Communication via Shared Memory Improves Multi-agent Pathfinding

Alsu Sagirova[1,2] **Yuri Kuratov**[1,2] **Mikhail Burtsev**[3]

[1]AIRI, Moscow, Russia [2]Neural Networks and Deep Learning Lab, MIPT, Dolgoprudny, Russia
[3]London Institute for Mathematical Sciences, London, UK
{alsu.sagirova, yurii.kuratov}@phystech.edu, mb@lims.ac.uk

Abstract

Multi-agent systems are gaining attention in AI research due to their ability to solve complex problems in a distributed manner, but coordinating multiple agents remains challenging. Inspired by the global workspace theory, we introduce the Shared Recurrent Memory Transformer (SRMT).[1] SRMT extends memory transformers to multi-agent settings by pooling and globally broadcasting individual working memories, enabling agents to communicate and coordinate their behavior implicitly. We evaluate SRMT on the Partially Observable Multi-Agent Path Finding in a bottleneck navigation task requiring agents to pass through a narrow corridor. SRMT consistently outperforms a range of reinforcement learning baselines, especially under challenging reward structures with sparse feedback. Our experiments also demonstrate that SRMT generalizes effectively to environments with significantly longer corridors than those seen during training, highlighting its scalability and robustness. These results suggest incorporating shared memory structures into transformer-based architectures can enhance communication and coordination in decentralized multi-agent systems.

1 Introduction

Multi-agent systems have significant potential to solve complex problems through distributed intelligence and collaboration. However, coordinating the interactions between multiple agents remains challenging, often requiring sophisticated communication protocols and decision-making mechanisms. We propose a novel approach to address this challenge by introducing a *shared memory* as a global workspace for agents to coordinate behavior. The global workspace theory [Baars, 1988] suggests that in the brain, there are independent functional modules that can cooperate by exchanging information through a global workspace. Inspired by this theory, we consider the agents in MAPF as independent modules with shared memory and propose a *Shared Recurrent Memory Transformer* (SRMT) as a mechanism to learn emergent task-dependent communication to improve coordination and avoid deadlocks. SRMT extends memory transformers [Burtsev et al., 2020, Bulatov et al., 2022, Yang et al., 2022, Cherepanov et al., 2024] to multi-agent settings by pooling and globally broadcasting individual working memories.

In this study, we test the shared memory approach on Partially Observable Multi-agent Path Finding (PO-MAPF) [Stern et al., 2019], where each agent has to reach its goal while observing the state of the environment, including locations and actions of the other agents and/or static obstacles, only locally. Multi-agent pathfinding can be considered in the *decentralized* manner, where each agent independently collects rewards and makes decisions on its actions. There are many attempts to solve

[1]https://github.com/Aloriosa/srmt

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

this problem: in robotics [Van den Berg et al., 2008, Zhu et al., 2022], in machine and reinforcement learning field [Damani et al., 2021, Ma et al., 2021b, Wang et al., 2023, Sartoretti et al., 2019, Riviere et al., 2020]. Such methods often involve manual reward-shaping and external demonstrations. Also, several works utilize the communication between agents to solve decentralized MAPF [Ma et al., 2021a, Li et al., 2022, Wang et al., 2023]. Yet still, the resulting solutions are prone to deadlocks and poorly generalizable to the maps not used for training. In this work, we compare SRMT to a range of RL baselines and show that it consistently outperforms them in the bottleneck navigation task.

2 Shared Recurrent Memory Transformer

In SRMT, the input sequence for each agent at each time step is constructed by combining three key components: the agent's personal memory vector; the historical sequence of the agent's observations from the past $h = 8$ time steps; and the current step's observation. This sequence undergoes a full self-attention mechanism. Next, the output of the self-attention is passed through a cross-attention layer between current hidden representations and shared memory. The shared memory consists of a globally accessible, ordered sequence of all agents' memory vectors for the current time step. This interaction between personal and shared memory enables each agent to incorporate global context into its decision-making process. The resulting output is then passed through a memory head, which updates the agent's personal memory vector for the next time step. Simultaneously, the output is also fed into the decoder part of the policy model, which generates the agent's action (Fig. 1A).

We use POGEMA [Skrynnik et al., 2024a] framework for the experiments. In POGEMA the two-dimensional environment is represented as a grid composed of obstacles and free cells. At each time step each agent can perform actions of two types: moving to an adjacent cell or remaining at their current position. Agents have limited observational capabilities, perceiving other agents only within a local 5×5 area centered on their current position. The episode ends when the predefined time step, episode length, is reached. The episode can also end before this time step if certain conditions are met, i.e. all agents reach their goal locations.

As an initial test of our approach, we selected a simple two-agent coordination task where the agents must navigate through a narrow passage. The environment consists of two rooms connected by a one-cell-wide corridor, as illustrated in Fig. 1B. Each agent starts in one room, and their goal is located in the opposite room, requiring both agents to pass through the narrow corridor to complete the task. To train the model, we utilized 16 maps with corridor lengths uniformly selected between 3 and 30 cells.

