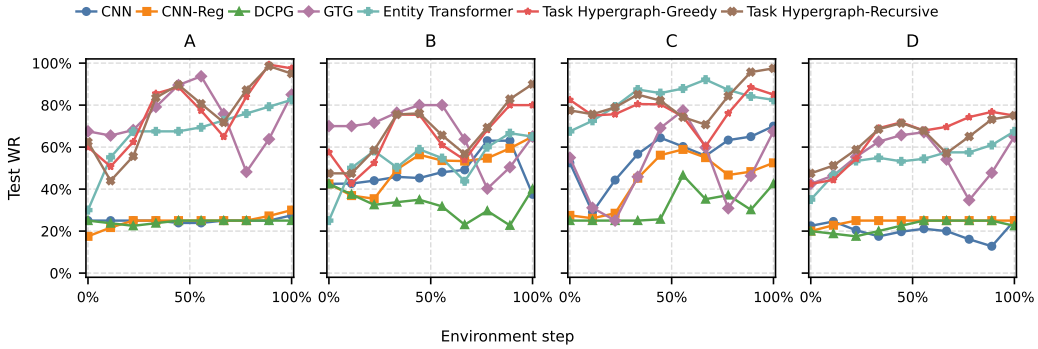

Figure 3: Test win-rate curves of each method on maps A–D. Each policy is trained for 20 million environment steps.

4.3 Main Results

As shown in Table 1, both variants of our method achieve the strongest overall generalization. Task Hypergraph-Greedy obtains an overall test win rate of 86.9%, while Task Hypergraph-Recursive further improves it to 89.4%. The recursive variant also achieves the highest spatial and compositional win rates, at 92.5% and 86.3%, respectively. Meanwhile, the greedy variant attains the highest worst-map win rate of 77.5%. The per-map curves in Figure 3 further show that these improvements extend across different test maps.

Compared with CNN, Task Hypergraph-Greedy improves the overall test win rate from 43.1% to 86.9%, with corresponding gains under spatial changes (90.0% versus 46.3%), compositional changes (83.8% versus 40.0%), and on the worst-performing map (77.5% versus 15.0%). Despite achieving a 90.0% training win rate, CNN generalizes poorly to held-out maps. CNN-Reg and DCPG obtain overall test win rates of only 43.1% and 32.5%, indicating that generic regularization and optimization techniques provide limited benefits for the compositional shifts considered here.

Relational baselines perform better than the convolutional methods: Entity Transformer and GTG achieve overall test win rates of 74.4% and 80.6%, respectively. Nevertheless, Task Hypergraph-Greedy exceeds them by 12.5 and 6.3 percentage points. On compositional maps, it reaches 83.8%, compared with 75.0% for Entity Transformer and 73.8% for GTG. These results suggest that task-semantic higher-order relations are better suited to compositional generalization than dense attention or predefined pairwise relations.

Overall, Table 1 and Figure 3 show that the proposed method consistently outperforms non-relational, generalization-oriented, and relational baselines. The results support task-hypergraph reconstruction as an effective inductive bias for generalization across entity configurations.

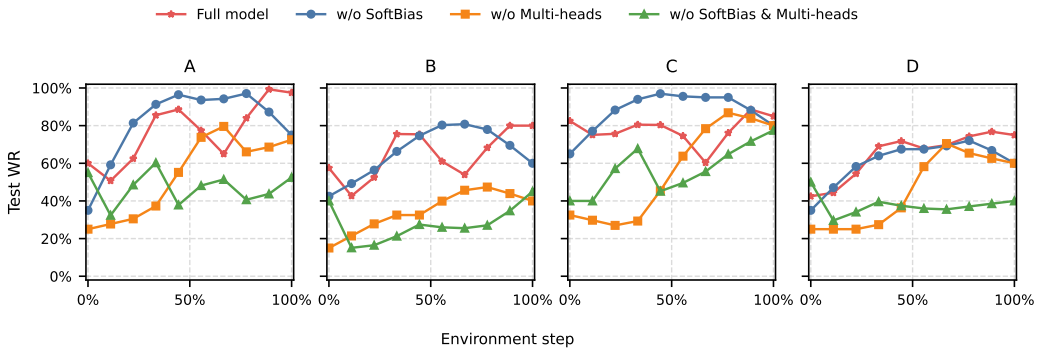

Figure 4: Ablation test win-rate curves for each model variant on Maps A–D. Each variant is trained for 20 million environment steps

