A SEMI-DECENTRALIZED APPROACH TO SCALABLE MULTIAGENT CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Scalable multiagent control under stochastic communication remains an open challenge. Existing planning methods such as RS-SDA* scale poorly, sampling-based approaches such as Dec-MCTS assume fixed-period communication, and MARL methods typically impose either strict execution-time decentralization (QMIX, MAPPO) or full centralization (MAZero), respectively limiting coordination opportunities or ignoring the stochastic communication constraints that arise at decision time. We consider the Semi-Decentralized POMDP (SDecPOMDP), which unifies the Dec-POMDP and MPOMDP through a distribution over communication-sojourn times. We introduce SDecMCTS, a multiagent MCTS algorithm that achieves solution quality comparable to RS-SDA* while requiring less computation across benchmarks. Building on the SDecPOMDP's semi-Markov communication dynamics, SDecMCTS uses an options framework in which decentralized solvers execute between communication events, enabling planning and execution that faithfully reflects semi-decentralization. We extend SDecMCTS to Semi-Decentralized Zero (SDecZero), a scalable AlphaZero-style algorithm that learns semi-decentralized value estimates used by decentralized solvers. SDecZero additionally learns the stochastic communication dynamics through a dedicated network communication head. SDecZero substantially outperforms model-based and model-free methods across large multi-drone search scenarios constructed from real cave systems associated with the DARPA Subterranean Challenge.

1 INTRODUCTION

Several real-world multiagent problems, including robot exploration teams (Saboia et al., 2022; Rouček et al., 2019), cooperative driving (Zhao et al., 2021; Li et al., 2023), and satellite collision avoidance (Dolan et al., 2023; Zhao et al., 2025), feature stochastic communication dynamics conditioned on the underlying environment or actions taken. These *semi-decentralized* systems (Al-Husseini et al., 2026a) necessitate planning over the stochastic information structure under which actions are executed and observations are received. Consider a robot team conducting a search and rescue mission in a subterranean cave with line-of-sight communication, depicted in Figure 1. The robots may temporarily share observations and coordinate a joint plan, lose contact while moving through different tunnels, and later reconnect with new local histories. Most MARL methods, including centralized training with decentralized execution (CTDE) (Amato, 2024) and learned-communication approaches (Zhu et al., 2024), assume a fixed execution-time information structure. Agents either execute decentralized policies, possibly conditioned on learned messages, or operate under an always-centralized planner. Few methods perform online joint-action planning at intermittent communication opportunities; exceptions include the Dec-POMDP planners (Al-Husseini et al., 2026b) that inspire our work. Recent multiagent model-based tree-search algorithms (Liu et al., 2023; Hao et al., 2024) demonstrate improved sample efficiency, but again assume consistency in communication. There is a well-defined gap between scalable and efficient MARL algorithms and domains in which communication opportunities are stochastic, recurrent, and action- or state-dependent. We address this by explicitly modeling the execution-time information structure as part of the control problem, enabling agents to plan over both actions and the future communication regimes those actions may induce.

We adopt the *Semi-Decentralized POMDP* (Al-Husseini et al., 2026a), a multiagent model in which opportunities to communicate and coordinate arise stochastically from the environment dynamics.

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.

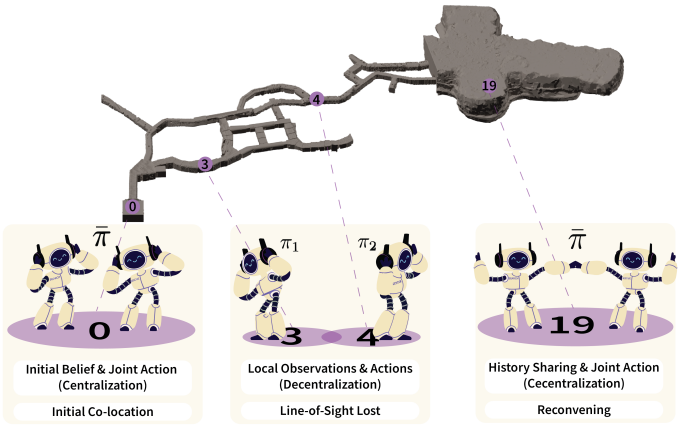


Figure 1: Illustrating semi-decentralization in the DARPA Subterranean Challenge environment.

We first introduce *SDecMCTS*, a sampling-based tree search over the shared belief in which every loss of communication opens an option. Each decentralized window is solved as a small Dec-POMDP whose exit beliefs are valued by continued centralized planning. This allows joint action selection to account for the communication regime it may induce. We then scale this same semi-decentralized search with *SDecZero*, an AlphaZero-style method that replaces the planner's hand-built heuristics and exit prices with policy and value networks trained from self-play. Finally, we learn the stochastic communication-sojourn model during online search through a dedicated network head. Our results show that search over communication regimes, guided by hand-crafted heuristics in SDecMCTS and learned networks in SDecZero, recovers much of the value of centralized coordination under stochastic communication loss while retaining the per-episode computational cost of online planning.

Contributions

- We introduce SDecMCTS, a semi-decentralized multiagent extension of MCTS with excellent performance across 17 standard semi-decentralized planning benchmarks.
- We introduce SDecZero, the AlphaZero-like extension of SDecMCTS, to enhance decentralized policy generation and enable significant scaling. We further learn the communication process by adding a network communication head.
- We show that SDecZero substantially outperforms model-free (MAPPO, IPPO, QMIX) and model-based (MAZero) learning methods in three real-world semi-decentralized cave system simulations, including those affiliated with the DARPA Subterranean Challenge.

2 Related Work

Planning approaches. Multiagent planning is commonly formalized using the Dec-POMDP (Oliehoek et al., 2016), which provides a principled model for cooperative decentralized agents acting under partial observability. Dec-POMDP solvers such as MAA* (Szer et al., 2005) and modern recursive small-step variants (Koops et al., 2023; 2024) optimize policy trees with admissible heuristic search and partial policy pruning, but remain limited by the combinatorial growth of joint policies and observation histories. More directly related to our setting, the SDec-POMDP formalism and RS-SDA* extend this line of work to semi-decentralized systems in which communication evolves stochastically per the environment or control strategy (Al-Husseini et al., 2026a;b). Hybrid decentralized search techniques like Dec-MCTS (Best et al., 2019) and Dec-MCTS-SP (Li et al., 2019) improve scalability by maintaining agent-specific search-trees aided by prediction-based heuristics; they are however limited to the narrow case of fixed-period communication, and do not implement networks for scale.

Model-free multiagent reinforcement learning. Model-free MARL has produced scalable algorithms for cooperative and mixed multiagent domains, primarily through CTDE. Actor-critic methods such as MADDPG (Lowe et al., 2017) and COMA (Foerster et al., 2018) use centralized critics to

stabilize learning and address multiagent credit assignment while deploying local policies at execution time. Value-factorization methods such as VDAC (Su et al., 2021), QMIX (Rashid et al., 2020), and QPLEX (Wang et al., 2021) learn structured decompositions of the joint action-value function that permit decentralized action selection. Policy-gradient methods such as MAPPO (Yu et al., 2022) have proven performant across standard cooperative MARL benchmarks. Although scalable and generally effective, model-free approaches perform poorly in multiagent domains with stochastic information flow, sparse reward signals, and critical joint actions. Model-free approaches are sample inefficient, which degrades performance when interaction with the environment is limited. We discuss MARL-COM in the appendix, which optimizes message content or routing under a prescribed execution-time communication structure; this is distinct from optimizing over problem-specific semi-decentralized communication dynamics.