To evaluate the performance, we utilize three metrics: *Cooperative Success Rate (CSR)* – a binary measure indicating whether all agents reached their goals before the episode's end; *Individual Success Rate (ISR)* – the fraction of the agents that achieved their goals during the episode; and *Sum-of-Costs (SoC)* – the total number of time steps taken by all agents to reach their respective goals (the lower the value the better). The policy function is approximated with a deep neural network (see Appendix Fig. 4). The agents are assumed to be homogeneous, so the policy is shared between agents during training. To benchmark the effectiveness of SRMT, we compared it against several baselines: recurrent memory transformer without memory sharing (RMT), transformer without memory (Attention), direct connection of spatial encoder to actor-critic action decoder (Empty), GRU RNN (RNN). Among these baselines, RMT, Attention core and Empty core methods can be considered as ablations of SRMT architecture. We also trained and evaluated MAMBA [Egorov and Shpilman, 2022] and QPLEX [Wang et al., 2020] as baselines. Additional details regarding the training procedure can be found in Appendix A.1.

3 Results and Discussion

We first evaluated SRMT against the baseline models using three variations of the reward function: *Directional*, *Moving Negative*, and *Sparse*.[2] In the Directional setting, the agent was rewarded for reaching the goal and for every step that brought it closer to the goal. In Moving Negative movements are slightly penalized for minimizing the path to the goal but no information about the goal location is provided. In contrast, the Sparse reward function only gives a reward when the agent successfully moves into the goal cell, offering no intermediate rewards.

[2]See Appendix A for details of rewards and extra results.

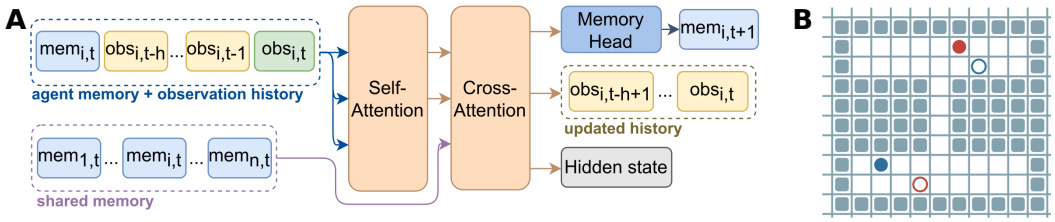

Figure 1: **Model and task overview.** A. Shared Recurrent Memory Transformer (SRMT) establishes a communication channel between recurrent memories $mem_{i,t}$ of individual agents at a moment t and provides global access to them via cross-attention. B. In the *Bottleneck Task* two agents start in rooms opposite to their goals and should coordinate passing the corridor. Agents are shown as solid colored circles; their goals are empty circles with the same colored border.

The comparison of evaluation scores, shown in Figure 2, demonstrates that SRMT successfully solves the task under considered reward functions. In particular, SRMT consistently performed well, even in the challenging Moving Negative and Sparse reward scenarios where other methods struggled. Considering the Sparse reward, the difference between SRMT and RMT scores is statistically significant (Wilcoxon signed-rank test, $p < 0.001$). The ability to coordinate across agents via shared memory proved critical, especially when feedback from the environment was minimal.

Among the non-memory sharing methods, RMT achieved the best performance, outperforming both the GRU-based RNN and the vanilla transformer models. This indicates that the combination of self-attention with recurrence in RMT offers distinct advantages in multi-agent coordination, particularly in tasks that require sequential decision-making over time. However, without the shared memory, RMT's performance in the Sparse reward setting was still limited compared to SRMT, highlighting the importance of global information exchange.

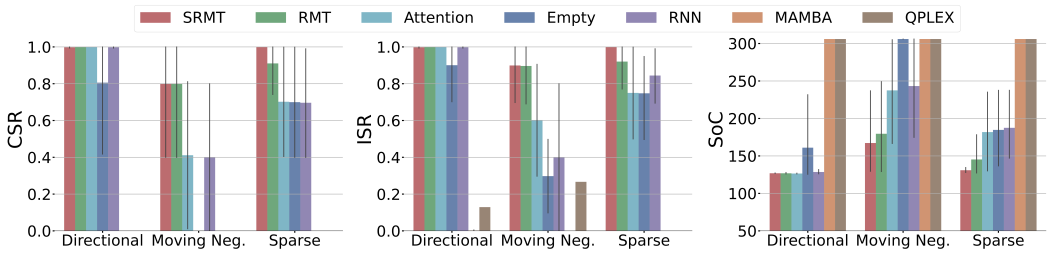

Figure 2: **SRMT effectively solves the Bottleneck Task with different reward functions.** Trained with Directional (positive when moved towards a goal and achieved it) reward, all the methods except Empty core policy, MAMBA, and QPLEX solve the task. For the case with the negative reward for movement and no directional reward (Moving Negative) memory-based methods (SRMT and RMT) demonstrate a clear advantage over memory-less (Attention, Empty, RNN). With the Sparse (only on-goal) reward, SRMT maintains the score while other methods drop. Error bars indicate 95% confidence intervals. For CSR and ISR, higher values are better, for SoC - the lower the better.

To further assess the generalization capabilities of the trained policies, we evaluated them on bottleneck environments with corridor lengths significantly larger than those used during training, ranging from 5 to 1000 cells. The evaluation results[3] for the Sparse and Moving Negative reward functions are shown in Figure 3.

The results indicate that trained with the Sparse reward function, the SRMT-based policy consistently outperforms the baseline methods regarding the individual success rate (ISR) and the total time spent in the environment (SoC). Remarkably, SRMT scales effectively up to corridor lengths of 400 cells without any significant performance degradation. For corridor lengths beyond 400 steps, the Cooperative Success Rate (CSR) of SRMT drops from 1.0 to 0.8, which still surpasses the other models, except for RMT. RMT shows the highest stability across all tested corridor lengths,

[3]For these evaluations, we adjusted the episode length to $2 \cdot \textit{corridor\_len} + 100$ to ensure that both agents had sufficient time to reach their goals within each episode.

