A Convergence Analysis of the Distributed Priority Auction

This appendix provides a formal analysis of the convergence properties of the distributed auction mechanism used for dynamic priority scheduling in ADAPT, as described in Subsection 3.2. We model the auction within a game-theoretic framework and leverage established results from auction theory to demonstrate finite-time convergence under ideal communication assumptions.

A.1 Auction Formulation as a Strategic Game

At each timestep t, the priority scheduling process can be modeled as a strategic game:

- **Players:** The set of agents $\mathcal{A} = \{1, 2, \ldots, N\}$.
- **Actions:** In each iteration of the auction, agent i computes potential bids for priority slots $d \in \{1, \ldots, N\}$. The core action is submitting the bid p_{i,d_i^*} for its preferred slot d_i^*, computed via Eq. (10).
- **Utilities:** Each agent i seeks to secure a priority slot d that maximizes its net utility. The utility $v_t^i[d]$ for obtaining slot d is predicted by the scoring network θ_S (Eq. (8)). If agent i wins slot d_i^* at price $p_{d_i^*}$, its net utility is:

$$u_i(\text{assignment}) = v_t^i[d_i^*] - p_{d_i^*}. \tag{20}$$

Agents implicitly aim to maximize this net utility through the bidding protocol defined by Eq. (9) and Eq. (10).

A.2 Theoretical Foundation and Convergence Guarantee

The distributed auction mechanism in ADAPT builds on the discrete bidding framework introduced by Deng et al. [2], which establishes finite-time convergence under mild conditions. Our formulation satisfies these conditions, enabling a formal guarantee of stability and convergence for the priority scheduling process.

Proposition 1 (Finite-Time Convergence of the Priority Auction in ADAPT)**.** *Let the agent-slot utilities $v_t^i[d]$ be bounded for all agents $i \in \{1, \ldots, N\}$ and slots $d \in \{1, \ldots, N\}$, and let the bid increment $\delta > 0$ be fixed and positive. Then, the distributed priority auction mechanism used in ADAPT converges to a stable slot assignment $\mathbf{P}_t$ in a finite number of iterations.*

Proof. The proof proceeds by establishing the monotonic increase of prices, the boundedness of utility ranges, and the stability of the resulting assignment.

Monotonicity of Slot Prices. Let $p_d(k)$ denote the price of slot d at iteration k. At each step, agent i bids for its most preferred slot d_i^*, submitting $p_{i,d_i^*} = v_t^i[d_i^*] - \widetilde{u}_t^i + \delta$, where $\widetilde{u}_t^i$ is the utility from the second-best available slot given current prices (see Eq. (9)). Upon receiving the slot, the price p_d is updated to the bid value p_{i,d_i^*}. This ensures that $p_d(k+1) \geq p_d(k) + \delta$ whenever the slot assignment changes. Thus, the price sequence for each slot is monotonically non-decreasing, with strict increases during competitive bidding.

Bounded Progress. The utilities $v_t^i[d]$ are assumed to be bounded due to the finite range of the scoring network θ_S. Since prices increase in discrete increments of at least δ, and are upper-bounded by the maximum utility differences, the number of price updates for any slot is finite. As there are N agents and N slots, the total number of such updates across the system is finite.

Stability of Final Assignment. The auction terminates when no agent can improve its net utility by submitting a higher bid for any slot. At convergence, each agent is assigned a slot such that deviating (i.e., outbidding another agent) would yield no net utility gain under the prevailing price vector. This constitutes a stable assignment under the auction dynamics.

Therefore, the iterative bidding process converges to a stable priority allocation in a finite number of steps. □

Remark. This convergence guarantee holds under the assumption of synchronized and reliable message exchange. Extension to asynchronous or lossy communication settings remains a subject of future investigation.