Model-based multiagent reinforcement learning. Model-based MARL improves sample efficiency by exploiting known or learned dynamics models to enable imagined roll-outs. MAMBA learns a multiagent communication world model to reduce environment interaction in cooperative tasks (Egorov & Shpilman, 2022). MuZero-style methods that incorporate tree search into multi-agent learning have become increasingly popular: MAZero uses a centralized model with MCTS to enhance policy search and introduces two search efficiency techniques (Liu et al., 2023), MA Gumbel MuZero replaces the traditional PUCT exploration mechanism with Gumbel-top-k search to overcome combinatorial joint-action spaces (Hao et al., 2024), and MALinZero cleverly projects joint-action returns into a low-dimensional space and solves a contextual linear bandit problem (Tang et al., 2026). Notably, these algorithms assume a fixed execution-time information structure. In contrast, our methods explicitly model stochastic transitions between communication regimes and derive centralized and decentralized behavior from a single semi-decentralized search procedure.

3 PRELIMINARIES

Definition 1. The *semi-Markov property for communication*, or *semi-decentralization*, admits a distribution over time for what information agents can store in memory.

Sojourn communication time τ is a general continuous random variable representing the time for an agent to return to an information-sharing state. As with SMDPs, we can define

$$Q(\bar{\mathcal{T}} \leq \bar{\tau}', s' \mid \bar{\tau}, s, \bar{a}, \bar{a}'), \tag{1}$$

as the product of $F(\bar{\tau}' \mid s', \bar{a}', \bar{\tau})$ and $T(s' \mid s, \bar{a}, \bar{\tau})$, where $\bar{\tau}'$ may be conditioned on the subsequent joint action set $\bar{a}'$. In a standard SMDP, the next sojourn is measured from a single decision epoch, so the temporal origin can be taken as zero. In an SDec-POMDP, agents may occupy different temporal phases when a joint state is reached. The vector $\bar{\tau}$ therefore records these agent-specific temporal offsets, and the distribution of the subsequent communication sojourn $\bar{\tau}'$ is conditioned on $\bar{\tau}$. When $\tau = 0$, information sharing can occur coinciding with a *communication epoch*. We assume noise-free instantaneous broadcast communication resulting in a single communicating agent set as in a blackboard (Erman et al., 1980; Craig, 1988). Semi-decentralization may however incorporate multiple distinct communicating sets. Whereas semi-Markov control systems toggle model transition dynamics using τ, semi-decentralized systems toggle updating histories using $\bar{\tau}$.

We pose the problem as a semi-decentralized partially observable Markov decision process (SDec-POMDP) Al-Husseini et al. (2026a). The SDec-POMDP is a principled model for cooperative multiagent teams subject to stochastic communication constraints. The SDec-POMDP unifies the Dec-POMDP and MPOMDP frameworks along a communication dimension.

3.1 SDEC-POMDPS

Shown in Figure 2, the *semi-decentralized partially observable Markov decision process* (SDec-POMDP) is a multiagent decision process for sequential decision-making under partial observability and probabilistic communication characterized by tuple $\langle I, S, \bar{A}, \bar{\mathcal{O}}, T, O, R, F \rangle$, where:

- I is a finite set of k agents,

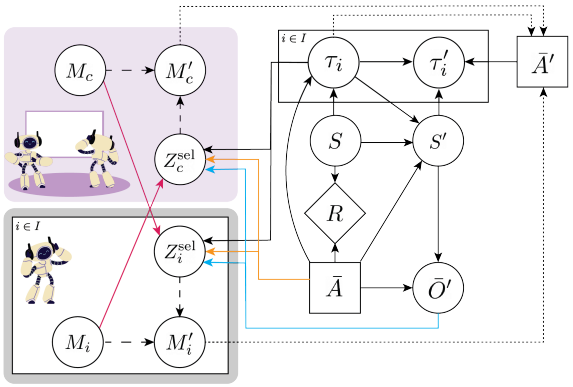


Figure 2: The SDec-POMDP Bayes net. Policy infrastructure is shown on the left and model on the right. The purple backdrop is the blackboard with memory M_c generated from the histories of communicating agents. The gray backdrop with plate notation includes the individual agent memories M_i. Z selector nodes are selectively toggled by $\bar{\tau}$ to facilitate memory propagation η, represented by dashed lines. Policy ψ edges are represented by dotted lines.

- S is a finite set of states,
- $\bar{A} = \times_i A_i$ is a finite set of joint actions,
- $\bar{\mathcal{O}} = \times_i \mathcal{O}_i$ is a finite set of joint observations, and
- $T : S \times \bar{A} \times \bar{\mathbb{R}}^+ \times S \to [0, 1]$ is a state transition function where $T(s' \mid s, \bar{a}, \bar{\tau})$ is the probability of state s' given joint action $\bar{a}$ being performed in state s with joint sojourn $\bar{\tau}$,
- $O : \bar{\mathcal{O}} \times S \times \bar{A} \to [0, 1]$ is a joint-observation function where $O(\bar{o}' \mid s', \bar{a})$ specifies the probability of attaining joint-observation $\bar{o}'$ when joint action $\bar{a}$ results in state s',
- $R : S \times \bar{A} \to \mathbb{R}$ is a reward function such that $R(s, \bar{a})$ is the immediate reward for performing joint action $\bar{a}$ in state s, and
- $F(\tau' \mid s', \bar{a}', \bar{\tau})$ is the *communication sojourn distribution*, the probability an agent is decentralized for duration τ' conditioned on state s', joint action $\bar{a}'$, and last joint sojourn $\bar{\tau}$.

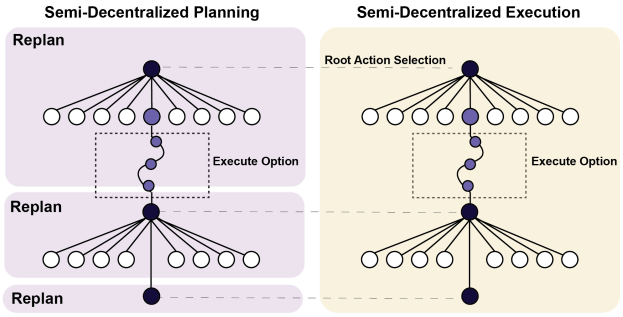


Figure 3: SDecMCTS. Search (left) alternates between synchronized nodes and decentralized windows, each solved as a depth-capped Dec-POMDP and backed up as an option. Execution (right) follows the tree: joint actions in contact, window policy out of contact, and replan on resync.

4 SDecMCTS

Overview. To our knowledge, SDecMCTS is the first sampling-based online planner for the semi-decentralized problem class. SDecMCTS begins from an initial belief and constructs a regime-faithful search tree that alternates between centralization and decentralization (Figure 3). At centralized nodes, joint actions are selected by a selection rule (UCT, PUCT, Gumbel) over the shared belief.

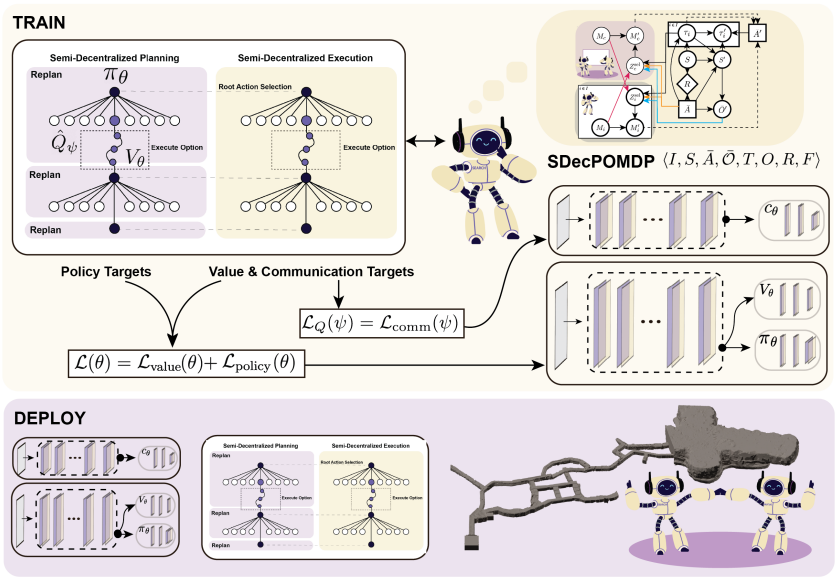


Figure 4: SDecZero algorithm outline.