4.4 Ablation Studies

The ablation studies compare the two hypergraph solvers and evaluate the contributions of the main network components. As shown in Table 1, both solver variants use the same policy trained with greedy construction and differ only during evaluation. Replacing the greedy solver with the exact recursive solver improves the overall, spatial, and compositional win rates by 2.5 percentage points, reaching 89.4%, 92.5%, and 86.3%, respectively. Thus, exact task-instance selection improves hypergraph construction and evaluation performance without policy retraining. Figure 4 evaluates the specialized action heads with their relation encodings and the hyperedge-type-dependent soft biases, using greedy construction throughout. Removing the soft biases slightly reduces peak performance and causes a larger late-training decline, suggesting that they stabilize task-aligned action selection. Removing the specialized heads and relation encodings produces a greater degradation, demonstrating their importance for mapping hypergraph representations to heterogeneous action components. Removing both yields the weakest performance, indicating that message passing alone is insufficient. Overall, the exact solver improves aggregate performance, while the network ablations show that specialized action generation and hyperedge-type-dependent soft guidance make distinct, complementary contributions to the task-hypergraph policy.

5 Conclusion

This paper studied compositional generalization across initial entity configurations in full-game RTS environments. We proposed a task-hypergraph-based policy architecture that represents entities as nodes and selected task instances as hyperedges. Hypergraph construction was formulated as a constrained optimization problem using shared task schemas, feasibility constraints, and a task-instance-set objective, and solved using an exact recursive solver or an efficient greedy solver. A task hypergraph network then generates entity-level actions through node-to-hyperedge-to-node message passing, specialized action heads, relation encodings, and hyperedge-type-dependent soft biases.

Experiments in Gym-μRTS showed that our architecture outperforms convolutional, generalization-oriented, and relational baselines. Recursive construction improved aggregate performance over greedy construction, while ablations confirmed the importance of specialized heads and the complementary role of soft biases. These results suggest that consistent task-level state decomposition provides an effective inductive bias for generalization across RTS scenarios. The method nevertheless relies on manually designed task schemas and objectives. Future work may explore learned task structures, more efficient solvers, and larger RTS environments.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, LaTeX formatting, and the development and debugging of code for experiments and visualizations. All AI-assisted text was reviewed and revised by the authors, and all generated or modified code was inspected and tested for correctness. The authors take full responsibility for the methodology, experiments, claims, and final content of this work.

REFERENCES

Yunpeng Bai, Chen Gong, Bin Zhang, Guoliang Fan, Xinwen Hou, and Yu Liu. Cooperative multi-agent reinforcement learning with hypergraph convolution. In *International Joint Conference on Neural Networks, IJCNN 2022, Padua, Italy, July 18-23, 2022*, pp. 1–8. IEEE, 2022. doi: 10.1109/IJCNN55064.2022.9891942. URL https://doi.org/10.1109/IJCNN55064.2022.9891942.

Georgios Bakirtzis, Michail Savvas, Ruihan Zhao, Sandeep Chinchali, and Ufuk Topcu. Reduce, reuse, recycle: Categories for compositional reinforcement learning. In Ulle Endriss, Francisco S. Melo, Kerstin Bach, Alberto José Bugarín Diz, Jose Maria Alonso-Moral, Senén Barro, and Fredrik Heintz (eds.), *ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, volume 392 of *Frontiers in Artificial Intelligence and Applications*, pp. 2653–2660. IOS Press, 2024. doi: 10.3233/FAIA240797. URL https://doi.org/10.3233/FAIA240797.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinícius Flores Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Çaglar Gülçehre, H. Francis Song, Andrew J. Ballard, Justin Gilmer, George E. Dahl, Ashish Vaswani, Kelsey R. Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matthew M. Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *CoRR*, abs/1806.01261, 2018. URL http://arxiv.org/abs/1806.01261.

Izzeddin Gur, Natasha Jaques, Yingjie Miao, Jongwook Choi, Manoj Tiwari, Honglak Lee, and Aleksandra Faust. Environment generation for zero-shot compositional reinforcement learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 4157–4169, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/218344619d8fb95d504ccfa11804073f-Abstract.html.

Lei Han, Jiechao Xiong, Peng Sun, Xinghai Sun, Meng Fang, Qingwei Guo, Qiaobo Chen, Tengfei Shi, Hongsheng Yu, and Zhengyou Zhang. Tstarbot-x: An open-sourced and comprehensive study for efficient league training in starcraft II full game. *CoRR*, abs/2011.13729, 2020. URL https://arxiv.org/abs/2011.13729.

Shengyi Huang, Santiago Ontañón, Chris Bamford, and Lukasz Grela. Gym-μrts: Toward affordable full game real-time strategy games research with deep reinforcement learning. *CoRR*, abs/2105.13807, 2021. URL https://arxiv.org/abs/2105.13807.