A.3 Complexity Analysis

The number of iterations required for convergence in the distributed auction mechanism depends primarily on the utility range and the bid increment δ. While the exact complexity is influenced by implementation specifics such as tie-breaking and communication ordering, existing analyses of discrete auction protocols [2] suggest a worst-case iteration bound of $\mathcal{O}(N \cdot V_{\max}/\delta)$, where $V_{\max}$ denotes the maximum possible utility difference across all agents and slots. This bound arises from the fact that slot prices increase monotonically in discrete steps and are upper-bounded due to the assumed boundedness of agent utilities.

A.4 Practical Implications and Robustness

The convergence guarantee provides theoretical support for the stability and reliability of ADAPT's dynamic priority scheduler. The bidding increment δ introduces a practical trade-off between assignment precision and convergence speed:

- Smaller values of δ enable finer-grained prioritization but may increase the number of auction rounds.
- Larger values reduce the number of iterations but can lead to coarser or less discriminative slot assignments.

In practice, the auction premium δ is treated as a tunable hyperparameter. As shown in Table 3, a moderate value of $\delta = 0.5$ yields the best trade-off between convergence efficiency and coordination precision. This setting provides sufficient bidding differentiation while avoiding excessive slot reassignments, contributing to stable and effective execution ordering.

Moreover, the theoretical assumptions underlying Proposition 1 are met in our implementation. Specifically, agent-slot utility scores $v_t^i[d]$ are outputs of a neural scoring network θ_S with bounded activations, ensuring that utility values remain finite. The bid increment δ is explicitly fixed and positive throughout training and evaluation. These conditions guarantee monotonic price increases and bounded total bidding rounds, ensuring convergence in finite time. Empirically, we observe stable convergence behavior across tasks, further validating these assumptions in practice.

B Comparative Analysis of Auction Mechanisms

We further examine how ADAPT's distributed ϵ-increment auction compares to a Sequential Vickrey auction. The latter is often used to guarantee incentive compatibility, but its application to priority scheduling raises efficiency and scalability concerns. Assigning a

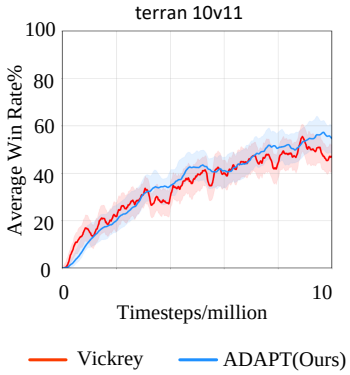


Figure B.1. Comparison of ADAPT's distributed auction with a Sequential Vickrey auction on the `Terran_10_vs_11` map. Metrics include win rate and convergence time (in millions of timesteps).

complete ordering among N agents requires either a full VCG mechanism, which entails solving N assignment problems at cost $\Theta(N^4)$ using the Hungarian algorithm, or N sequential single-item auctions. The sequential variant is simpler but depends on a centralized auctioneer, creating both a communication bottleneck and a single point of failure.

Table B.1. Average message size per timestep (MesB, in bytes) on the `Terran_10_vs_11` map.

Method	MesB (Bytes)
ADAPT (Ours)	**8,837**
Sequential Vickrey	16,240

In practice, the sequential procedure operates iteratively: at each round, all agents submit bids, the highest bidder is assigned the current slot, and the winner is removed before the next round begins. Payments correspond to the second-highest bid, though they are only conceptual in our setting since no explicit economic transfers are made. While this ensures theoretical incentive compatibility, it provides no real advantage here because ADAPT's bids are deterministic functions of shared dependency vectors and thus not subject to strategic misreporting.

Empirically, ADAPT achieves higher coordination quality with substantially lower cost. As reported in Figure B.1 and Table B.1, Sequential Vickrey lags behind by 4.85 win-rate points and incurs nearly double the communication overhead (+83.8%). Moreover, its reliance on a central auctioneer introduces additional latency and vulnerability, whereas ADAPT's decentralized bidding scheme avoids both bottlenecks and single points of failure. These findings highlight that ADAPT's distributed auction not only matches the theoretical robustness of Vickrey but also surpasses it in efficiency and practicality for real-time multi-agent scheduling.