Chance outcomes, including communication outcomes and joint observations, are sampled under progressive widening, enabling search over both deterministic and stochastic observation models. At decentralized nodes, a bounded-horizon Dec-POMDP window is built and solved for per-agent policies by an offline planner from the multiagent A* family (Oliehoek et al., 2008; Koops et al., 2023). The tree treats the resulting decentralized policy as an option (Sutton et al., 1999): a temporally extended policy whose cumulative reward and terminal belief are backed up in search.

The window model is built over state-information-state pairs, merging joint action-observation histories of equal depth that induce the same joint belief. Each resynchronization is therefore belief-determinate, and its continuation is valued by the centralized value of the corresponding exit belief. This value is computed exactly by branch and bound in deterministic-observation domains and approximated by the fully observable relaxation in stochastic-observation domains. Probability mass that reaches the depth cap while agents remain out of contact receives zero continuation value, providing a conservative estimate that encourages window policies to regain contact. The decentralized solver therefore proposes out-of-contact policies under a centralized upper bound on semi-decentralized continuation, while the search tree evaluates those proposals by simulating their execution under the true dynamics and continuing search from the beliefs at which communication resumes. We refer to this as *Semi-Decentralized Planning and Execution* (SDPE).

5 SDecZero

Overview. SDecZero (Figure 4) extends SDecMCTS with learned policy and value approximators in the tradition of AlphaZero (Silver et al., 2018) and BetaZero (Moss et al., 2024). The search structure is unchanged. Synchronized nodes are expanded by PUCT and decentralized epochs are solved as depth-capped windows. A joint-action policy head $\pi_\theta(\phi(b), h)$ supplies PUCT priors, while a value head $\hat{V}_\theta(\phi(b), h)$ supplies leaf, blackout, and window exit values. Where SDecMCTS prices exits using an optimistic centralized continuation bound, SDecZero instead learns their value from returns realized under semi-decentralized execution.

Training proceeds by minimizing

$$\mathcal{L}_{\text{net}} = -\frac{1}{N_\pi} \sum_{i \in \mathcal{D}_{\text{sync}}} \pi_i^\top \log p_\theta(\cdot \mid \phi(b_i), h_i) + \frac{1}{N} \sum_{i \in \mathcal{D}} \left(V_\theta^{k_i}(\phi(b_i), h_i) - z_i \right)^2, \tag{2}$$

where π_i is the normalized PUCT root-visit distribution, z_i is the realized normalized return, and $k_i \in \{\text{sync}, \text{boundary}\}$ selects the corresponding value output. Exit beliefs are additionally sampled for

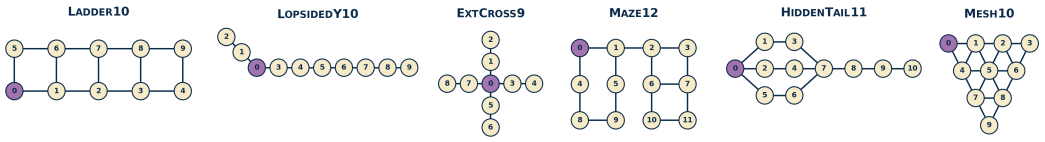


Figure 5: Small labyrinth layouts

continuation rollouts, directly supervising the boundary values used to price subsequent decentralized windows. Actions during these windows remain those of the decentralized solver.

Learned Communication. When the communication process is unknown, SDecZero separately learns F, the communication component of Q, from synchronization events observed during execution. The communication model shares neither parameters nor loss terms with the policy/value network and affects them only indirectly by shaping search. It is queried whenever search branches on communication availability, including window entry and exit.

6 EXPERIMENTS

We evaluate in two stages. First, SDecMCTS tests whether semi-decentralized search can recover high-quality solutions on problems where strong planning baselines are available. Second, SDecZero tests whether learned policy, value, and communication models preserve those advantages at substantially larger scale. We compare SDecMCTS against RS-MAA*, RS-SDA*, and Dec-MCTS on standard semi-decentralized benchmarks, and SDecZero against QMIX, MAPPO, IPPO, MASAC, and SDec-MAZero on large cave-search problems.

Setup. All experiments were conducted on an AMD Ryzen 9 9900X3D 12-Core Processor (4400 MHz). SDecMCTS and SDecZero are implemented in Julia and code is made available in the supplementary materials to support reproducibility. We compare against and co-opt the RS-MAA* implementation provided by Koops et al. and the RS-SDA* implementation provided by Al-Husseini et al.. MAPPO, IPPO, MASAC, and QMIX were sourced from the BenchMARL library Bettini et al. (2024) and MAZero was sourced from the MAZero library Liu et al. (2023). MA-Gumbel-MuZero Hao et al. (2024) and MALinZero Tang et al. (2026) libraries have not been made publicly available and were therefore not considered.

Planning Benchmarks. We evaluate SDecMCTS on three standard SDec-POMDP benchmarks from Al-Husseini et al. (2026a), SDEC-TIGER, SDEC-MARS, and MARITIMEMEDEVAC, and on two families of two-agent labyrinth problems from Al-Husseini et al. (2026b). In SDEC-TIGER, communication is triggered by the agents' actions, whereas in SDEC-MARS and MARITIMEMEDEVAC it is determined by their positions. The labyrinth benchmarks include SEARCH & RETURN and SEARCH & RESCUE. In the former, agents search a graph for a hidden target and must return the discovery to the start node under deterministic observations. In the latter, target detection is noisy and the agents must commit to a rescue at the target; the second agent can assist from nearby, while rescues attempted at an incorrect node incur a large penalty. In both families, agents share information and select joint actions only while in line of sight, making communication a function of the environment geometry. We evaluate on the six graphs shown in Figure 5 and on two 15-node three-dimensional cave graphs, TUNNELS-15 and CHAMBERS-15.

Learning Benchmarks. We use *Prospector* (Ward et al., 2025) to simulate real world data from the Louisville Mega Cavern, part of the DARPA Subterranean Challenge. We also consider the northern California Grapevine cave. Prospector re-meshes 3D data gathered from caving expeditions to enable simulation. We transform these maps into realistic 25-node traversability and coordination graphs. SDecZero performance is then compared against a wide range of model-free and model-based multiagent learning algorithms. We also evaluate our ability to learn the communication dynamic directly across benchmarks, in Fig. 7.

Table 1: SDecMCTS performance across standard semi-decentralized planning benchmarks. TO denotes timeout (>1200s). SDecMCTS results are finite-sample empirical returns and may exceed the exact optimal expected-value of the fully centralized problem.