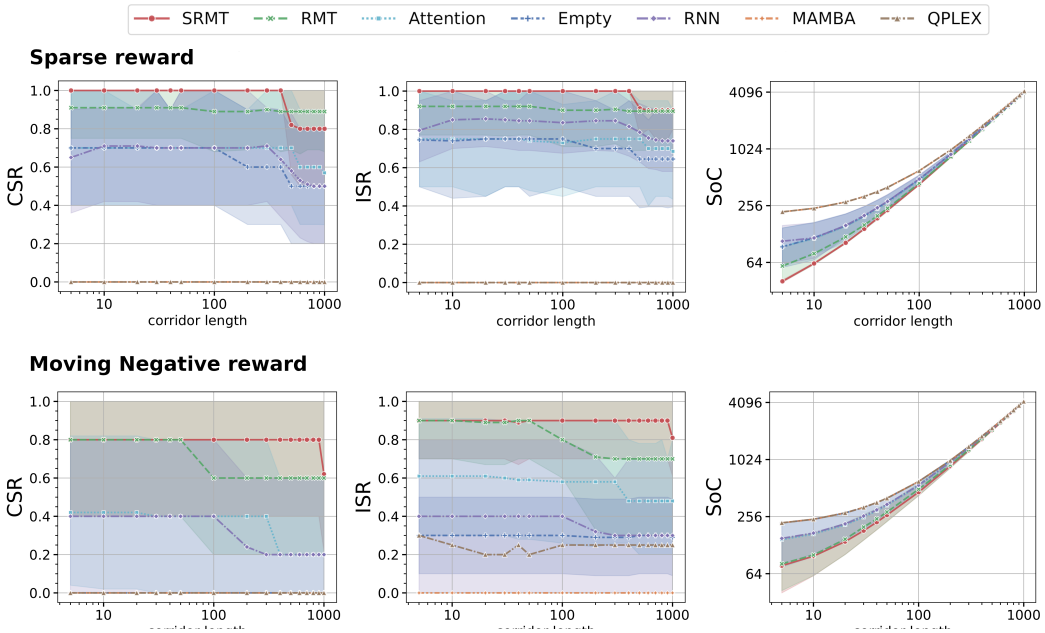

Figure 3: **SRMT agents generalize on corridor lengths up to 1000.** After training on corridor sizes from 3 to 30 cells all methods were evaluated on longer passages up to 1000. All models show good scaling till 100 while memory-based transformers were able to scale beyond that. For the Sparse reward, SRMT leads up to 400 and then drops below RMT for collective performance. For the Moving Negative reward, SRMT shows the top-1 performance on all three metrics. The shaded area indicates 95% confidence intervals.

maintaining a score of 0.9 for CSR across the board. For the Moving Negative reward function, resulting policies with SRMT show the top-1 scores for any corridor length up to 1000 (see Fig. 3). Evaluation scores for the other reward functions and analysis of memory representations can be found in Appendix A.

4 Conclusion

In this paper, we introduced the Shared Recurrent Memory Transformer (SRMT) as a novel architecture for enhancing communication and coordination in multi-agent systems. SRMT enables agents to learn how to exchange information implicitly and coordinate actions without explicit communication protocols. Our experimental results on the bottleneck navigation task demonstrate that SRMT consistently outperforms baseline models, especially in challenging scenarios with sparse rewards and extended corridor lengths. The shared memory mechanism allows agents to generalize their learned policies to environments with significantly longer corridors than those seen during training, demonstrating the scalability and robustness of our approach. These findings highlight the potential of incorporating shared memory structures in transformer-based architectures for multi-agent reinforcement learning.

Limitations

As in the majority of research related to Multi-Agent Path Finding (MAPF), in this work we assume that the agents have flawless localization and mapping abilities. Our primary focus is on the decision-making aspect of the problem. We also consider that the agents execute actions accurately and that their moves are synchronized. Additionally, we treat obstacles as fixed elements of the environment.

Finally, it's important to note that our approach, like other prominent learnable methods designed for (PO)-MAPF – such as PRIMAL [Sartoretti et al., 2019], PRIMAL2 [Damani et al., 2021], DHC [Ma et al., 2021a], and PICO [Li et al., 2022] – does not offer theoretical guarantees that agents will

reach their destinations. However, extensive experimental evidence, both from our work and from the referenced studies, demonstrates that these learnable methods are practically powerful and scalable solutions for complex MAPF problems.

Acknowledgments and Disclosure of Funding

This work was partially supported by a grant for research centers, provided by the Analytical Center for the Government of the Russian Federation in accordance with the subsidy agreement (agreement identifier 000000D730324P540002) and the agreement with the Moscow Institute of Physics and Technology dated November 1, 2021 No. 70-2021-00138.

References

B. J. Baars. A Cognitive Theory of Consciousness. Cambridge University Press, New York, 1988.

A. Bulatov, Y. Kuratov, and M. Burtsev. Recurrent memory transformer. Advances in Neural Information Processing Systems, 35:11079–11091, 2022.

