2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)
University of Southern California, LA, USA, December 4-5, 2017

Memory-Augmented Multi-Robot Teams That Learn To Adapt

Shauharda Khadka, Jen Jen Chung and Kagan Tumer

Abstract— **As multi-robot teams increasingly permeate real world environments like factories, homes and other extraterrestrial surfaces, the ability to form joint strategies that can effectively adapt to new observations and changes in teammates' policy becomes more vital. Memory is a crucial part of adaptive behavior since it is the mechanism by which knowledge from past observations can be used to modify future behavior. Previous works have studied adaptive behaviors primarily with the tools of memory but have largely been limited to single robot approaches. In this paper, we formulate the Extended T-Maze domain that marries the difficulties of a task requiring explicit adaptive behavior, to the complexities introduced by concurrent actions of multiple robots acting in the same environment. We develop a memory-based learning approach using Gated Recurrent units with Memory Block as robot policies, and demonstrate its efficacy in diverse sets of experiments in the Extended T-Maze domain that vary across multiple axes of difficulty including team size and task depth. Our results show that the memory-based multi-robot controllers are able to form effective joint strategies significantly outperforming feedforward controllers trained without memory.**

I. INTRODUCTION

Autonomous multi-robot teams can accomplish complex tasks in highly dynamic and stochastic environments improving on both speed and effectiveness over single robot approaches. However, multi-robot coordination is a complex control problem especially when the task requires adaptive behaviors from the group of coordinating robots.

The requirement for adaptive behavior imposed by the task definition institutes an added layer of complexity in learning. This complexity is orthogonal to the one imposed by the need for adaptation to each other, inherent in a team of robots acting concurrently. It is important to disambiguate these two requirements for adaptive behavior, and consider each independently. The first statement of adaptation is imposed by the environment itself. In well-defined, static and stable environments, fixed and reactive robot behavior is generally sufficient. However, when the environment is ever changing, highly dynamic, or is responsive to the robot's own policies, adaptive behaviors provide a huge selective advantage.

An example would be a mobile robot operating in a factory, that is tasked with moving a package from Point *A* to Point *B* every hour. The robot takes route *Z* which is the most efficient way to travel from *A* to *B*. However, sometimes this path is obstructed due to construction or unforeseen disturbances. An adaptive robot here would take path *Z* and observe the obstruction a couple of times, but then would alter its policy to take another path *Y* for the next couple of deliveries as it adapts to the obstruction.

Extending these adaptive tasks to a multi-robot system adds another layer of complexity imposed by the concurrent actions of multiple robots operating within the same environment. The individual robots could be learning or executing static policies, and can differ in capability or objective. The robots may even differ by versions, where a new iteration of robots are added to the team for the same task.

For example, a new and more capable generation of package delivery robots may be introduced to the factory in our previous example. Instead of entirely decommissioning or reprogramming the previous team, it is cheaper and much more effective to build the new team of robots to work alongside the team that is already operational in the factory. Adapting to the previous team would allow the new team of robots to augment its capability, and leverage the resources already present in the factory towards a more effective strategy for package delivery. As robots are increasingly deployed in the real world, such adaptive behaviors will become increasingly important. This is particularly true in commercial applications like cleaning robots, mowing robots or factory robots where heterogeneity within the same task can be frequently borne out of iterations through the years, and adaptive behavior will be crucial for effective reconciliation.

Features relevant to adaptive decision making are often distributed over a robot's current as well as its past states. Considering the entire set of past observations for decision making is intractable as the length of the temporal sequence increases. Truncating the temporal window of observations provides a method to achieve computational tractability while trading off on the quality of information the state representation can capture. However, this introduces a free parameter of temporal window depth that the designer has to set manually, and requires *a priori* knowledge of the the domain. One approach to alleviate this problem is the integration of memory that can be used to selectively remember past observations and use it towards future decision making [1].

Remembering past events can allow a robot to more effectively adjust its behavior and adapt to local changes in the environment [2][3]. Memory can also allow robots to better model other robots' capabilities, limitations and policies, empowering them to form better joint strategies and more complex coordination protocols. Integration of memory is a key cognitive component in most social living organisms that exhibit adaptive behaviors. Adaptive behaviors for single robot systems have been explored in the past largely with the tools of memory [4][5][1]. However, the addition of further complexity in the form of other robots acting concurrently is not widely explored for this class of adaptive tasks.