C Empirical Analysis of Co-Adaptation Stability

To ensure stable training in ADAPT, we explicitly manage gradient flow across the five core loss functions: observation encoding ($\mathcal{L}_{\text{E}}$), message generation ($\mathcal{L}_{\text{G}}$), priority scoring ($\mathcal{L}_{\text{S}}$), observation reconstruction ($\mathcal{L}_{\text{R}}$), and policy learning ($\mathcal{L}_{\text{D}}$). The encoding loss $\mathcal{L}_{\text{E}}$ updates the encoder ϕ and the value head. The message generation loss $\mathcal{L}_{\text{G}}$ updates the message generator θ_G and the variational network ξ, without affecting ϕ. The priority scoring loss $\mathcal{L}_{\text{S}}$ trains the scorer θ_S independently from the target assignment $\mathbf{P}_t^*$. The reconstruction loss $\mathcal{L}_{\text{R}}$ updates the reconstruction module θ_R without propagating gradients to either θ_G or ϕ. Lastly, the policy loss $\mathcal{L}_{\text{D}}$ updates the policy network θ, while blocking gradients to θ_R, V_ϕ, and θ_S. This structured design ensures decoupled training among modules, enabling convergence without cross-interference.

Although the message generation and priority scoring components influence one another during training, where message content affects dependency estimation and dependencies determine priority ordering, this interaction does not form a circular dependency. Instead, the two modules co-adapt in a stable manner throughout learning. We present empirical results to confirm that the optimization process remains well-behaved and free from degenerative feedback loops.

We focus this analysis on the `Terran_10_vs_11` scenario from the SMACv2 benchmark. This task poses asymmetric and dynamic challenges, including numerical disadvantage, diverse unit types, and shifting tactical configurations. It is widely recognized as a rigorous benchmark for testing the stability of learning-based coordination strategies in multi-agent systems.

C.1 Step 1: Monitoring Module-Specific Loss Functions

To evaluate the stability of ADAPT's joint optimization, we track the evolution of each module's loss function over 10 million timesteps in the `Terran_10_vs_11` scenario. The following components are monitored:

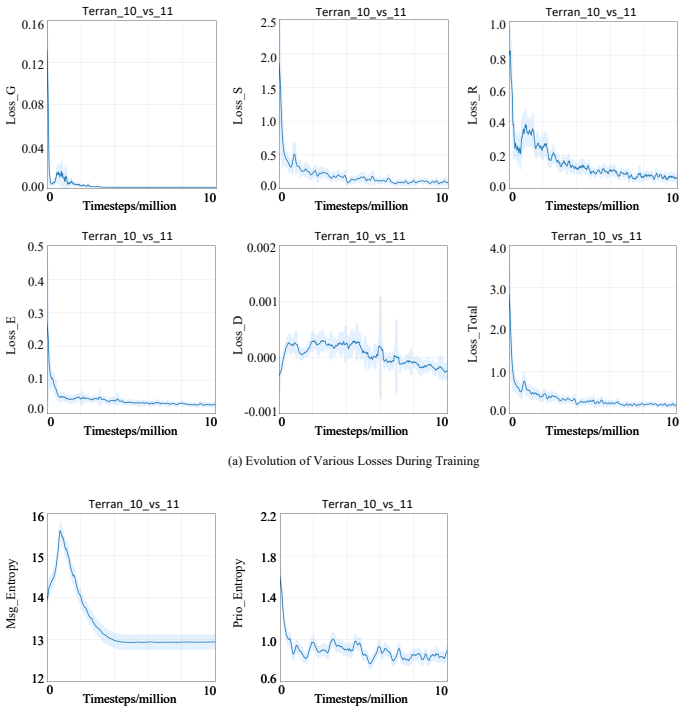
- $\mathcal{L}_{\text{E}}$: The encoder and value head loss, computed using temporal-difference (TD) learning. This term stabilizes the representation space and supports value estimation.
- $\mathcal{L}_{\text{G}}$: The message generator and variational module loss, which maximizes mutual information between the generated message c_t^i and the subsequent action a_t^i, conditioned on the encoded observation $\widehat{o}_t^i$.
- $\mathcal{L}_{\text{S}}$: The priority scoring loss, optimized via cross-entropy to align predicted priority distributions with dependency-aware reference assignments.
- $\mathcal{L}_{\text{R}}$: The reconstruction loss, measured by the mean squared error between original and reconstructed observation encodings.
- $\mathcal{L}_{\text{D}}$: The policy loss, computed using Proximal Policy Optimization (PPO) with a clipped surrogate objective to ensure stable policy updates.
- $\mathcal{L}$: The total objective, aggregating all module-specific losses.