M. S. Burtsev, Y. Kuratov, A. Peganov, and G. V. Sapunov. Memory transformer. arXiv preprint arXiv:2006.11527, 2020.

E. Cherepanov, A. Staroverov, D. Yudin, A. K. Kovalev, and A. I. Panov. Recurrent action transformer with memory, 2024. URL `https://arxiv.org/abs/2306.09459`.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555, 2014.

M. Damani, Z. Luo, E. Wenzel, and G. Sartoretti. Primal _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong. IEEE Robotics and Automation Letters, 6(2):2666–2673, 2021.

V. Egorov and A. Shpilman. Scalable multi-agent model-based reinforcement learning. arXiv preprint arXiv:2205.15023, 2022.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

W. Li, H. Chen, B. Jin, W. Tan, H. Zha, and X. Wang. Multi-agent path finding with prioritized communication learning. 2022 International Conference on Robotics and Automation (ICRA), pages 10695–10701, 2022.

Z. Ma, Y. Luo, and H. Ma. Distributed heuristic multi-agent path finding with communication. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 8699–8705. IEEE, 2021a.

Z. Ma, Y. Luo, and J. Pan. Learning selective communication for multi-agent path finding. IEEE Robotics and Automation Letters, 7(2):1455–1462, 2021b.

A. Petrenko, Z. Huang, T. Kumar, G. S. Sukhatme, and V. Koltun. Sample factory: Egocentric 3d control from pixels at 100000 FPS with asynchronous reinforcement learning. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of Proceedings of Machine Learning Research, pages 7652–7662. PMLR, 2020. URL `http://proceedings.mlr.press/v119/petrenko20a.html`.

B. Riviere, W. Hönig, Y. Yue, and S.-J. Chung. Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning. IEEE Robotics and Automation Letters, 5 (3):4249–4256, 2020.

G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. S. Kumar, S. Koenig, and H. Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. IEEE Robotics and Automation Letters, 4 (3):2378–2385, 2019.

A. Skrynnik, A. Andreychuk, A. Borzilov, A. Chernyavskiy, K. Yakovlev, and A. Panov. Pogema: A benchmark platform for cooperative multi-agent navigation, 2024a. URL https://arxiv.org/abs/2407.14931.

A. Skrynnik, A. Andreychuk, M. Nesterova, K. Yakovlev, and A. Panov. Learn to follow: Decentralized lifelong multi-agent pathfinding via planning and learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 38, pages 17541–17549, 2024b.

R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. S. Kumar, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In Proceedings of the 12th Annual Symposium on Combinatorial Search (SoCS 2019), pages 151–158, 2019.

J. Van den Berg, M. Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In Proceedings of The 2008 IEEE International Conference on Robotics and Automation (ICRA 2008), pages 1928–1935. IEEE, 2008.

J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang. Qplex: Duplex dueling multi-agent q-learning. arXiv preprint arXiv:2008.01062, 2020.

Y. Wang, B. Xiang, S. Huang, and G. Sartoretti. Scrimp: Scalable communication for reinforcement- and imitation-learning-based multi-agent pathfinding. In 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 9301–9308. IEEE, 2023.

Y. Yang, G. Chen, W. Wang, X. Hao, J. Hao, and P.-A. Heng. Transformer-based working memory for multiagent reinforcement learning with action parsing. Advances in Neural Information Processing Systems, 35:34874–34886, 2022.

H. Zhu, B. Brito, and J. Alonso-Mora. Decentralized probabilistic multi-robot collision avoidance using buffered uncertainty-aware voronoi cells. Autonomous Robots, 46(2):401–420, 2022.

A Appendix

A.1 Training details

The environments were created with POGEMA Skrynnik et al. [2024a] framework. The Sample Factory codebase [Petrenko et al., 2020] was used for policy model training.

The policy neural network is presented in Figure 4. We adopt the Skrynnik et al. [2024b] approach for configuring the model and input-output data pipelines. The model input is the agent's local observation tensor of shape $3 \times m \times m$ tensor, where m is the observation range. The channels of the tensor encode the static obstacle locations combined with the current path, the other agents, and their targets.

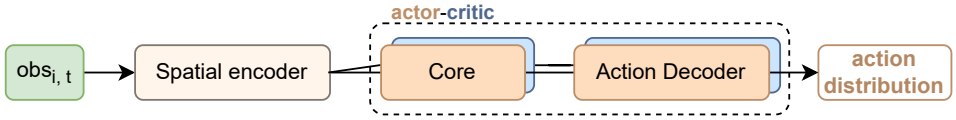

Figure 4: **Learnable policy architecture.**. The actor-critic consists of the core subnetwork and action decoder.

The spatial encoder is a ResNet [He et al., 2016] with an additional Multi-Layer Perceptron (MLP) in the output layer. The Action Decoder and the Critic Head are Dense layers. As the core sub-network, we consider the GRU architecture [Chung et al., 2014] or the attention block implemented in the Huggingface GPT-2[4] model.

We apply no advanced heuristics or methods for path planning to test the impact of memory addition. Our path-planning strategy is simple: each agent aims to follow the shortest path to the goal at each time step. If according to the planned movements the agents are to collide, their final decision will be to retain their current positions until the next step. We hypothesize that a shared memory mechanism will help to solve such bottleneck problems.

Training parameters for all tested methods are listed in Table 1. A single Tesla P100 was used for training policy models for approximately 1 hour each. The results for models trained with Sparse and Dense reward functions were averaged over 10 runs with different random seeds. The results of training policies with Directional and Directional Negative rewards were averaged over 5 runs with different random seeds as they showed less variation during training. Each run evaluation was first averaged over 10 different evaluation procedure random seeds. We carried out the grid search for the SRMT training entropy coefficient (range $[0.00001, 0.0003]$) and learning rate (range $[0.01, 0.05]$).