	decentralized		*periodic*		*semi-decentralized*				*centralized*	
	RS-MAA* (lower bound)		Dec-MCTS		App. RS-SDA*		SDecMCTS **(Our Approach)**		Exact RS-SDA* (upper bound)	
h	value	time	value	time	value	time	value	time	value	time
SDEC-TIGER										
10	13.57	2	29.09 ± 10.76	5	60.26 ± 0.86	1	62.28 ± 2.34	<1	60.51	1
12	20.76	2	35.61 ± 10.98	6	73.61 ± 2.85	1	76.46 ± 2.64	<1	73.39	1
15	25.95	2	47.63 ± 12.51	9	94.76 ± 3.09	2	95.94 ± 2.72	<1	92.67	1
20	28.01	3	65.41 ± 14.06	14	120.94 ± 4.46	4	130.66 ± 3.14	<1	125.19	1
SDEC-MARS (Right Band Rendezvous)										
7	20.90	2	10.62 ± 1.73	5	21.17	1	19.96 ± 0.21	<1	21.17	<1
8	22.48	2	8.20 ± 1.86	6	23.65	1	23.42 ± 0.22	<1	23.65	<1
9	24.31	3	9.27 ± 2.02	7	25.21	2	24.74 ± 0.31	<1	25.70	<1
10	26.31	4	7.88 ± 2.25	7	28.28	2	28.03 ± 0.30	<1	28.61	<1
MARITIMEMEDEVAC										
7	3.25	<1	4.44 ± 1.58	15	6.36	<1	6.29 ± 0.51	<1	6.62	<1
8	8.03	<1	5.63 ± 1.65	20	10.61	<1	10.89 ± 0.56	<1	10.88	<1
9	10.79	<1	8.33 ± 1.50	27	12.74	<1	12.98 ± 0.52	<1	13.01	<1
10	12.15	<1	7.98 ± 1.59	34	13.35	<1	13.37 ± 0.44	<1	13.61	<1
LABYRINTH EXTCROSS9 SEARCH & RETURN (N = 2)										
6	71.25	15	75.81 ± 16.17	5	84.38	1	84.38	<1	84.38	1
7	–	TO	79.95 ± 13.26	6	84.25	2	83.97 ± 0.06	1	84.25	5
8	–	TO	83.31 ± 11.31	7	96.75	1	96.75	1	96.75	1
LABYRINTH LOPSIDEDY10 SEARCH & RETURN (N = 2)										
5	40.78	<1	56.57 ± 28.76	4	74.67	<1	74.67	<1	74.67	<1
6	51.44	<1	71.86 ± 26.18	4	85.67	<1	85.62 ± 0.04	<1	85.67	<1
7	51.00	<1	75.84 ± 22.36	5	96.78	<1	96.78	<1	96.78	<1
LABYRINTH LADDER10 SEARCH & RETURN (N = 2)										
5	40.78	<1	56.63 ± 27.85	1	74.56	<1	74.49 ± 0.03	<1	74.67	<1
6	62.67	<1	68.33 ± 18.52	5	95.89	<1	96.33	1	96.33	<1
7	62.33	<1	70.33 ± 18.24	5	96.44	<1	96.56	2	96.78	<1
LABYRINTH MAZE12 SEARCH & RETURN (N = 2)										
5	32.45	2	39.93 ± 29.60	4	60.09	<1	60.05 ± 0.02	<1	60.18	1
6	59.35	2	40.08 ± 23.46	5	77.64	<1	77.73	2	86.90	1
7	–	TO	46.95 ± 24.74	6	94.82	<1	95.82	3	96.00	1
LABYRINTH HIDDENTAIL11 SEARCH & RETURN (N = 2)										
4	36.80	<1	46.95 ± 27.92	3	57.20	<1	57.08 ± 0.05	<1	57.20	<1
5	36.20	13	63.64 ± 24.83	4	66.90	<1	66.78 ± 0.04	1	66.90	1
6	65.90	2	74.23 ± 18.89	5	95.80	2	96.20	2	96.20	1
LABYRINTH MESH10 SEARCH & RETURN (N = 2)										
5	52.00	<1	58.87 ± 20.70	1	85.67	1	85.60 ± 0.07	<1	85.67	3
6	73.89	6	61.69 ± 19.64	5	96.67	2	96.33	3	96.78	7
7	–	TO	80.64 ± 10.89	6	96.67	3	96.44	5	96.78	8
LABYRINTH EXTCROSS9 SEARCH & RESCUE (N = 2)										
10	17.62	3	45.66 ± 15.14	5	80.11 ± 4.80	14	69.01 ± 4.50	11	–	TO
LABYRINTH LOPSIDEDY10 SEARCH & RESCUE (N = 2)										
10	39.91	62	52.13 ± 17.83	24	65.00 ± 5.55	4	63.90 ± 2.92	4	–	TO
LABYRINTH LADDER10 SEARCH & RESCUE (N = 2)										
10	36.03	59	52.17 ± 18.86	16	83.00 ± 4.29	19	82.98 ± 2.24	14	–	TO
LABYRINTH MAZE12 SEARCH & RESCUE (N = 2)										
10	21.53	62	32.11 ± 23.41	23	43.75 ± 5.24	35	51.45 ± 4.65	38	–	TO
LABYRINTH HIDDENTAIL11 SEARCH & RESCUE (N = 2)										
10	21.02	65	33.86 ± 18.86	31	73.26 ± 5.10	41	70.96 ± 4.66	10	–	TO
LABYRINTH MESH10 SEARCH & RESCUE (N = 2)										
10	48.20	17	48.92 ± 14.70	30	88.10 ± 2.54	115	79.12 ± 3.84	16	–	TO
DARPA 3D TUNNELS-15 SEARCH & RETURN (N = 2)										
11	–	TO	76.05 ± 13.86	9	86.71	25	86.71	8	–	TO
12	–	TO	74.41 ± 16.58	10	86.50	29	83.93 ± 2.39	16	–	TO
13	–	TO	83.42 ± 8.99	12	93.50	34	93.64	10	–	TO
DARPA 3D CHAMBER-15 SEARCH & RETURN (N = 2)										
9	–	TO	65.45 ± 14.75	10	80.64	275	80.41 ± 0.02	48	–	TO
10	–	TO	77.72 ± 12.84	12	87.57	250	87.48 ± 0.05	73	–	TO
11	–	TO	82.90 ± 6.15	14	93.71	252	94.50	88	–	TO

Results. SDecMCTS substantially outperforms periodic decentralized search while matching the solution quality of approximate semi-decentralized planning at substantially lower computational cost. Across the standard benchmarks, its consistent advantage over Dec-MCTS shows the value of planning over stochastic communication opportunities rather than approximating them with fixed-period coordination. At the same time, SDecMCTS closely tracks approximate RS-SDA* across both the tabular problems and SEARCH & RETURN. The distinction becomes computation time as scale grows. On the 15-node DARPA caves, SDecMCTS obtains essentially identical returns in 8 s–88 s per episode, compared with 25 s–275 s for approximate RS-SDA*. SEARCH & RESCUE is less uniform under stochastic observations, but SDecMCTS remains broadly competitive with RS-SDA*.

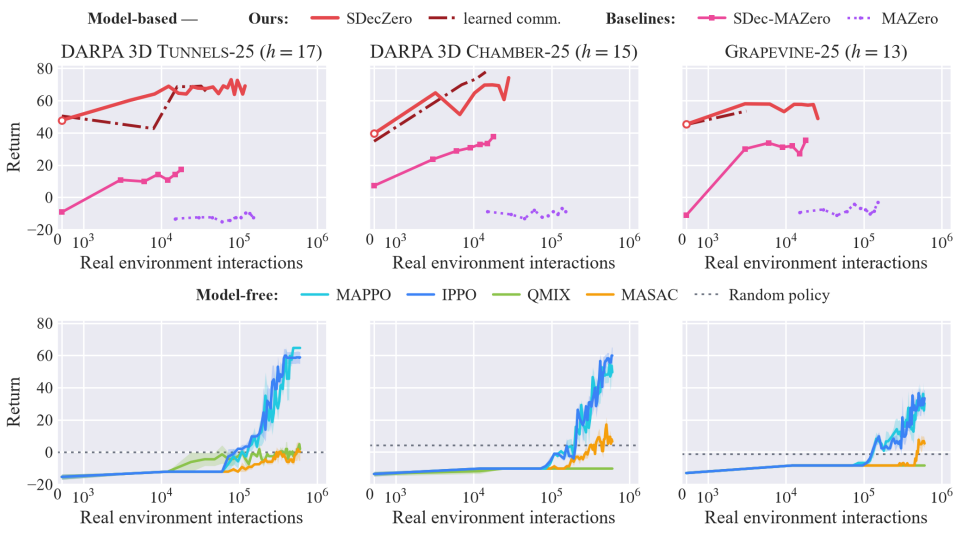


Figure 6: DARPA and Grapevine 25-node learning curves for model-based and model-free methods. Environment-step counts should be interpreted as sample efficiency, not computational efficiency.