In this paper, we leverage the developments towards adaptive behavior facilitated through the integration of memory,

978-1-5090-6309-3/17/$31.00 ©2017 IEEE

and extend it to tasks that require, or benefit from, a team of robots learning to adapt and coordinate with each other. The T-Maze task [4][6] is a popular benchmark task often used to test for adaptive behavior in single robot applications. Here, the reward is present in one of the T-Maze endpoints, and alters periodically. The robot traverses the maze for multiple trials and has to find the reward. When the reward location is altered, the robot should alter its strategy accordingly and explore other endpoints and remember the new position in the future. In this work, we extend this task of adaptability to a multi-robot definition where observing the reward altering through the maze's endpoints either requires multiple robots to be at the same endpoint (tightly coupled), or benefits from having multiple robots explore different endpoints concurrently (loosely coupled).

We then develop a neuroevolutionary approach to evolve Gated Recurrent Unit with Memory Block (GRU-MB) [7] as robot policies. GRU-MB is a type of memory augmented neural network which features a modular architecture where the memory is decoupled from the central feedforward operation. This decoupling facilitates independent read and write access to memory allowing for selective memorization of past observations that can be retained across variable temporal scales. This flexible framework representing individual robot policies, combined with the global optimization potency of our neuroevolutionary approach leads to effective joint robot policies that can adapt effectively. Results in a variety of experiments spanning across multiple axes of difficulty in the Extended T-Maze domain demonstrates the efficacy of our approach.

II. Background and Related Work

A body of work has leveraged memory to train adaptive behaviors in single agent systems. Grabowski et al.[8] looked at the evolution of memory usage in environments where information about past experience is required for optimal decision making. Bakker used reinforcement learning in conjunction with a Long Short Term Memory (LSTM) to solve a single T-Maze task [4]. Bayer et al. evolved varying LSTM cell structures [5] while Greve et al. recently evolved Neural Turing Machines [6] to solve a more complicated version of the T-Maze. However, all of these work deal with single agent systems that learn adaptive behavior as exacted by the environment. The addition of further complexity in the form of other agents acting concurrently is not widely explored for this class of adaptive tasks.

Extending single agent approaches to multiagent settings presents challenges in ensuring each agent learns to act in a way that benefits the entire team. A viable decomposition of the team goal is rarely available or easily computable, which necessitates for agents that learn to adapt to each other. Stone et al. [9] highlights this need for collaboration of autonomous agents without preordained protocols for coordination. Distributed policy learning is well suited to these kinds of problems and has received much attention in the field of multiagent coordination. Colby et al. [10] utilized a cooperative coevolutionary algorithm to train multi-robot teams exploring an unknown space environment with humans in the loop. Knudson and Tumer [11] coevolved multi-robot teams to solve a tightly coupled task requiring multiple robot observations and extended it to heterogeneous multi-robot teams with varying capabilities. Hsieh et al. [12] developed a framework for deployment of an adaptive heterogeneous team consisting of aerial and ground robots, for an urban surveillance task. The extended T-Maze domain introduced in this work serves to marry the difficulties of a multi-robot system with the complication of adaptive behavior prescribed by the task itself.

A. Gated Recurrent Unit with Memory Block

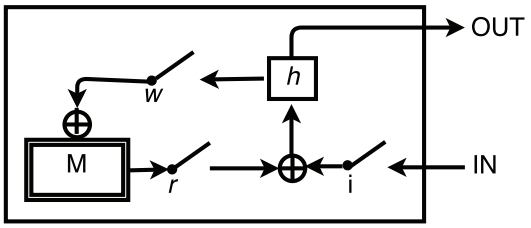


Fig. 1. Schematic of a GRU-MB neural architecture (extracted from [7])

The Gated Recurrent Unit with Memory Block (GRU-MB) [7] is a class of recurrent neural network (RNN) that uses an external block of memory with which it interacts using independent read and write gates. It was developed as a bridge between Gated Recurrent Units (GRUs) [13] and Neural Turing Machines (NTMs) [14] synthesizing the simple structure of the former with the detached memory architecture of the latter. Unlike most other gated RNN architectures which inherently entangle memory with their central feedforward computation, the GRU-MB has the ability to entirely decouple these two structures. Figure 1 illustrates this modular structure where opening the read and write gates shown as w and r, leads to complete detachment of the memory (M) with the feedforward operation of the network. This unique property of the GRU-MB architecture allows it to choose when to read from memory, update it, or simply ignore it. This provides a regimented method to shield the memory from distractions and noise; while interacting with it when required to read, store and process useful information over an extended time scale of activations. GRU-MB has been shown to outperform traditional memory-based methods using Long Short Term Memory (LSTM) [15] in a series of deep memory tasks [7].