Figure C.2(a) presents the mean loss trajectories, accompanied by standard deviations across five random seeds. All loss terms show consistent convergence trends, with no indication of instability or interference across modules. Notably, $\mathcal{L}_{\text{D}}$ decreases into the negative range, consistent with the optimization of PPO's clipped objective. Its deviation remains low (less than 0.04), reflecting stable policy learning across runs. Although $\mathcal{L}_{\text{S}}$ exhibits mild early-stage fluctuations (within the first 2 million timesteps), these are quickly dampened as the scoring network adapts to the evolving dependency structures characteristic of asymmetric Terran combat.

C.2 Step 2: Monitoring Entropy Dynamics

To further assess co-adaptation, we track the entropy of message generation and priority assignment to evaluate convergence and stability:

- **Message Entropy:** The Shannon entropy of message vectors c_t^i, estimated over a batch of 800 agents. This measures the diversity of communication content. Higher entropy suggests more ex-



(a) Evolution of Various Losses During Training

(b) Evolution of Message and Priority Entropy During Training

Figure C.2. Training dynamics on `Terran_10_vs_11`. Subfigure (a) shows the convergence of individual loss terms. Subfigure (b) reports entropy stabilization of message generation and priority scoring. All modules exhibit smooth co-adaptation with low variance.

pressive messaging, while lower entropy indicates convergence to more deterministic communication patterns.

- **Priority Entropy:** The entropy of the softmax-normalized priority logits v_t^i, averaged across agents. This reflects the network's confidence in its slot assignments—lower entropy corresponds to more confident (i.e., peaked) priority scores.

Figure C.2(b) illustrates the evolution of both entropy metrics across training. Entropies stabilize within the first 4 million timesteps, with message entropy converging to approximately 13 and priority entropy to around 0.9. Variability across seeds remains minimal (standard deviation < 0.2), underscoring the robustness of the co-adaptive training process. The rapid stabilization of these metrics further supports the absence of degenerative feedback loops, even under the high coordination demands of `Terran_10_vs_11`.

C.3 Step 3: Staged Training Ablation

To examine whether the mutual influence between message generation and priority scoring induces instability, we conduct a staged training experiment consisting of two decoupled phases, followed by comparison with standard joint optimization:

1. **Phase 1 (Message Pretraining):** Priority assignments are sampled randomly, and only the message generator θ_G is trained.
2. **Phase 2 (Priority Learning):** The message generator is frozen, and only the priority scoring network θ_S is optimized.
3. **Baseline (Joint Training):** All modules are trained concurrently, as in the standard ADAPT framework.

Figure C.3 presents the win rate, message entropy, and priority entropy trajectories for both schemes in the `Terran_10_vs_11` scenario. Joint training achieves superior performance, with a higher final win rate (53.7% versus 41.8%) and faster convergence (7.8 million versus 8.6 million timesteps). Moreover, the entropy dynamics under joint training exhibit reduced variance, particularly in message entropy (standard deviation approximately 0.04 compared to 0.1 under staged training). This suggests that freezing one module while optimizing the other impairs the co-adaptation process and delays convergence.