Zhengyao Jiang, Pasquale Minervini, Minqi Jiang, and Tim Rocktäschel. Grid-to-graph: Flexible spatial relational inductive biases for reinforcement learning. In Frank Dignum, Alessio Lomuscio, Ulle Endriss, and Ann Nowé (eds.), *AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021*, pp. 674–682. ACM, 2021. doi: 10.5555/3463952.3464034. URL https://www.ifaamas.org/Proceedings/aamas2021/pdfs/p674.pdf.

Kishor Jothimurugan, Steve Hsu, Osbert Bastani, and Rajeev Alur. Robust subtask learning for compositional generalization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine*

Learning Research, pp. 15371–15387. PMLR, 2023. URL https://proceedings.mlr.press/v202/jothimurugan23a.html.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *J. Artif. Intell. Res.*, 76:201–264, 2023. doi: 10.1613/JAIR.1.14174. URL https://doi.org/10.1613/jair.1.14174.

Ezgi Korkmaz. A survey analyzing generalization in deep reinforcement learning. *CoRR*, abs/2401.02349, 2024. doi: 10.48550/ARXIV.2401.02349. URL https://doi.org/10.48550/arXiv.2401.02349.

Marcelo Luiz Harry Diniz Lemos, Ronaldo E Silva Vieira, Anderson Rocha Tavares, Leandro Soriano Marcolino, and Luiz Chaimowicz. Enhancing deep reinforcement learning for scale flexibility in real-time strategy games. *Entertain. Comput.*, 52:100843, 2025. doi: 10.1016/J.ENTCOM.2024.100843. URL https://doi.org/10.1016/j.entcom.2024.100843.

Chiqiang Liu and Dazi Li. HYGMA: hypergraph coordination networks with dynamic grouping for multi-agent reinforcement learning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/liu25af.html.

Zhuang Liu, Xuanlin Li, Bingyi Kang, and Trevor Darrell. Regularization matters in policy optimization - an empirical study on continuous control. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=yr1mzrH3IC.

Jorge A. Mendez, Marcel Hussing, Meghna Gummadi, and Eric Eaton. Composuite: A compositional reinforcement learning benchmark. In Sarath Chandar, Razvan Pascanu, and Doina Precup (eds.), *Conference on Lifelong Learning Agents, CoLLAs 2022, 22-24 August 2022, McGill University, Montréal, Québec, Canada*, volume 199 of *Proceedings of Machine Learning Research*, pp. 982–1003. PMLR, 2022. URL https://proceedings.mlr.press/v199/mendez22a.html.

Seungyong Moon, JunYeong Lee, and Hyun Oh Song. Rethinking value function learning for generalization in reinforcement learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/e19ab2dde2e60cf68d1ded18c38938f4-Abstract-Conference.html.

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2661–2670. PMLR, 2017. URL http://proceedings.mlr.press/v70/oh17a.html.

Zhen-Jia Pang, Ruo-Ze Liu, Zhou-Yu Meng, Yi Zhang, Yang Yu, and Tong Lu. On reinforcement learning for full-length game of starcraft. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 4691–4698. AAAI Press, 2019. doi: 10.1609/AAAI.V33I01.33014691. URL https://doi.org/10.1609/aaai.v33i01.33014691.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John P. Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy P. Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. Starcraft II: A new challenge for reinforcement learning. *CoRR*, abs/1708.04782, 2017. URL http://arxiv.org/abs/1708.04782.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Çaglar Gülçehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, 575(7782):350–354, 2019. doi: 10.1038/S41586-019-1724-Z. URL https://doi.org/10.1038/s41586-019-1724-z.

Kang Wang, Zhishu Shen, Zhen Lei, Xianhui Liu, and Tiehua Zhang. Toward multi-agent reinforcement learning based traffic signal control through spatio-temporal hypergraphs. *IEEE Trans. Mob. Comput.*, 24(9):8258–8271, 2025. doi: 10.1109/TMC.2025.3556243. URL https://doi.org/10.1109/TMC.2025.3556243.

Weizheng Wang, Aniket Bera, and Byung-Cheol Min. Hyper-samarl: Hypergraph-based coordinated task allocation and socially-aware navigation for multi-robot systems. *CoRR*, abs/2409.11561, 2024. doi: 10.48550/ARXIV.2409.11561. URL https://doi.org/10.48550/arXiv.2409.11561.

Xiangjun Wang, Junxiao Song, Penghui Qi, Peng Peng, Zhenkun Tang, Wei Zhang, Weimin Li, Xiongjun Pi, Jujie He, Chao Gao, Haitao Long, and Quan Yuan. SCC: an efficient deep reinforcement learning agent mastering the game of starcraft II. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 10905–10915. PMLR, 2021. URL http://proceedings.mlr.press/v139/wang21v.html.