A.2 Evaluation scores

In this section, we provide the evaluation results for Dense, Directional, and Directional Negative reward functions. The Figures 5, 6, 7 have superior or comparable performance compared to the baselines.

A.3 Memory Analysis

We also explored the relations between the SRMT agents' memory representations and the spatial distances between agents on the map. Fig. 8 shows that SRMT distances between memory representations are aligned with distances between agents for different corridor lengths. Starting the episode, the agents move closer to each other quickly, and the respective cosine distances decrease significantly. Then, agents face each other in the environment (marked with a triangle on Fig. 8) and move in the same direction, keeping the spatial distance between them constant. Next, after the moment when one of the agents reaches its goal and disappears from the environment (marked with a star), the other agent moves away to reach the goal. This part of the episode is depicted as increasing memory distance at the end of the episode.

[4] `https://huggingface.co/docs/transformers/model_doc/gpt2`

Table 1: Training hyperparameters.

Parameter	Value
Optimizer	Adam
Learning rate	0.00013
LR Scheduler	Adaptive KL
γ (discount factor)	0.9716
Recurrence rollout	8
Clip ratio	0.2
Batch size	16384
Optimization epochs	1
Entropy coefficient	0.0156
Value loss coefficient	0.5
GAE_λ	0.95
MLP hidden size	16
ResNet residual blocks	1
ResNet filters	8
Attention hidden size	16
Attention heads	4
GRU hidden size	16
Activation function	ReLU
Network Initialization	orthogonal
Rollout workers	4
Envs per worker	4
Training steps	2×10^7
Episode length	512
Observation patch	5×5
Number of agents	2

Table 2: Tested reward functions. We list the reward values for achieving the goal, moving on the path toward the goal, or taking other actions (moving not in the direction of the goal and staying in the same position).

Type	On goal	Move towards goal	Else
Directional	+1	+0.005	0
Sparse	+1	0	0
Dense	+1	-0.01	-0.01
Directional Negative	+1	-0.005	-0.01
Moving Negative	+1	-0.01	-0.01 for moving, -0.005 for holding

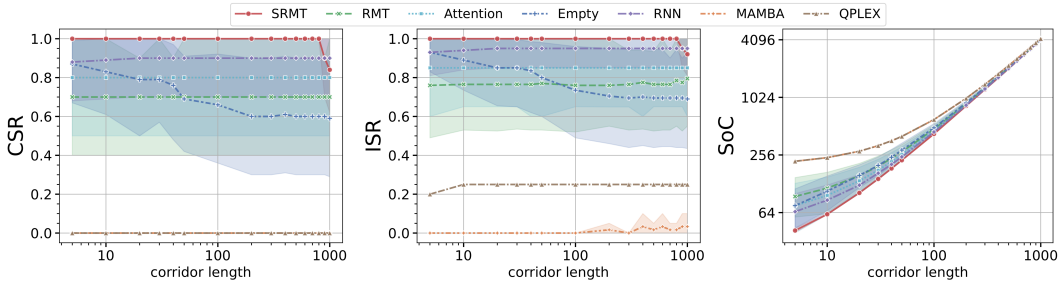

Figure 5: Trained with **Dense** reward, all models except empty core policy scale with enlarging corridor length. SRMT consistently outperforms baselines both in success rates and in the time needed to solve the task. The shaded area indicates 95% confidence intervals.

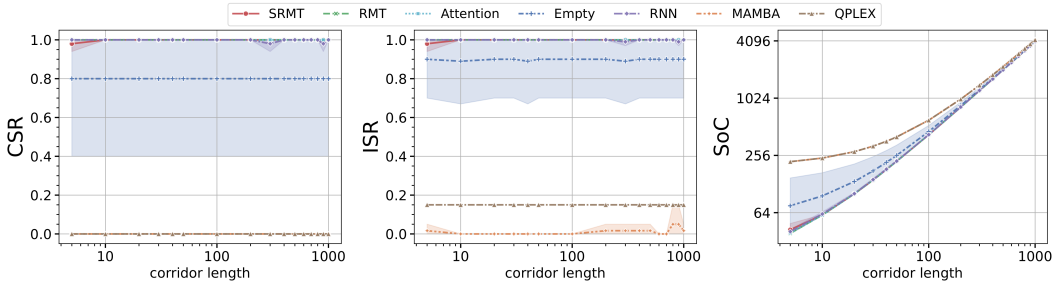

Figure 6: **Directional** reward training leads to all the methods preserving the scores for all tested corridor lengths. The shaded area indicates 95% confidence intervals.

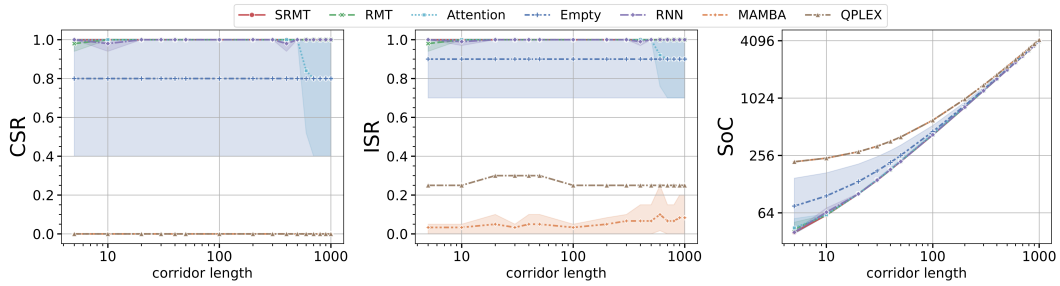

Figure 7: Results of training with **Directional Negative** reward. Vanilla attention fails to scale at corridor lengths of more than 400, compared to the SRMT which preserves the highest scores. That proves the sufficiency of the proposed SRMT architecture. The shaded area indicates 95% confidence intervals.