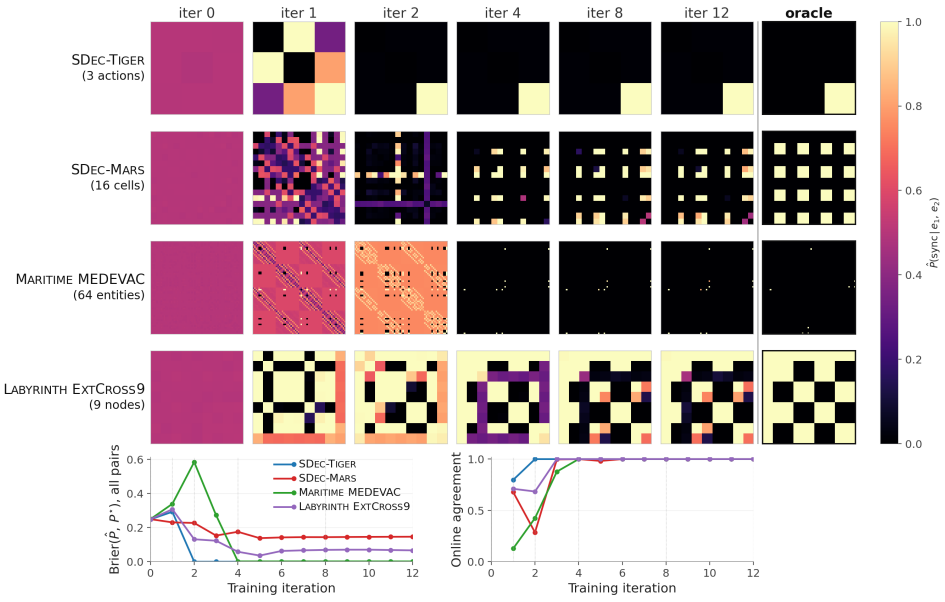
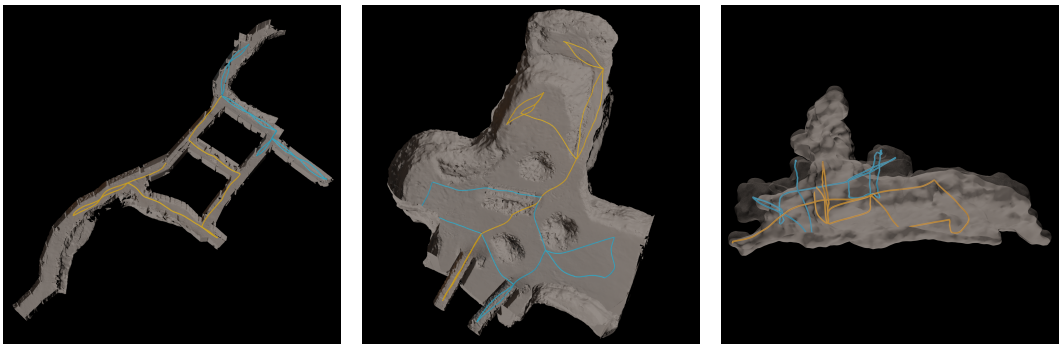


Figure 7: Learning problem communication dynamics (SDecPOMDP function F) across benchmarks.

SDecZero tests whether learned guidance can scale the same semi-decentralized search structure. Across the 25-node caves, it reaches strong returns within thousands of environment interactions, whereas the model-free baselines require orders of magnitude more data or remain near random; this reflects sample efficiency rather than equal computation, as SDecZero trades model-based online search for reduced interaction. Interestingly, either learned search estimate is nearly sufficient on its own. The policy-only and value-only variants closely track the joint model, indicating substantial redundancy between learned action priors and semi-decentralized continuation values, with search compensating for the missing signal. Learning also removes the need to specify the communication process. The communication head rapidly recovers F on the standard benchmarks, and planning with the learned kernel remains within 0.25 return of true-F planning. On the 25-node caves, the learned-F variant likewise closely tracks the known-F model.



(a) *Tunnels* (b) *Chamber* (c) *Grapevine*

Figure 8: Paths flown by the two quadrotors executing the SDecZero tours (agent one blue, agent two orange), shown for one trial per cave.

Table 2: Low-level execution of SDecZero tours in Prospector (Ward et al., 2025). Each tour is flown by two quadrotors, 10 trials per environment (mean ± standard deviation). *Time*: launch to arrival at the final node. *Waiting time*: time spent waiting for the other agent to get into communications range. *Path ratio*: distance actually flown over the total straight line length of the planned node path. *Clearance*: closest approach to an occupied cell (the collision radius is 0.1 m).

	Successes	Time (s)	Waiting time (s)	Path flown (m)	Path ratio	Average speed (m/s)	Clearance (m)
DARPA 3D Tunnels-25 Search & Return (N = 2) 17 tour steps, 1 communication rendezvous							
Agent 1	10/10	50.61 ± 4.34	0.25 ± 0.14	73.24 ± 0.50	1.004 ± 0.007	1.459 ± 0.107	0.174 ± 0.005
Agent 2	10/10	43.48 ± 0.23	0.00 ± 0.00	70.25 ± 0.28	1.013 ± 0.004	1.396 ± 0.105	0.194 ± 0.019
Team	10/10	50.61 ± 4.34	0.25 ± 0.14	143.49 ± 0.78	1.009 ± 0.005	1.427 ± 0.106	0.172 ± 0.004
DARPA 3D Chamber-25 Search & Return (N = 2) 15 tour steps, 3 communication rendezvous							
Agent 1	10/10	83.16 ± 0.14	23.56 ± 0.08	107.27 ± 0.11	1.025 ± 0.001	1.766 ± 0.004	0.182 ± 0.007
Agent 2	10/10	82.43 ± 0.13	3.26 ± 0.10	144.31 ± 0.16	1.023 ± 0.001	1.800 ± 0.003	0.258 ± 0.008
Team	10/10	83.16 ± 0.14	26.82 ± 0.15	251.57 ± 0.24	1.024 ± 0.001	1.783 ± 0.003	0.182 ± 0.007
Grapevine-25 Search & Return (N = 2) 13 tour steps, 3 communication rendezvous							
Agent 1	10/10	71.19 ± 2.72	5.12 ± 0.04	84.20 ± 0.25	1.070 ± 0.003	1.090 ± 0.039	0.169 ± 0.014
Agent 2	10/10	81.94 ± 2.98	18.70 ± 2.61	92.67 ± 0.22	1.059 ± 0.003	1.438 ± 0.017	0.204 ± 0.027
Team	10/10	81.94 ± 2.98	23.82 ± 2.64	176.87 ± 0.44	1.064 ± 0.003	1.264 ± 0.025	0.168 ± 0.013

Simulation. We perform simulated quadrotor experiments using real-world 3D cave geometries shown in Fig. 8 and high fidelity quadrotor dynamics Ward et al. (2025). These experiments quantify how dynamically feasible the computed node-level policies are when actually executed by a low-level controller. We use a sampling-based optimal control method (Model Predictive Path Integral Control Williams et al. (2016)) to compute quadrotor velocity commands at 10Hz, tracked by a 100Hz attitude controller, with the agents guided by a cost function that balances liveness (agents are guided to the desired node locations) and safety (avoid crashing into walls). Across 10 trials per cave, every tour was completed without collision (30/30), with mission times of 50.6 ± 4.3 s (Tunnels), 83.2 ± 0.1 s (Chamber), and 81.9 ± 3.0 s (Grapevine), and with flown paths within 1–7% of the straight line node-path length (Table 2), indicating plans produced by SDecZero are dynamically feasible and realizable on robotic agents in challenging, real-world environments.

7 Conclusion

We introduce SDecMCTS and SDecZero for scalable semi-decentralized multiagent control. SDecMCTS matches the solution quality of offline semi-decentralized planning while scaling to larger problems, and SDecZero further scales the same search structure through learned policy, value, and communication models. SDecZero learns strong policies on 25-node cave environments within thousands of interactions. Future work will learn the presently known environment dynamics and extend to larger agent teams.