Bin Zhang, Yunpeng Bai, Zhiwei Xu, Dapeng Li, and Guoliang Fan. Efficient policy generation in multi-agent systems via hypergraph neural network. In Mohammad Tanveer, Sonali Agarwal, Seiichi Ozawa, Asif Ekbal, and Adam Jatowt (eds.), *Neural Information Processing - 29th International Conference, ICONIP 2022, Virtual Event, November 22-26, 2022, Proceedings, Part II*, volume 13624 of *Lecture Notes in Computer Science*, pp. 219–230. Springer, 2022. doi: 10.1007/978-3-031-30108-7_19. URL https://doi.org/10.1007/978-3-031-30108-7_19.

Delin Zhao, Yanbo Shan, Chang Liu, Shenghang Lin, Yingxin Shou, and Bin Xu. SDHN: skewness-driven hypergraph networks for enhanced localized multi-robot coordination. *CoRR*, abs/2504.06684, 2025. doi: 10.48550/ARXIV.2504.06684. URL https://doi.org/10.48550/arXiv.2504.06684.

Linfeng Zhao, Lingzhi Kong, Robin Walters, and Lawson L. S. Wong. Toward compositional generalization in object-oriented world modeling. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 26841–26864. PMLR, 2022. URL https://proceedings.mlr.press/v162/zhao22b.html.

Sen Zhao, Wei Wei, Yifan Liu, Ziyang Wang, Wendi Li, Xian-Ling Mao, Shuai Zhu, Minghui Yang, and Zujie Wen. Towards hierarchical policy learning for conversational recommendation with hypergraph-based reinforcement learning. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pp. 2459–2467. ijcai.org, 2023. doi: 10.24963/IJCAI.2023/273. URL https://doi.org/10.24963/ijcai.2023/273.

Allan Zhou, Vikash Kumar, Chelsea Finn, and Aravind Rajeswaran. Policy architectures for compositional generalization in control. *RLJ*, 5:2264–2283, 2024. URL https://rlj.cs.umass.edu/2024/papers/Paper327.html.

Niklas Zwingenberger. Transformers as policies for variable action environments. *CoRR*, abs/2301.03679, 2023. doi: 10.48550/ARXIV.2301.03679. URL https://doi.org/10.48550/arXiv.2301.03679.

A TASK-SCHEMA AND OBJECTIVE DEFINITIONS

This appendix provides the concrete task schemas and the complete task-instance-set objective used for task hypergraph construction.

A.1 NOTATION

Let $\mathcal{I}_t \subseteq \mathcal{V}_t^{\mathrm{ctrl}}$ denote the currently actionable controlled entities. For compactness, define $\mathcal{W}_t$, $\mathcal{B}_t$, $\mathcal{D}_t$, $\mathcal{C}_t$, and $\mathcal{R}_t$ as the sets of controlled workers, bases, barracks, combat units, and neutral resources, respectively. Let $\mathcal{Y}_t$ denote the opponent entities and $\mathcal{Y}_t^{\mathrm{atk}} \subseteq \mathcal{Y}_t$ the opponent workers and combat units. The controlled attack-capable entities are

$$\mathcal{Q}_t = \mathcal{W}_t \cup \mathcal{C}_t.$$

For a role assignment e, e^r denotes the entities assigned to role r. The predicate $\mathrm{Idle}(v; s_t)$ indicates that v is currently actionable, while $\mathrm{Hom}(A)$ indicates that all entities in A have the same type. The maximum size of a homogeneous entity group is

$$m(a) = \begin{cases} 5, & a = \text{worker}, \\ 3, & a \in \{\text{light}, \text{heavy}, \text{ranged}\}, \\ 1, & a \in \{\text{base}, \text{barrack}, \text{resource}\}. \end{cases}$$

Let $d(A, B)$ denote the minimum Manhattan distance between two nonempty entity sets and $d_c(A, B)$ the Manhattan distance between their centers. For singleton sets, the braces are omitted. Finally, $\mathrm{Reach}(v_w, v_r, v_b; s_t)$ indicates that a valid worker–resource–base route exists in state s_t.

For every task category q, the schema $\mathsf{M}_q = (\mathcal{A}_q, \Gamma_q, \Psi_q)$ specifies its roles and admissible entity types, role-capacity condition, and state-dependent validity condition, respectively.