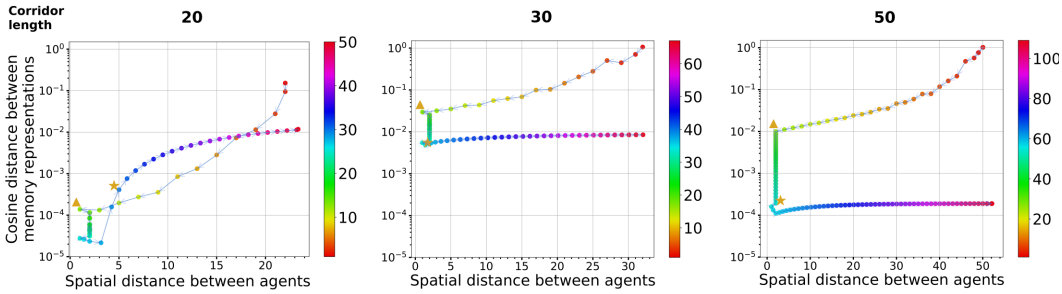

Figure 8: **SRMT memory distances are aligned with the distances between agents.** The figure shows how cosine distances between agents' memory vectors are related to the Euclidean distances between agents on the map for SRMT. The triangle marks the step when agents face each other in the environment, the star shows the episode step when the first goal was achieved. The color bar shows the step number.

We also provide the heatmaps of cosine distances between agent memory states during the episode. Figures 9 and 10 show how SRMT memory for states are related to each other. Figures 11 and 12 depict the paired distances between memory vectors on different time steps in the episode for both agents.

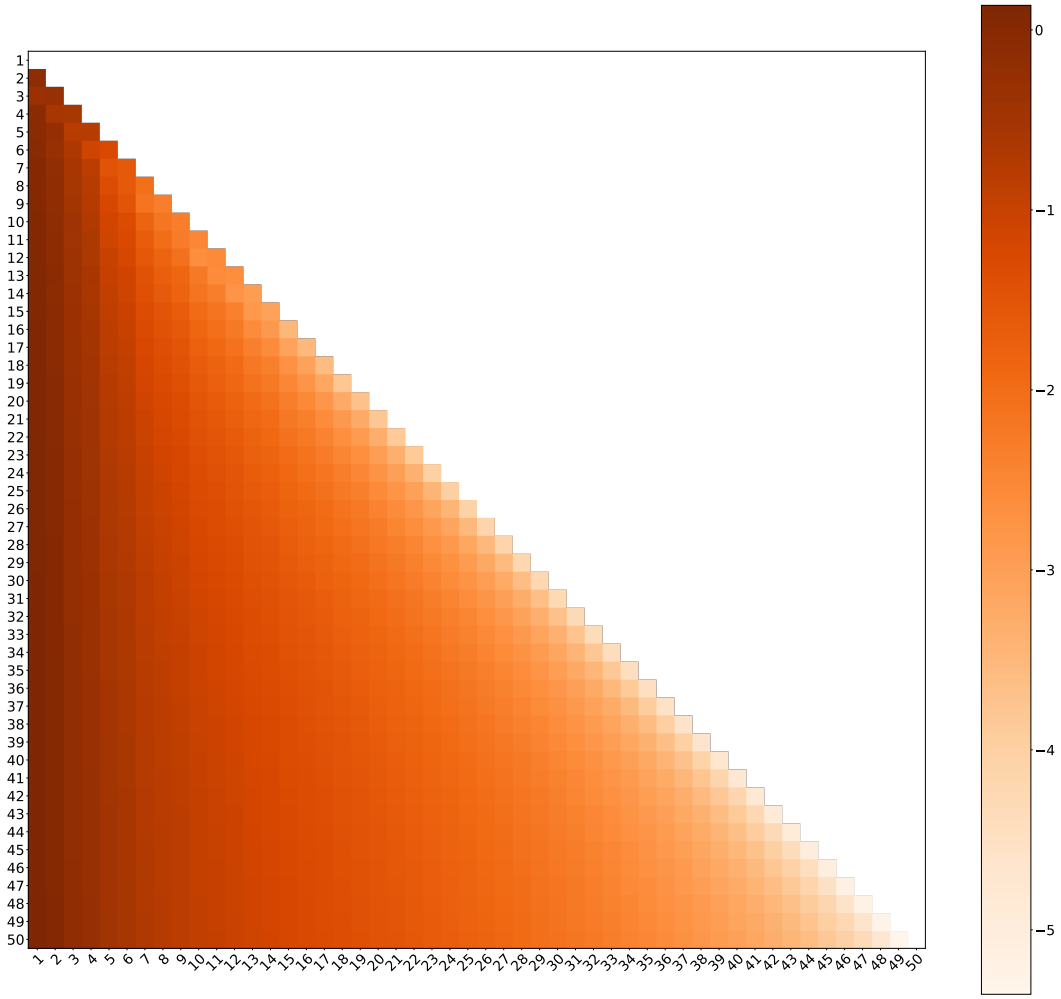

Figure 9: SRMT agent 1 heatmap of distances between memory states.