B. Cooperative Coevolutionary Algorithm

Cooperative Coevolutionary Algorithms (CCEAs) are an extension of Evolutionary Algorithms (EA) [16][17] that deal with multiple evolving sub-populations, and have been shown to perform well in cooperative multi-robot domains [18]. Multiple populations evolve in parallel, with each population developing a policy for one of the robots in the team. Policies from each population are drawn to form a team of robots, and the overall performance of this team is evaluated using a fitness function $F(z)$, where z is the joint

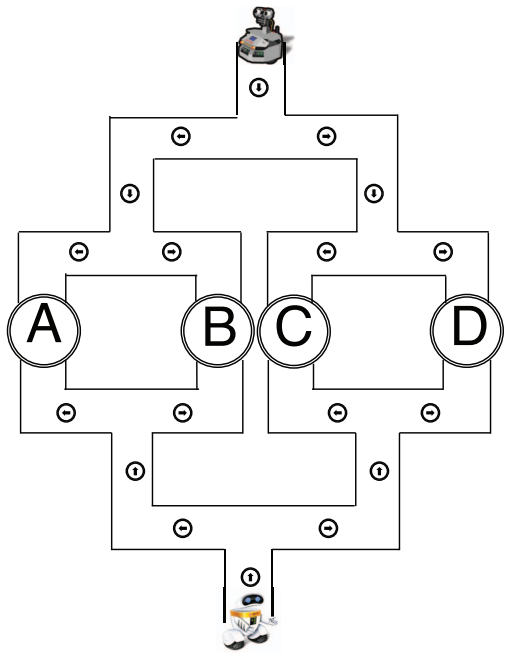


Fig. 2. Illustration of a 2-deep Extended T-Maze with 2 robots. Each robot travels through the corridor and goes left or right at each junction, ending up in one of the endpoints A, B, C or D. The number of robots can be scaled arbitrarily by adding orthogonal planes sharing the endpoints.

state of all robots in the team. This fitness is then assigned to each policy in the team. The key difference between a CCEA and an EA is the assignment of fitness to each evolving robot's policy. In a traditional EA, the fitness of a policy is simply the system performance attained by that policy. However, in a CCEA, all the robots in the team affect the overall system performance; this means that the fitness of an robot's policy is based on its interactions with all other robots it collaborates with, resulting in context-dependent and subjective fitness assignment [19].

III. Extended T-Maze Domain

The T-Maze is a popular benchmark domain used to test for memory based adaptive behaviors in single robot applications [4][6]. We extend the T-Maze domain to a multi-robot paradigm where multiple robots traverse the maze in search for the reward. The location of the reward is changed periodically, and multiple robots are required to coordinate tightly in exploring and observing the reward. The corridors of the maze are identical and indistinguishable which prevents the robots from learning explicit value functions for the states. The robots are instead forced to coordinate tightly with their teammates and learn joint strategies that adapt to the changing reward location.

Figure 2 illustrates a 2-deep extended T-Maze domain with 2 robots. The number of junctions that a robot has to navigate past to reach the endpoint is termed the depth of the T-Maze. The 2-deep T-Maze with two binary decision points (junctions) consists of four separate endpoints marked A, B, C and D. Each robot starts off as shown, and navigates to one of the endpoints. This process is termed a **trial**. During each trial, the reward is present at one of the endpoints, and this location changes 2 times over the course of one **evaluation**. One full evaluation consists of n trials where n is the number of unique endpoints multiplied by 3. For example, a 2-deep T-Maze has 4 unique endpoints leading to 12 trials per evaluation.

The robot records three values as its input. The first input measures the distance to the next junction. The second and third inputs are binary measurements that specify whether other robots were present and whether reward was observed, respectively, in the robot's last trial. The goal of the robot is to maximize the number of trials in which it observes the reward in coordination with the other robots. In a single robot T-Maze case, a policy would be to explore the endpoints in some defined sequence until the reward is observed. The robot then revisits the endpoint where the reward was observed until the location of the reward changes. At this point, the robot then resumes exploration of the endpoints to find the new location. This policy is optimal in the single robot case, and will be termed the **static policy**.