A.2 TASK SCHEMAS

No-op schema. The no-op schema assigns an actionable controlled entity to an explicit fallback task:

$$\mathcal{A}_{\mathrm{noop}} = \{\mathrm{idle} : \mathcal{I}_t\},$$
$$\Gamma_{\mathrm{noop}}(e) = \mathbb{I}[|e^{\mathrm{idle}}| = 1],$$
$$\Psi_{\mathrm{noop}}(e; s_t) = \mathbb{I}\left[\forall v \in e^{\mathrm{idle}}, \mathrm{Idle}(v; s_t)\right].$$

The exclusive executor set and resource cost are

$$U_\tau = e^{\mathrm{idle}}, \qquad c_\tau = 0.$$

Production schema. The production category contains barracks construction, worker production, and combat-unit production. Barracks construction uses

$$\mathcal{A}_{\mathrm{build}} = \{\mathrm{builder} : \mathcal{W}_t\},$$
$$\Gamma_{\mathrm{build}}(e) = \mathbb{I}[|e^{\mathrm{builder}}| = 1].$$

Writing $e^{\mathrm{builder}} = \{v_b\}$, its validity condition is

$$\Psi_{\mathrm{build}}(e; s_t) = \mathbb{I}\big[\mathrm{Idle}(v_b; s_t) \wedge N_t^{\mathrm{base}} \geq 1 \\ \wedge\ N_t^{\mathrm{barrack}} + N_t^{\mathrm{progress}} < 2 \wedge B_t \geq 5\big].$$

Unit production uses

$$\mathcal{A}_{\text{unit-produce}} = \{\mathrm{producer} : \mathcal{B}_t \cup \mathcal{D}_t\},$$
$$\Gamma_{\text{unit-produce}}(e) = \mathbb{I}[|e^{\mathrm{producer}}| = 1].$$

Writing $e^{\mathrm{producer}} = \{v_p\}$, worker-production validity is

$$\Psi_{\mathrm{worker}}(e; s_t) = \mathbb{I}[v_p \in \mathcal{B}_t \wedge \mathrm{Idle}(v_p; s_t) \wedge B_t \geq 1],$$

and combat-unit-production validity is

$$\Psi_{\text{combat}}(e; s_t) = \mathbb{I}[v_p \in \mathcal{D}_t \wedge \text{Idle}(v_p; s_t) \wedge B_t \geq 2]\,.$$

Let $\eta_\tau \in \{\text{barrack}, \text{worker}, \text{combat}\}$ denote the production intent. The complete production schema is the union of these three variants:

$$\mathsf{M}_{\text{produce}} = \mathsf{M}_{\text{build}} \cup \mathsf{M}_{\text{worker}} \cup \mathsf{M}_{\text{combat}}.$$

Its executor set and resource cost are

$$(U_\tau, c_\tau) = \begin{cases} (e^{\text{builder}}, 5), & \eta_\tau = \text{barrack}, \\ (e^{\text{producer}}, 1), & \eta_\tau = \text{worker}, \\ (e^{\text{producer}}, 2), & \eta_\tau = \text{combat}. \end{cases}$$

Harvesting schema. The harvesting schema contains a worker, a resource, and a return base:

$$\mathcal{A}_{\text{harvest}} = \{\text{worker} : \mathcal{W}_t,\ \text{resource} : \mathcal{R}_t,\ \text{return-base} : \mathcal{B}_t\}.$$

Its capacity condition is

$$\Gamma_{\text{harvest}}(e) = \mathbb{I}\big[|e^{\text{worker}}| = |e^{\text{resource}}| = |e^{\text{return-base}}| = 1\big]\,.$$

Let $e^{\text{worker}} = \{v_w\}$, $e^{\text{resource}} = \{v_r\}$, and $e^{\text{return-base}} = \{v_b\}$. The validity condition is

$$\begin{aligned} \Psi_{\text{harvest}}(e; s_t) = \mathbb{I}\big[\,&\text{Idle}(v_w; s_t) \wedge d(v_r, v_b) \leq 8 \\ &\wedge\ v_b \in \arg\min_{b \in \mathcal{B}_t} d(v_r, b) \\ &\wedge\ \text{Reach}(v_w, v_r, v_b; s_t)\big]. \end{aligned}$$

Only the worker is exclusively assigned:

$$U_\tau = e^{\text{worker}}, \qquad c_\tau = 0.$$

To control the candidate-set size, at most the two nearest reachable workers are retained as separate instances for each resource. This is a candidate-pruning rule rather than part of Γ_{harvest} or Ψ_{harvest}.