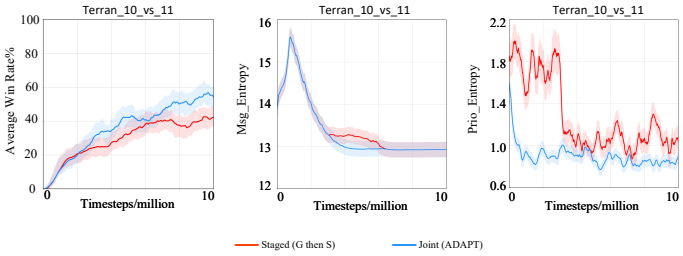


Figure C.3. Staged vs. joint training on `Terran_10_vs_11`. Joint optimization achieves higher win rates, smoother entropy dynamics, and faster convergence compared to staged training.

These findings support the conclusion that joint training not only avoids instability but also enables synergistic co-adaptation. While the modules can be trained independently, the coordinated learning process leads to more efficient optimization and improved communication-structure alignment. In summary, our empirical investigation yields the following conclusions:

- Joint training leads to improved performance and faster convergence relative to staged training.
- All monitored loss and entropy metrics remain stable, with low variance across seeds.
- Co-adaptation is achieved without cyclic gradient interference, benefiting from early random initialization and agent-wise normalization.

Although the message generation and priority scoring modules are mutually dependent during training, this interaction does not constitute a circular dependency in the optimization graph. Instead, the modules evolve cooperatively and reliably, as confirmed by the convergence patterns, entropy dynamics, and the superiority of joint training over staged alternatives.

D Complexity Derivation and Scalability Analysis

D.1 Computational Complexity

We analyze the per-timestep computational complexity of ADAPT by decomposing the core operations into three components. Let N denote the number of agents, D_o the observation dimension, D_a the action dimension, and D_c the message dimension.

Observation Encoding and Dependency Estimation. Each agent encodes its observation and evaluates the influence of messages from all other agents using KL divergence. This results in $O(N^2)$ pairwise comparisons, each involving Transformer-based computations over inputs of dimension $D_o + D_c$. The complexity of this component is:

$$O\left(N^2(D_o + D_c)^2\right). \tag{21}$$

Distributed Priority Scheduling. Priority scheduling involves a decentralized auction in which each agent evaluates utilities across N priority slots in every round. Following [2], the number of auction rounds typically scales logarithmically with agent count, yielding:

$$O\left(N^2 \log N\right). \tag{22}$$

Autoregressive Action Inference. Action selection proceeds sequentially, with each agent conditioning on the reconstructed global features and the actions of higher-priority agents. This induces a chain of N conditional decisions, each involving policy computation, leading to:

$$O\left(N^3(D_o + D_a)\right). \tag{23}$$

Total Complexity. Combining the above, the total computational cost per timestep is:

$$O\left(N^3(D_o + D_a) + N^2(D_o + D_c)^2\right). \tag{24}$$

D.2 Communication Complexity

Communication complexity includes both message exchange and auction-related coordination. Each agent broadcasts a D_c-dimensional message to all other agents at each timestep, forming a fully connected communication graph with:

$$O\left(N^2 D_c\right) \tag{25}$$

bytes of message transmission.

In addition, the distributed auction requires agents to exchange bids and slot price updates. Assuming $N_r = \mathcal{O}(\log N)$ auction rounds and up to $N_b < N$ active slots per round, the additional coordination cost per timestep is:

$$O\left(N^2 \log N\right) \tag{26}$$

for auction-related communication.

Total Communication Cost. The combined communication complexity per timestep is:

$$O\left(N^2 D_c + N^2 \log N\right), \tag{27}$$

with the message broadcasting term typically dominating as D_c grows. ADAPT mitigates this cost through two design strategies: (1) compressed 24-dimensional dense messages, and (2) selective auction triggering based on dependency change (Δw_t^i), which reduces redundant bidding.