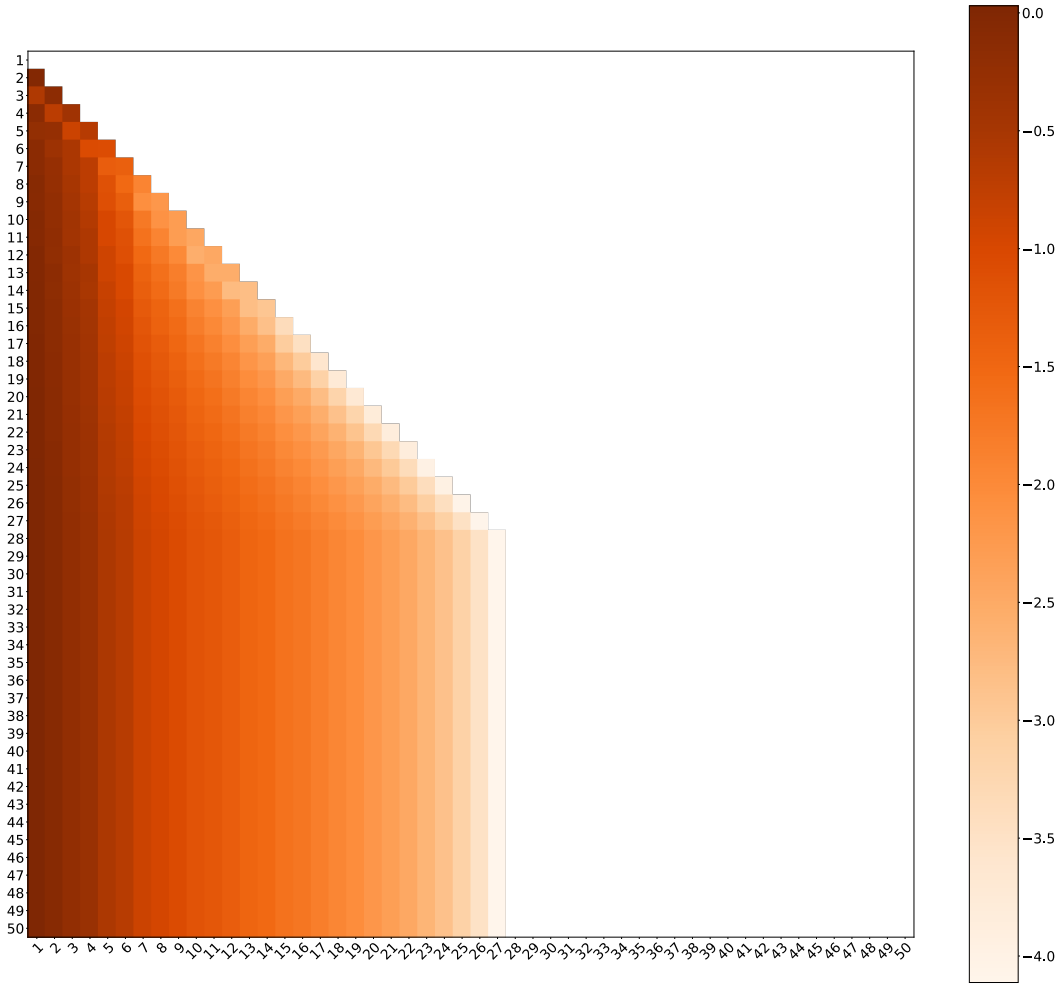

Figure 10: SRMT agent 2 heatmap of distances between memory states.