Attacking schema. The attacking schema contains an attacker group and an opponent target group:

$$\mathcal{A}_{\text{attack}} = \begin{cases} \text{attacker} : \mathcal{Q}_t, \\ \text{target} : \mathcal{Y}_t. \end{cases}$$

Its capacity condition is

$$\begin{aligned} \Gamma_{\text{attack}}(e) = \mathbb{I}\big[\,&\text{Hom}(e^{\text{attacker}}) \wedge \text{Hom}(e^{\text{target}}) \\ &\wedge\ 1 \leq |e^{\text{attacker}}| \leq m(\text{type}(e^{\text{attacker}})) \\ &\wedge\ 1 \leq |e^{\text{target}}| \leq m(\text{type}(e^{\text{target}}))\big]. \end{aligned}$$

The state-dependent validity condition is

$$\begin{aligned} \Psi_{\text{attack}}(e; s_t) = \mathbb{I}\big[&e^{\text{attacker}} \subseteq \mathcal{Q}_t \cap \mathcal{I}_t \\ &\wedge\ e^{\text{target}} \subseteq \mathcal{Y}_t\big]. \end{aligned}$$

The attacker group is exclusive, whereas the target group may be shared:

$$U_\tau = e^{\text{attacker}}, \qquad c_\tau = 0.$$

Each attack instance additionally records the group-center distance $d_\tau = d_c(e^{\text{attacker}}, e^{\text{target}})$ and a binary defense-intent attribute indicating whether its target lies within eight cells of a controlled base.

Retreating schema. The retreating schema contains a controlled retreater group and an opponent threat group:

$$\mathcal{A}_{\text{retreat}} = \begin{cases} \text{retreater} : \mathcal{Q}_t, \\ \text{threat} : \mathcal{Y}_t^{\text{atk}}. \end{cases}$$

Its capacity condition is

$$\begin{aligned} \Gamma_{\text{retreat}}(e) = \mathbb{I}\big[& \operatorname{Hom}(e^{\text{retreater}}) \wedge \operatorname{Hom}(e^{\text{threat}}) \\ & \wedge\ 1 \le |e^{\text{retreater}}| \le m(\text{type}(e^{\text{retreater}})) \\ & \wedge\ 1 \le |e^{\text{threat}}| \le m(\text{type}(e^{\text{threat}}))\big]. \end{aligned}$$

The state-dependent validity condition is

$$\begin{aligned} \Psi_{\text{retreat}}(e; s_t) = \mathbb{I}\big[& e^{\text{retreater}} \subseteq \mathcal{Q}_t \cap \mathcal{I}_t \\ & \wedge\ e^{\text{threat}} \subseteq \mathcal{Y}_t^{\text{atk}} \\ & \wedge\ d_c(e^{\text{retreater}}, e^{\text{threat}}) \le 10\big]. \end{aligned}$$

The retreater group is exclusive, whereas the threat group is nonexclusive:

$$U_\tau = e^{\text{retreater}}, \qquad c_\tau = 0.$$

Unified task-instance set. The state-local task-instance set is

$$\begin{aligned} \mathcal{T}_t = \bigcup_{q \in \mathcal{K}} \big\{ & \tau = (q, e_\tau, U_\tau, \eta_\tau) : \\ & \mathcal{A}_q(e_\tau) = 1,\ \Gamma_q(e_\tau) = 1,\ \Psi_q(e_\tau; s_t) = 1 \big\}. \end{aligned}$$

Here, $\mathcal{A}_q(e_\tau) = 1$ means that every role is assigned entities of admissible types. The predicates Γ_q and Ψ_q describe only instance-level validity. Executor conflicts, complete executor coverage, and the shared resource budget are imposed separately through the feasible family $\mathfrak{F}_t$.

A.3 Task-Instance-Set Objective

For a feasible task-instance set $S \in \mathfrak{F}_t$, the objective is

$$\begin{aligned} F(S; s_t) = & V_{\text{local}}(S; s_t) + V_{\text{combat}}(S; s_t) \\ & - C_{\text{dist}}(S; s_t) - C_{\text{res}}(S; s_t). \end{aligned}$$

State-dependent demand. Let P_t^{own} and P_t^{enemy} denote the total combat power of the two players. The military and economic demand terms are

$$D_t^{\text{mil}} = \sigma\left(\frac{P_t^{\text{enemy}} - P_t^{\text{own}}}{4} \right), \qquad D_t^{\text{eco}} = \frac{1}{1 + B_t/5},$$

where $\sigma(x) = (1 + \exp(-x))^{-1}$.

The target number of workers is

$$N_t^{\text{worker}\star} = \max\big\{1,\ \min\{6,\ 2N_t^{\text{base}} + N_t^{\text{near-res}}\}\big\},$$

where $N_t^{\text{near-res}}$ is the number of resources within eight cells of a controlled base.