AI USE STATEMENT

We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, interpret results, or support qualitative and thematic data analysis, and generate synthetic data sets, assist in the writing of proofs, assist with translation, clean and reformat dataset are not applicable to this work. We however used generative AI tools to identify relevant literature, create and edit software code, and create and modify scientific figures or images. We have reviewed all AI-assisted work. All LLM-generated code was verified and tested for correctness and a manual literature review was used to support the AI-assisted version. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide access to our complete code repository in the supplementary files, and discuss in the appendix, to ensure complete reproducibility of all experiments. This includes SDecMCTS, SDecZero, and SDecMAZero, and considered baselines and benchmarks.

REFERENCES

Mahdi Al-Husseini, Kyle H Wray, and Mykel J Kochenderfer. A semi-decentralized approach to multiagent control. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2026a.

Mahdi Al-Husseini, Kyle H Wray, Isaac R Ward, and Mykel J Kochenderfer. Approximate heuristic search for semi-decentralized systems. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2026b.

Christopher Amato. An introduction to centralized training for decentralized execution in cooperative multi-agent reinforcement learning. *ArXiv*, abs/2409.03052, 2024.

Graeme Best, Oliver M Cliff, Timothy Patten, Ramgopal R Mettu, and Robert Fitch. Dec-mcts: Decentralized planning for multi-robot active perception. *Int. J. Rob. Res.*, 38(2–3):316–337, March 2019. doi: 10.1177/0278364918755924.

Matteo Bettini, Amanda Prorok, and Vincent Moens. Benchmarl: Benchmarking multi-agent reinforcement learning. *Journal of Machine Learning Research*, 25(217):1–10, 2024. URL `http://jmlr.org/papers/v25/23-1612.html`.

Iain D Craig. Blackboard systems. *Artificial Intelligence Review*, 2(2):103–118, 1988.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. TarMAC: Targeted multi-agent communication. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *International Conference on Machine Learning (ICML)*, volume 97, pp. 1538–1546, 2019.

Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

Sydney Dolan, Siddharth Nayak, and Hamsa Balakrishnan. Satellite navigation and coordination with limited information sharing. In Nikolai Matni, Manfred Morari, and George J. Pappas (eds.), *Learning for Dynamics and Control Conference*, pp. 1058–1071, 2023.

Vladimir Egorov and Alexei Shpilman. Scalable multi-agent model-based reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 381–390, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450392136.

Lee D Erman, Frederick Hayes-Roth, Victor R Lesser, and D Raj Reddy. The hearsay-ii speech-understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys (CSUR)*, 12(2):213–253, 1980.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11794. URL https://ojs.aaai.org/index.php/AAAI/article/view/11794.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2145–2153, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

Xiaotian Hao, Jianye Hao, Chenjun Xiao, Kai Li, Dong Li, and Yan Zheng. Multiagent gumbel muzero: efficient planning in combinatorial action spaces. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i11.29121.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems (NeurIPS)*, NIPS'18, pp. 7265–7275, 2018.

Wietze Koops, Nils Jansen, Sebastian Junges, and Thiago D Simao. Recursive small-step multi-agent a* for dec-POMDPs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

Wietze Koops, Sebastian Junges, and Nils Jansen. Approximate dec-POMDP solving using multi-agent A*. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2024. ISBN 978-1-956792-04-1. doi: 10.24963/ijcai.2024/745.

Jinlong Li, Runsheng Xu, Xinyu Liu, Jin Ma, Zicheng Chi, Jiaqi Ma, and Hongkai Yu. Learning for vehicle-to-vehicle cooperative perception under lossy communication. *IEEE Transactions on Intelligent Vehicles*, 8(4):2650–2660, 2023. doi: 10.1109/TIV.2023.3260040.

Minglong Li, Wenjing Yang, Zhongxuan Cai, Shaowu Yang, and Ji Wang. Integrating decision sharing with prediction in decentralized planning for multi-agent coordination under uncertainty. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 450–456, 2019. ISBN 9780999241141.

Qihan Liu, Jianing Ye, Xiaoteng Ma, Jun Yang, Bin Liang, and Chongjie Zhang. Efficient multi-agent reinforcement learning by planning. In *International Conference on Learning Representations (ICLR)*, 2023.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6382–6393, 2017.

Robert J. Moss, Anthony Corso, Jef Caers, and Mykel J. Kochenderfer. BetaZero: Belief-State Planning for Long-Horizon POMDPs using Learned Approximations. In *Reinforcement Learning Conference (RLC)*, 2024.

Frans A. Oliehoek, Matthijs T. J. Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32(1):289–353, 2008.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*. Springer, 2016.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(1), 2020.

Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanský, Daniel Heřt, Matěj Petrlík, Tomáš Báča, Vojěch Spurný, François Pomerleau, Vladimír Kubelka, Jan Faigl, Karel Zimmermann, Martin Saska, Tomáš Svoboda, and Tomáš Krajník. Darpa subterranean challenge: Multi-robotic exploration of underground environments. In *International Conference o n Modelling and Simulation for Autonomous Systems (MESAS)*, pp. 274–290, Berlin, Heidelberg, 2019. Springer-Verlag. ISBN 978-3-030-43889-0. doi: 10.1007/978-3-030-43890-6_22.

Maira Saboia, Lillian Clark, Vivek Thangavelu, Jeffrey A. Edlund, Kyohei Otsu, Gustavo J. Correa, Vivek Shankar Varadharajan, Angel Santamaria-Navarro, Thomas Touma, Amanda Bouman, Hovhannes Melikyan, Torkom Pailevanian, Sung-Kyun Kim, Avak Archanian, Tiago Stegun Vaquero, Giovanni Beltrame, Nils Napp, Gustavo Pessin, and Ali-akbar Agha-mohammadi. Achord: Communication-aware multi-robot coordination with intermittent connectivity. *IEEE Robotics and Automation Letters*, 7(4):10184–10191, 2022. doi: 10.1109/LRA.2022.3193240.

Pedro P. Santos, Diogo S. Carvalho, Miguel Vasco, Alberto Sardinha, Pedro A. Santos, Ana Paiva, and Francisco S. Melo. Centralized training with hybrid execution in multi-agent reinforcement learning via predictive observation imputation. *Artif. Intell.*, 348(C), November 2025. doi: 10.1016/j.artint.2025.104404.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. doi: 10.1126/science.aar6404.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Individualized controlled continuous communication model for multiagent cooperative and competitive tasks. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=rye7knCqK7.

Jianyu Su, Stephen Adams, and Peter Beling. Value-decomposition multi-agent actor-critics. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pp. 11352–11360, 2021. doi: 10.1609/aaai.v35i13.17353. URL https://ojs.aaai.org/index.php/AAAI/article/view/17353.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999. doi: https://doi.org/10.1016/S0004-3702(99)00052-1.

Daniel Szer, François Charpillet, and Shlomo Zilberstein. Maa*: a heuristic search algorithm for solving decentralized POMDPs. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 576–583, 2005. ISBN 0974903914.

Sizhe Tang, Jiayu Chen, and Tian Lan. MALinzero: Efficient low-dimensional search for mastering complex multi-agent planning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026. URL https://openreview.net/forum?id=Rxdon7jWni.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. {QPLEX}: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=Rcmk0xxIQV.

Isaac Ronald Ward, Mark Paral, Kristopher Riordan, Michelle Ho, Maximilian Adang, and Mykel J Kochenderfer. Prospector: a cave simulation environment for rotorcraft. https://github.com/isaac-ward/prospector, 2025.

Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 1433–1440. IEEE, 2016.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Advances in Neural Information Processing Systems (NeurIPS)*, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Chengcheng Zhao, Xiaoming Duan, Lin Cai, and Peng Cheng. Vehicle platooning with non-ideal communication networks. *IEEE Transactions on Vehicular Technology*, 70(1):18–32, 2021. doi: 10.1109/TVT.2020.3046165.