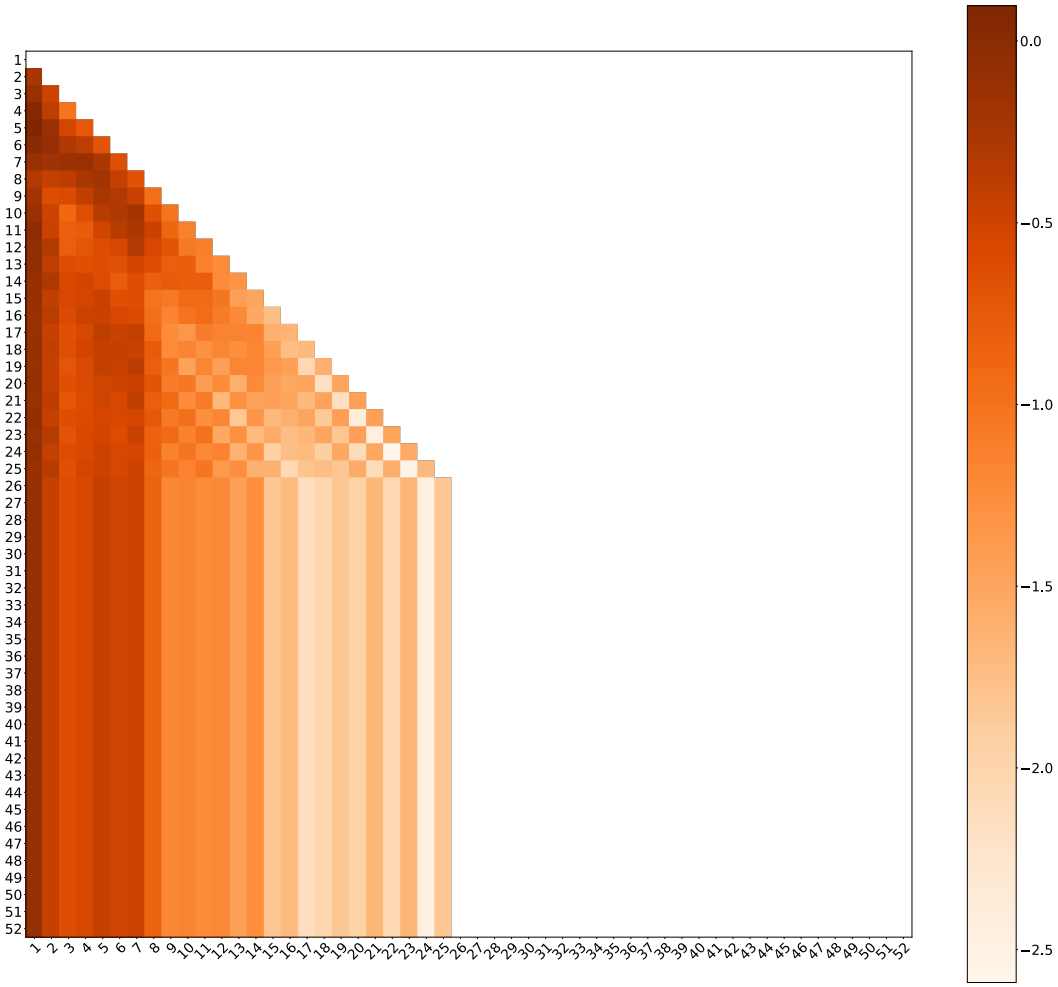

Figure 11: Agent 1 with RMT policy heatmap of distances between memory states.

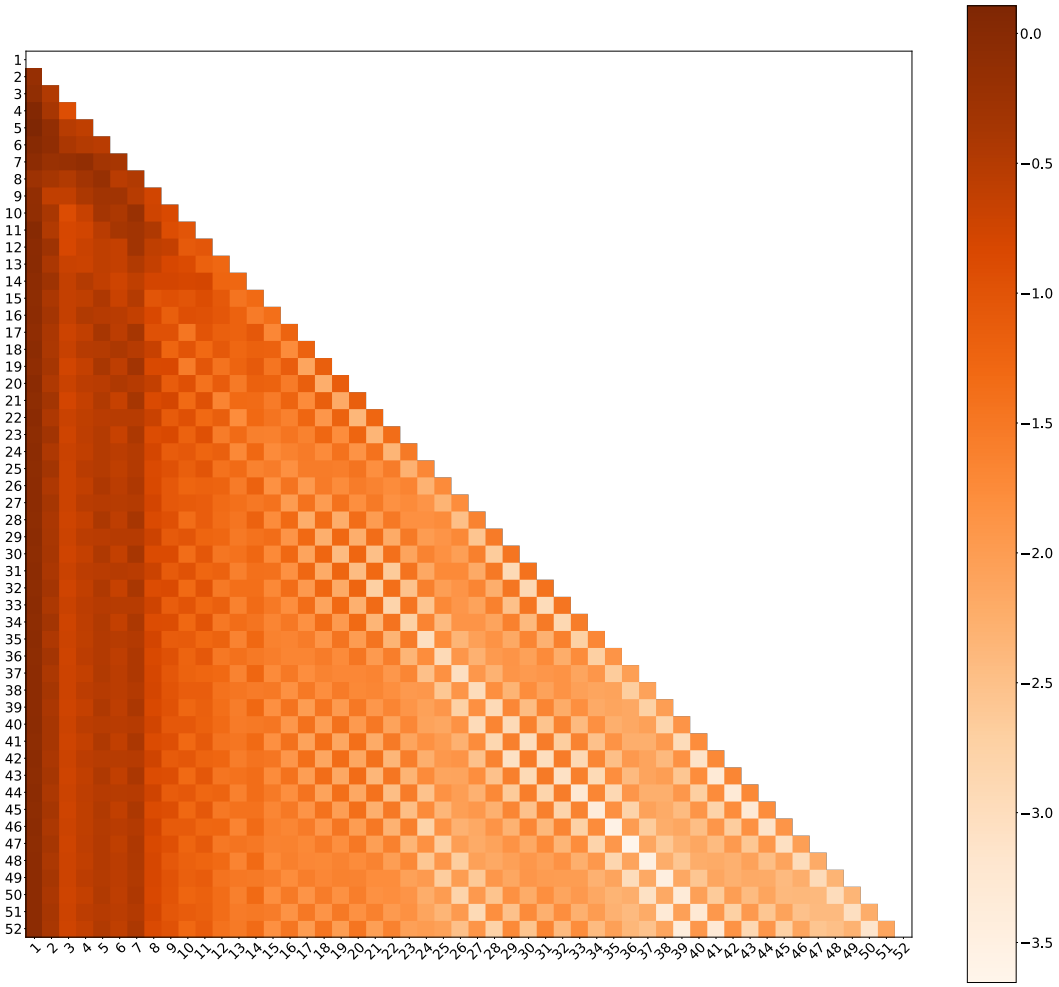

Figure 12: Agent 1 with RMT policy heatmap of distances between memory states.