Empirical results (Table 1) confirm that ADAPT transmits averagely 3.77% fewer bytes than CommFormer and averagely 51.54% less than SeqComm in SMACv2 and GRF. For example, in the Terran_10_vs_11 scenario, ADAPT transmits 8,837 bytes per timestep, compared to 9,216 (CommFormer) and 18,600 (SeqComm).

The choice of $D_c = 24$ is empirically justified in Table 3 where increasing D_c beyond 48 yields minimal win rate improvement while significantly increasing message size. Therefore, 24-dimensional messages strike a balance between expressiveness and communication efficiency, contributing to ADAPT's scalability under bandwidth constraints.

D.3 Empirical Scalability Results

We evaluate ADAPT's scalability on SMACv2 by varying team sizes from 5 to 20 agents per side. For each configuration, we report four metrics:

- **Win Rate (WinR)**: final performance after 10M training timesteps.
- **Convergence Time (ConT)**: steps required to reach 95% of WinR.
- **Message Size (MesB)**: average bytes transmitted per timestep.
- **Simulation Speed (FPS)**: environment frames per second on an RTX 3090 GPU.

Table D.2 summarizes these results across Terran, Protoss, and Zerg maps. As agent population increases, win rates decrease marginally—for example, in Terran scenarios from 79.14% (5_vs_5) to 76.43% (20_vs_20)—suggesting that ADAPT maintains coordination quality under growing inter-agent complexity.

Despite increasing agent count, convergence time remains within a narrow band of 7.0M–8.0M steps across all races. This indicates that training dynamics are not substantially impacted by scale, consistent with bounded gradient variance under autoregressive updates.

As expected from the theoretical communication complexity $O(N^2 D_c)$, message volume grows with agent count, from under 2KB (5 agents) to approximately 37KB (20 agents). However, selective auction triggering and dependency-based message pruning effectively limit overhead growth.

Simulation speed declines by less than 12% as the agent population quadruples, indicating that runtime cost from message processing and auction coordination scales sublinearly in practice. This observation supports the tractability of ADAPT's computational complexity $O(N^3(D_o + D_a))$ under efficient GPU-parallelized execution.

These findings collectively demonstrate that ADAPT scales gracefully in terms of coordination performance, convergence efficiency, communication overhead, and runtime throughput. Its decentralized design and dynamic prioritization mechanism are well-suited for large-scale partially observable MARL environments.

E Effect of Mutual Information on Message Representation

To assess the representational impact of the mutual information (MI) objective introduced in Section 3.1, we visualize the distribution of message embeddings using t-SNE. Figure E.4 compares the latent spaces learned with and without the MI loss.

Experiments are conducted on the `Terran_10_vs_10` scenario using five random seeds. For each seed, the final model is evaluated over 32 episodes. From the resulting trajectories, which contain $5 \times 32 \times 10 \times T$ data points (with 10 agents and T timesteps per episode), we uniformly sample 500 message embeddings for visualization. Each point is colored based on one of 16 discrete actions, reflecting the intended behavior associated with the corresponding message.

As shown in Figure E.4, applying the MI objective yields well-clustered embeddings, indicating that the message space captures structured behavioral semantics. In contrast, removing the MI objective leads to a more diffuse distribution, with reduced separability among action-relevant signals. This qualitative difference reflects the role of MI in promoting discriminative and policy-relevant message encoding. The visual evidence corroborates the quantitative performance gap reported in Table 2, and helps clarify the contribution of the MI term to coordination efficacy.

Table D.2. Scalability comparison of ADAPT with baselines (CommFormer and SeqComm) on SMACv2 maps across increasing agent team sizes. Metrics include final win rate (WinR), convergence time in millions of timesteps (ConT), average message size per timestep in bytes (MesB), and simulation speed in frames per second (FPS). Results are grouped by unit type (Terran, Protoss, and Zerg).