Zeyang Zhao, Huiwen Zuo, Qiang Shen, and Wei Wang. Event-triggered distributed model predictive control for spacecraft formation with collision avoidance. 59(20):905–910, 2025. doi: https://doi.org/10.1016/j.ifacol.2025.11.268. IFAC Symposium on Automatic Control in Aerospace.

Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, AAMAS '24, pp. 2845–2847, 2024. ISBN 9798400704864.

A Appendix

A.1 Extended Literature Review

Communication and hybrid execution in multiagent reinforcement learning. The MARLCOM literature (Zhu et al., 2024) augments decentralized policies with learned *explicit* communication mechanisms, including continuous differentiable messages, discrete protocols, gating, targeted communication, and attention-based aggregation. Representative methods such as DIAL (Foerster et al., 2016) learn communication channels jointly with policies, while later approaches such as IC3Net (Singh et al., 2019), TarMAC (Das et al., 2019), and ATOC (Jiang & Lu, 2018) learn when to communicate, whom to communicate with, or how to aggregate messages. Closest to our setting, MARO (Santos et al., 2025) studies *hybrid execution*, in which agents may encounter arbitrary communication levels at test time, ranging from fully decentralized to fully centralized execution, and proposes an autoregressive model for imputing missing observations under faulty or partial communication. These approaches improve coordination under partial observability and communication constraints, but they primarily learn message-conditioned policies or missing-observation predictors under a prescribed communication process.

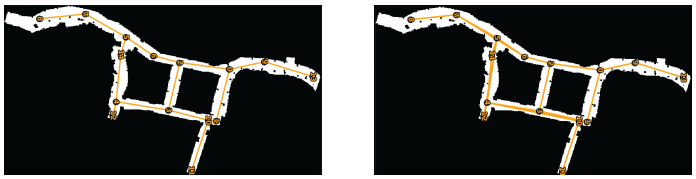
A.2 Code Repository

The supplementary archive contains the full implementation underlying all reported results. SDecMCTS and SDecZero are implemented in Julia (`julia/`), with Flux used for the learned policy, value, and communication heads. Benchmarks and cave graphs are provided in `benchmarks/`, and all learning baselines and launch scripts are in `baselines/`. A single entry point, `scripts/reproduce_paper.sh`, reproduces every experiment in the paper and rebuilds the figures from the stored outputs in `results/`, including training logs, evaluation traces, and checkpoints. Runs are deterministic given the seed and thread count. The archive is released under the MIT license, and the repository will be made public on acceptance.

A.3 Additional Benchmark Details

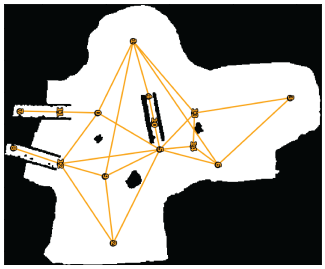
Figure 8 shows the paths flown by both quadrotors in one trial per cave, which we describe in additional detail here. The paths illustrate how the node-level tours are realized as explorative, continuous, collision-free trajectories. In *Tunnels* (a), both agents stay within a 0.4 m height band as they cover the narrow pathways. Agent one covers the northeastern branches and agent two the ladder and the southwestern tunnel, demonstrating a 'divide and conquer' behavior. In *Chamber* (b), agent one sweeps the open central chamber at constant height, while agent two climbs 3 m to fly the raised northern passage to its end and back. *Grapevine* (c) is fully three-dimensional: agent one climbs a near-vertical shaft of about 9 m. Paths follow smooth arcs through junctions. Out-and-back legs appear as narrow loops because the outbound and return passes differ slightly. The flown paths are 1–7% longer than the straight line paths between the nodes in all cases.

Figure 9 indicates where in these caves the nodes are placed, and which nodes can communicate with each other.

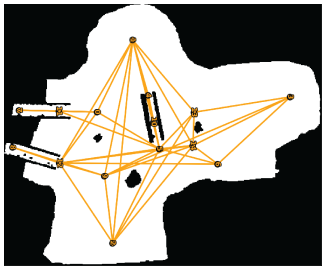


(a) *Tunnels* traversability graph.

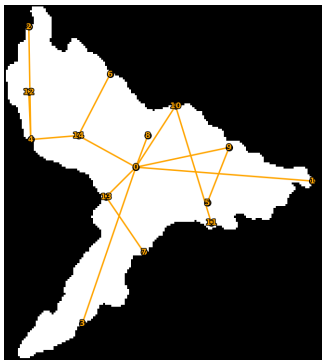
(b) *Tunnels* coordination graph.



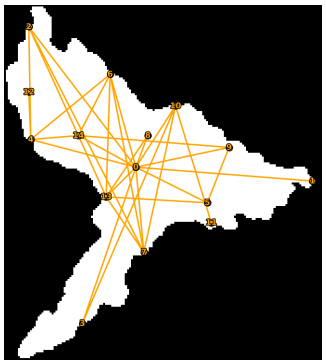
(c) *Chamber* traversability graph.



(d) *Chamber* coordination graph.



(e) *Grapevine* traversability graph.



(f) *Grapevine* coordination graph.

Figure 9: Fifteen node two-dimensional views of the traversability and coordination graphs for the DARPA Subterranean Challenge cave sites, *Tunnels* and *Chamber*, and for the *Grapevine* cave. Because *Grapevine* spans several levels, its connections appear denser. The traversability graph has an edge between two nodes if they are spatial neighbors, and the coordination graph has an edge between two nodes if they have a line-of-sight connection.

A.4 Baselines

MAPPO. Multi-Agent Proximal Policy Optimization (MAPPO) is a model-free policy gradient method that extends PPO to cooperative multiagent settings using centralized training with decentralized execution (CTDE) (Yu et al., 2022). Each agent executes a policy conditioned on its local observations, while a centralized critic can exploit global information during training to improve value estimation and coordination. MAPPO is very well-suited to cooperative tasks that require coordinated behavior among partially observing agents when global state information is available during training. MAPPO provides a strong model-free baseline for evaluating whether explicitly modeling intermittent communication provides benefits beyond standard CTDE.

IPPO. Independent Proximal Policy Optimization (IPPO) is a model-free method that applies PPO independently to each agent. It has policies and value functions that are learned from locally available information (de Witt et al., 2020). IPPO doesn't rely on a centralized critic, which makes it particularly natural to use for solving decentralized problems in which agents primarily act from their own observations, and where coordination requirements are relatively weak. Despite the simplicity

of the method, IPPO demonstrates strong performance on cooperative multiagent benchmarks and provides a very useful fully decentralized baseline for comparison.

QMIX. QMIX is a model-free, value-based MARL method that is designed for cooperative tasks under centralized training with decentralized execution (Rashid et al., 2020). QMIX learns a per-agent action-value function and in a multi-agent network, combines those action-value functions through a monotonic mixing network to represent a centralized joint-action value while preserving decentralized action selection. It is able to solve cooperative discrete-action problems with shared rewards in which coordinated behavior can be represented through this monotonic value factorization.

MASAC. Multi-Agent Soft Actor Critic (MASAC) is an off-policy actor-critic method that is model-free and extends maximum-entropy Soft Actor-Critic to the multi-agent setting. It does this by combining centralized information during training with decentralized policies at execution. The objective is entropy-regularized, which encourages exploration, while the replay buffer based learning reuses the previously collected experience (Haarnoja et al., 2018; Bettini et al., 2024). MASAC is particularly useful in cooperative multiagent problems where sample efficiency and exploration are important and acts as a strong off-policy counterpart to the on-policy IPPO and MAPPO baselines.