Local task utility. The local task utility is additive:

$$V_{\text{local}}(S; s_t) = \sum_{\tau \in S} u_{\text{local}}(\tau; s_t).$$

For a no-op instance,

$$u_{\text{local}}(\tau; s_t) = 0.$$

For a harvesting instance with route length L_τ,

$$u_{\text{local}}(\tau; s_t) = 2D_t^{\text{eco}} \exp(-L_\tau/10).$$

For worker production,

$$u_{\text{local}}(\tau; s_t) = 2\max\left\{0,\, 1 - \frac{N_t^{\text{worker}}}{N_t^{\text{worker}\star}}\right\}.$$

For barracks construction,

$$u_{\text{local}}(\tau; s_t) = \frac{4(0.3 + 0.7D_t^{\text{mil}})}{1 + N_t^{\text{barrack}}}.$$

For combat-unit production,

$$u_{\text{local}}(\tau; s_t) = 2.5(0.5 + 0.5D_t^{\text{mil}}).$$

For a retreat instance, let R_τ and Y_τ denote its retreater and threat groups. Its combat disadvantage is

$$\Delta_\tau = \max\left\{0,\, \min\left\{1,\, \frac{P(Y_\tau) - P(R_\tau)}{P(R_\tau) + \varepsilon}\right\}\right\},$$

and its local utility is

$$u_{\text{local}}(\tau; s_t) = 2\Delta_\tau \exp\left(-\frac{d_c(R_\tau, Y_\tau)}{8}\right).$$

Attack instances are evaluated jointly through V_{combat} rather than individually through V_{local}.

Joint combat utility. Let $\mathcal{Y}(S)$ be the distinct target groups induced by the attack instances in S. For $Y \in \mathcal{Y}(S)$, let $A_Y(S)$ be the union of the attackers assigned to Y, and let $d_Y^{\max}$ be their maximum recorded attacker–target distance.

The required combat power for target Y is

$$\begin{aligned} P_{\text{req}}(Y) = P(Y) &+ P_{\text{sup}}(Y, d_Y^{\max}) \\ &+ 2\mathbb{I}_{\text{base}}(Y) + \mathbb{I}_{\text{barrack}}(Y) \\ &+ 1.5\left\lceil \frac{\max\{0, d_Y^{\max} - 10\}}{10} \right\rceil. \end{aligned}$$

Here, $P_{\text{sup}}(Y, d)$ is the total power of other opponent attack-capable entities within distance d of Y, while $\mathbb{I}_{\text{base}}(Y)$ and $\mathbb{I}_{\text{barrack}}(Y)$ indicate whether Y contains a base or barracks.

The attacker-to-requirement ratio is

$$r_Y = \frac{P(A_Y(S))}{\max\{P_{\text{req}}(Y), \varepsilon\}}.$$

Its power-sufficiency value is

$$h(r_Y) = \begin{cases} -2(1 - r_Y), & r_Y < 1, \\ 1 - 0.25(r_Y - 1)^2, & r_Y \geq 1. \end{cases}$$

Let

$$u_Y = \exp\left(-\frac{d(Y, \mathcal{B}_t)}{8}\right)$$

denote the urgency of target Y relative to the controlled bases. The response factor is

$$g_Y = \begin{cases} \exp(-d_Y^{\max}/4), & Y \text{ is a defensive target}, \\ 0, & \text{otherwise}. \end{cases}$$

The combat value of target Y is then

$$v_{\text{combat}}(Y; S, s_t) = A(Y)\,(1 + 2u_Y + 2u_Y g_Y)\,h(r_Y).$$

Let $\mathcal{Y}_t^{\text{threat}}$ contain the attack-capable opponent entities within eight cells of a controlled base. A threat is covered when it is targeted by a defensive attack satisfying

$$r_Y \geq 1, \qquad d_Y^{\max} \leq 8.$$

If the currently available controlled combat power is sufficient to respond to nearby threats, leaving threat y uncovered incurs

$$p_{\text{ignore}}(y) = 3A(\{y\}) \left[1 + \exp\left(-\frac{d(\{y\}, \mathcal{B}_t)}{8}\right)\right].$$

Therefore,

$$V_{\text{combat}}(S; s_t) = \sum_{Y \in \mathcal{Y}(S)} v_{\text{combat}}(Y; S, s_t) - \sum_{y \in \mathcal{Y}_t^{\text{uncov}}(S)} p_{\text{ignore}}(y),$$

where $\mathcal{Y}_t^{\text{uncov}}(S)$ is the set of uncovered nearby threats. The second term is zero when the available controlled power is insufficient to respond.