Map	unit	CommFormer				SeqComm				ADAPT(ours)			
		WinR(%)	ConT(M)	MesB	FPS	WinR(%)	ConT(M)	MesB	FPS	WinR(%)	ConT(M)	MesB	FPS
Terran	5_vs_5	57.58	7.85	2048	123	68.92	8.52	4000	88	**79.14**	7.26	2009	116
	10_vs_10	57.17	8.50	9216	115	65.37	8.34	18600	40	**78.09**	7.90	8698	113
	20_vs_20	54.47	8.63	38912	111	63.41	8.59	83600	22	**76.43**	7.93	37378	110
Protoss	5_vs_5	61.81	7.87	2048	131	72.65	8.61	4000	90	**81.64**	7.05	2024	120
	10_vs_10	62.26	8.63	9216	118	69.88	7.88	18600	45	**78.87**	7.85	8736	118
	20_vs_20	58.66	8.47	38912	115	62.67	8.54	83600	24	**74.21**	7.94	37449	115
Zerg	5_vs_5	56.16	7.80	2048	117	54.75	7.05	4000	84	**68.82**	7.33	1973	108
	10_vs_10	54.54	8.06	9216	108	53.45	7.62	18600	38	**66.30**	7.78	8687	107
	20_vs_20	52.98	7.96	38912	106	52.02	8.66	83600	21	**66.27**	7.80	36916	106

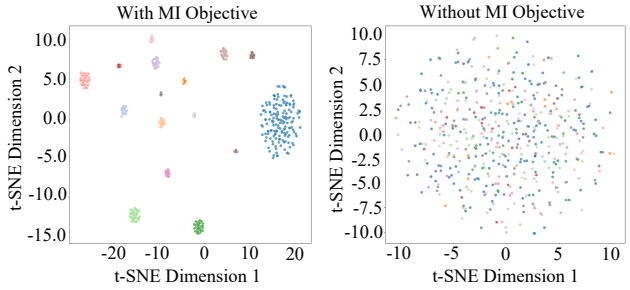


Figure E.4. t-SNE visualization of message embeddings. **Left**: with MI objective. **Right**: without MI. With MI, embeddings form distinct clusters aligned with behavioral patterns. Without MI, the message space becomes diffuse, indicating reduced semantic structure.

Table F.3. Message bytes comparison on challenging and large-scale SMACv1 and GRF scenarios.

Map	Method	MesB (Bytes)
5m_vs_6m	CommFormer	2048
	SeqComm	4000
	ADAPT (Ours)	**2015**
27m_vs_30m	CommFormer	46,980
	SeqComm	100,440
	ADAPT (Ours)	**44,982**
counterattack_hard	CommFormer	1,229
	SeqComm	2,384
	ADAPT (Ours)	**1,170**

F Evaluation on Challenging and Large-Scale Scenarios

We further evaluate ADAPT on three demanding tasks: 5m_vs_6m and 27m_vs_30m from SMACv1, and counterattack_hard from GRF. These maps involve numerical disadvantages and large team sizes, posing significant coordination challenges. As shown in Figure F.5 and Table F.3, ADAPT achieves consistently superior performance while reducing communication load.

- On 5m_vs_6m, which evaluates coordination under numerical disadvantage, ADAPT attains a 95.20% win rate, exceeding SeqComm by +5.1 percentage points.
- On the large-scale 27m_vs_30m map, ADAPT achieves a 98.38% win rate, outperforming SeqComm by +2.94 pp while using less than half the communication bandwidth, underscoring its scalability.
- In the complex counterattack_hard GRF scenario, demanding both offensive and defensive coordination, ADAPT reaches a 64.64% win rate, consistently surpassing all baselines.

These results demonstrate that ADAPT's dependency-aware prioritization scales effectively to scenarios with greater difficulty, larger agent populations, and more complex tactical demands.