MAZero. MAZero is a model-based multi-agent reinforcement learning (MARL) method that extends MuZero style planning to cooperative multiagent problems. It combines a learned centralized dynamics model with Monte Carlo Tree Search (MCTS). The learned model allows for planning over joint actions without requiring the knowledge of environment dynamics, while more specialized search mechanics improve scalability in larger joint-action spaces (Liu et al., 2023). Therefore, MAZero is very well suited to solve cooperative problems in which explicit lookahead and coordinated joint-action planning are valuable. It also acts as a model-based baseline that is useful for distinguishing the benefits of semi-decentralization from those of learned model search alone.

SDec-MAZero. SDec-MAZero is a variant of MAZero that we adapt to the SDec-POMDP setting (Al-Husseini et al., 2026a). It retains MAZero's learned-model planning properties while being able to account for changes in each agent's ability to coordinate as communication becomes available/unavailable during execution (in a non-uniform time period). Unlike SDecZero which uses decentralized solvers, SDec-MAZero generates decentralized policies by querying trained decentralized policy heads. SDec-MAZero is suited to solve cooperative planning problems with intermittent communication states and provides a strong model-based comparison for evaluating SDecZero.

A.5 SDecZero Ablations

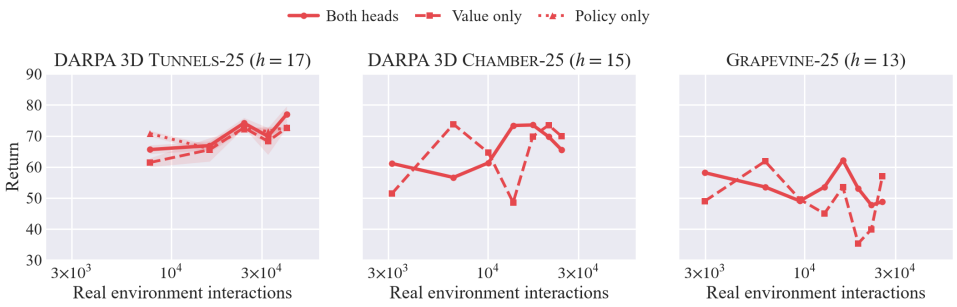


Figure 10: Evaluation-time head ablation of SDecZero on the 25-node labyrinths. Checkpoints from the known-communication run are re-evaluated on the 24-target slate at the run's depth cap with 1,000 simulations per decision using both heads, the value head only with uniform PUCT priors, or, on Tunnels-25, the policy head only with rollout-based values and SDecMCTS exit bounds. Either head alone remains close to the full planner, indicating that search can compensate for the missing learned signal. Tunnels-25 reports mean ± standard error over three protocol seeds. Chamber-25 and Grapevine-25 use one seed. Policy-only is omitted on the depth-4 maps because exact exit-bound evaluation is over an order of magnitude slower. Small differences from Fig. 6 arise from host-dependent tie-breaking.

A.6 Experiment Hyperparameters

All runs used one workstation (AMD Ryzen 9 9900X3D, 12 cores, 62 GB RAM; CPU only). SDecMCTS timings are cold: one process per table row, with compilation and model construction inside the measured time. Every labyrinth evaluation is the fixed-target slate, one episode per target node under deterministic dynamics. Trial k plans with protocol seed $74 + k$, so reported means also average over the planner's own randomness.

Table 3: SDecMCTS search settings. Where three values are given they belong to the three window regimes, keyed on the benchmark's state count and observation noise: small ($\leq 10^4$ states: tabular benchmarks and small labyrinths), large ($> 10^4$ states: 15-node maps) and Search & Rescue (stochastic observations).

Simulations per decision	3,000 labyrinths; 2,000 Mars and Search & Rescue; 500 Tiger and MEDEVAC
PUCT constant c; widening (k, α)	2.0; $(3.0,\ 0.1)$
Leaf heuristic	Q_{MDP} (Mars, MEDEVAC); POMDP relaxation (Tiger, labyrinths); one-step lookahead (Search & Rescue)
Belief representation	exact belief vectors
Window depth d	6 small; 4 large; 3 rescue
Boundary visits before a window solve	16 small; 16 large; 64 rescue
Exit price	exact centralized bound (small, large); V_{MDP} frontier (rescue)
Window solver	RS-MAA* with the recursive heuristic, iteration limit $L = 100$, lossless clustering
Trials per row	100 episodes (tabular); 5 slates (labyrinths); 3 slates (15-node maps)

Table 4: SDecZero settings on the 25-node labyrinths. Where two values are given they are TUNNELS-25 / CHAMBER-25 and GRAPEVINE-25.

Search	
Simulations per synchronized decision	1,000
PUCT constant c; prior	2.0; policy head
Window depth d; visits before a solve	6 / 4; 2
Window solver	RS-MAA*, recursive heuristic, $L = 100$
Leaf, blackout-node and exit values	value head
Self-play	
Episodes per generation; generations	16 (targets cycled); 18 / 8
Root Dirichlet noise	$\alpha = 0.3$, fraction 0.25
Action selection	first synchronized decision sampled from root visits, then most visited
Exit continuations per episode	32 / 16, drawn uniformly from the episode's priced exits
Targets	policy: root visit distribution; value: Monte-Carlo return
Network (one per map)	
Input	14 node features + node one-hot; 8 global features
Encoder; message passing	2×64 ReLU; 5 rounds of width 64 over the normalized adjacency
Trunk	2×128 ReLU on the pooled readout
Heads	policy: linear over legal joint actions; value: target-node embedding and trunk $\rightarrow 128 \rightarrow$ (synchronized, boundary)
Optimization	
Optimizer	Adam, lr 5×10^{-4}, weight decay 10^{-5}, gradient-norm clip 5
Steps per generation; batch; buffer	100; 64; 20,000 most recent samples
Loss	policy cross-entropy + value squared error
Learned communication head	
Model; fit	pair head, embedding 8; class-weighted BCE, 200 epochs, lr 0.05, weight decay 10^{-4}, refit every generation on executed transitions
Use in planning	communication predicted where the head exceeds 0.5
Evaluation	every generation: 24 targets, one greedy episode each, 1,000 simulations

Table 5: Baseline settings on the 25-node labyrinths. Model-free methods run in BenchMARL; unlisted values are the library defaults.

Model-free (MAPPO, IPPO, QMIX, MASAC)	
Budget; seeds	600,000 environment frames; 3
Evaluation	every 12,000 frames: 24 targets $\times$ 5 episodes, greedy policy
Network	GRU (256) over augmented observations + MLP $[256, 256]$, tanh; parameters shared across agents
Discount; optimizer	$\gamma = 1.0$; Adam, lr 5×10^{-5}, gradient-norm clip 5
MAPPO, IPPO	clip 0.2; GAE $\lambda = 0.95$; entropy coefficient $0.05 \rightarrow 0.01$; 2,400 frames per batch, 8 minibatch iterations of 1,200; MAPPO critic on the factored global state
QMIX	mixing embedding 32; TD($\lambda = 0.95$); replay 100,000; batch 1,200; 16 updates per 2,400 frames; ϵ-greedy $0.8 \rightarrow 0.01$; Polyak $\tau = 0.005$
MASAC	twin critics; target-entropy weight 0.5; $\alpha_0 = 1.0$, $\alpha_{\min} = 0.05$; replay, batch and updates as QMIX
SDec-MAZero (belief mode; our modification of MAZero)	
Search	200 simulations per step; 25 sampled joint actions; belief tree with mass pruning $\epsilon = 3 \times 10^{-3}$
Training	3,000 updates $\times$ 6 transitions (18,000 transitions); batch 64; lr 5×10^{-4}; unroll 3; TD steps $= H$; target network every 200 updates; per-target value head; 1 seed
Evaluation	every checkpoint: 24 targets $\times$ 5 rollouts, 200 simulations
MAZero (learned model)	
Search	10 simulations per step; 25 sampled joint actions
Training	25,000 updates over 150,000 transitions; batch 64; lr 5×10^{-4}; TD steps $= H$; 1 seed
Evaluation	24 targets $\times$ 5 rollouts