Attack-distance cost. The distance term penalizes the travel distance associated with selected attack instances:

$$C_{\text{dist}}(S; s_t) = \frac{0.5}{30} \sum_{\substack{\tau \in S \\ q_\tau = \text{attack}}} |U_\tau|\, d_\tau.$$

Resource-opportunity cost. The resource cost discourages unnecessary use of the available budget:

$$C_{\text{res}}(S; s_t) = 0.3 \frac{\sum_{\tau \in S} c_\tau}{B_t + 1}.$$

Combat-power and asset values. The combat-power function $P(\cdot)$ uses type-specific values

$$P_{\text{worker}} = 1, \qquad P_{\text{light}} = P_{\text{heavy}} = P_{\text{ranged}} = 3,$$

with zero combat power assigned to bases and barracks. The asset-value function uses

$$A_{\text{worker}} = 1, \qquad A_{\text{light}} = A_{\text{heavy}} = A_{\text{ranged}} = 3,$$
$$A_{\text{base}} = 12, \qquad A_{\text{barrack}} = 6.$$

All coefficients, capacities, distance thresholds, and type-specific values are fixed implementation hyperparameters shared across training and test scenarios.

B Training and Test Map Configurations

All experiments use 16×16 maps with empty terrain. Grid coordinates are zero-indexed and written as (x, y), with $(0, 0)$ at the upper-left corner. The two players are denoted by P0 and P1. Each player starts with five resources in its resource bank on every map. Neutral resource deposits are denoted by R, and the value following each resource coordinate indicates the amount stored in that deposit.

The training map is `basesWorkers16x16A`. The four held-out test maps, denoted by A–D, correspond to `basesWorkers16x16B`, `basesWorkers16x16L`, `self`, and `self2`, respectively. Table 2 summarizes their initial entity compositions, while Table 3 reports the exact initial positions.

Relative to the training map, Map A introduces a small displacement of all initial entities, whereas Map B produces a larger change in their spatial arrangement. Both maps preserve the initial entity types and counts. Map C changes the layout and replaces each player's initial worker with a barracks. Map D uses the spatial layout of Map A but adds a barracks for P1. Thus, Maps A and B evaluate compositional generalization induced by changes in relative entity positions, whereas Maps C and D additionally evaluate changes in the initial entity-type composition.

Although barracks do not occur in the initial state of the training map, they can be constructed during training episodes. Therefore, the held-out maps introduce neither unseen entity types nor new game mechanics; they instead contain unseen initial configurations of familiar entities.

Table 2: Initial configurations of the training and held-out test maps. Entity counts are reported separately for P0 and P1. Every map contains four neutral resource deposits, and both players start with five resources in their resource banks.

MAP	P0 ENTITIES	P1 ENTITIES	DEPOSITS	VALUE/DEPOSIT
Training	1 Base, 1 Worker	1 Base, 1 Worker	4	25
A	1 Base, 1 Worker	1 Base, 1 Worker	4	25
B	1 Base, 1 Worker	1 Base, 1 Worker	4	25
C	1 Base, 1 Barracks	1 Base, 1 Barracks	4	20
D	1 Base, 1 Worker	1 Base, 1 Worker, 1 Barracks	4	25

Table 3: Exact initial entity positions on the training and held-out test maps. B, W, D, and R denote a base, worker, barracks, and neutral resource deposit, respectively. Each resource is written as $(x, y; z)$, where z is its initial stored-resource amount.

MAP	P0	P1	NEUTRAL RESOURCES
Training	$B : (2, 2)$ $W : (1, 1)$	$B : (13, 13)$ $W : (14, 14)$	$R : (0, 0; 25), (0, 1; 25)$ $(15, 14; 25), (15, 15; 25)$
A	$B : (2, 3)$ $W : (1, 2)$	$B : (13, 12)$ $W : (14, 13)$	$R : (0, 1; 25), (0, 2; 25)$ $(15, 13; 25), (15, 14; 25)$
B	$B : (2, 4)$ $W : (1, 5)$	$B : (13, 11)$ $W : (14, 10)$	$R : (0, 4; 25), (0, 5; 25)$ $(15, 10; 25), (15, 11; 25)$
C	$B : (3, 1)$ $D : (3, 5)$	$B : (12, 14)$ $D : (12, 10)$	$R : (0, 0; 20), (1, 0; 20)$ $(14, 15; 20), (15, 15; 20)$
D	$B : (2, 3)$ $W : (1, 2)$	$B : (13, 12)$ $W : (14, 13)$ $D : (10, 10)$	$R : (0, 1; 25), (0, 2; 25)$ $(15, 13; 25), (15, 14; 25)$