G Robustness and Heterogeneous Task Performance

We extend our evaluation to examine ADAPT's robustness under non-ideal communication conditions and its ability to coordinate heterogeneous agent teams.

G.1 Robustness Under Simulated Message Latency

Although ADAPT assumes synchronous, reliable communication, real-world systems often experience latency. To assess robustness, we evaluated Terran_10_vs_11 with random message delays during decentralized execution. At each timestep, the delivery of agent messages was delayed by $d_{ij} \sim U[0, \tau_{max}]$ with $\tau_{max} = 2$, forcing agents to compute dependency scores using partially outdated peer information. To support this, each agent buffered its most recent messages and retrieved delayed entries when required. This setup mimics moderate asynchrony in networked environments.

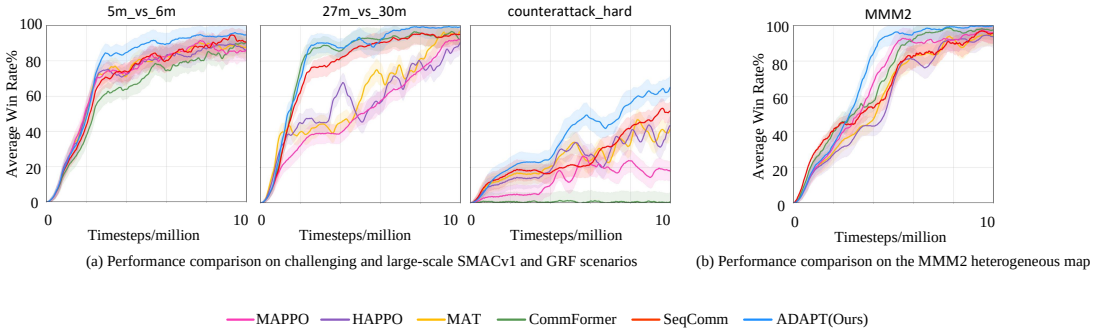
As shown in Figure G.6, introducing random delays led to only a 2.85-point reduction in win rate. This modest drop indicates that aggregated dependency estimation and dynamic priority scheduling preserve coordination effectiveness despite temporal misalignment, supporting ADAPT's suitability for deployment in less reliable communication settings.

G.2 Performance on the MMM2 Heterogeneous Scenario

We further validated ADAPT in heterogeneous coordination using the MMM2 scenario from SMACv1, which features Marines, Marauders, and Medivacs with complementary roles. Results are summarized in Figure F.5 (b) and Table G.4.

Table G.4. Message bytes comparison on MMM2.

Method	MesB (Bytes)
CommFormer	9,216
SeqComm	18,600
ADAPT (Ours)	**8,696**



(a) Performance comparison on challenging and large-scale SMACv1 and GRF scenarios

(b) Performance comparison on the MMM2 heterogeneous map

Figure F.5. Performance comparison on challenging, heterogeneous, and large-scale SMACv1 and GRF scenarios.

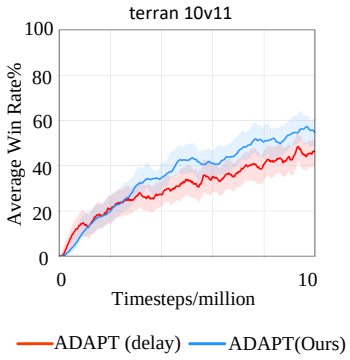


Figure G.6. Win rate of ADAPT on `Terran_10_vs_11` with and without simulated latency.

On `MMM2`, ADAPT attained a win rate of 99.65%, surpassing strong baselines including SeqComm (+0.77 pp), while converging faster and reducing communication cost. The dynamic priority mechanism successfully identifies tactical dependencies across functionally distinct agents, enabling Medivacs to support frontline units and Marauders to shield Marines. These results highlight ADAPT's ability to generalize beyond homogeneous teams and efficiently orchestrate mixed-role coordination.