The extended T-Maze domain marries two distinct requirements for adaptive behaviors in one general framework. The first requirement for adaptation is imposed by the task itself where the robot needs to alter its strategy in response to the change of reward location. We will refer to this as **task adaptation**. The difficulty level of task adaptation can be altered by modulating the depth of the task. The second requirement of adaptive behavior is imposed by concurrent actions of multiple robots acting within the same environment and will be referred to as **multi-robot adaptation**. Figure 2 illustrates two robots acting together, but the number of robots can be scaled arbitrarily. The nature of multi-robot adaptation can be specified by the *objective function* that maps the joint action of the multi-robot team to a system performance value. In this work, we specify two such distinct objective function definitions:

a) ***Converge****:* This objective function definition requires the reward and all the robots to be at the same endpoint for the reward to be considered observed in that trial. This is a tightly coupled objective definition which represents tasks like picking up heavy objects that might not be solvable by one robot, and explicitly require the co-presence of multiple robots.

b) ***Spread****:* This is a loosely coupled objective function definition which requires the reward and a minimum of one robot to be at the same endpoint for the reward to be considered observed in that trial. This objective function represents tasks like exploring or information collection which can be completed by one robot, but is able to benefit from larger team sizes.

IV. Multi-Robot Policy Training with Memory

We use a GRU-MB network as our robot control policy leveraging its unique architecture that decouples memory from its feedforward computation. This detachment is crucial as our robots must be able to effectively filter noise and capture the relevant information over the course of a traversal

of the Extended T-Maze. For example, a successful robot control policy should be able to ignore the noisy inputs as it travels the corridors of the T-Maze. Simultaneously however, in order to effectively localize itself, the robot should be able to remember the number of junctions it has passed, as each corridor of the Extended T-Maze looks virtually identical. Additionally, the robot also has to memorize relevant information about its last exploration endpoint so that it can explore a new one for its next trial.

We use CCEA modified for neuroevolution [20][21] to train our neural network control policies. However, memory-augmented architectures like the GRU-MB, can be a challenge to evolve, particularly due to the large number of weights that parameterize highly non-linear and correlated operations. To alleviate this problem, we use independent mutational operators for each weight structure within our neural architectures, and use probabilistic extinction to retain diversity in the population.

Algorithm 1 CCEA used to evolve our multi-robot teams

```
Initialize M populations of k neural network policies
Define a random number generator r() ∈ [0, 1)
for each Generation do
    for i = 1 : k × φ do
        Select a network from each population randomly
        Add networks to team T_i
        Run z independent simulations for team T_i
        Assign reward to each members of T_i
    for each Population do
        Rank networks based on fitness scores
        Select the first e networks as elites
        Probabilistically select (k−e) networks from the entire pool based on fitness, to form Set S
        for iteration (k − e) do
            Randomly extract network N_i from Set S
            for each weight matrix W ∈ N_i do
                if r() < mut_prob then
                    Randomly sample individual weights in W and mutate them by adding 10% Gaussian noise
        if r() < extinct_prob then
            for each Network N_i not in elites do
                if r() < extinct_mag then
                    Reinitialize N_i's weights
```

Algorithm 1 details the CCEA used to evolve our multi-robot teams. The population k was set to 100, and the number of elites to 4% of k. M represents the size of the team and varies between $2-4$ for varying experiments. The number of randomly initialized simulations conducted to compute an evaluation, z was set to 7. The number of different teams an individual robot was drafted to, in order to compute its fitness, ϕ was set to 5. The mutation probability mut_{prob}, extinction probability $extinct_{prob}$ and extinction magnitude $extinct_{mag}$ were set to 0.9, 0.004, and 0.5 respectively.

V. Experimentation

We design a variety of experimental setups to evaluate out neural network controllers in the Extended T-Maze domain. The axes of variations that characterize our experiments are summarized below:

a) Control Policy Architecture: We test two distinct types of neural architectures as control policies that learn. The first kind is a **GRU-MB network** that has memory, while the second type is a standard **feedforward neural network (FF)** without any memory. The same training approach described in Section IV was used to evolve the FF control policies. This axis of variation evaluates the role of memory.

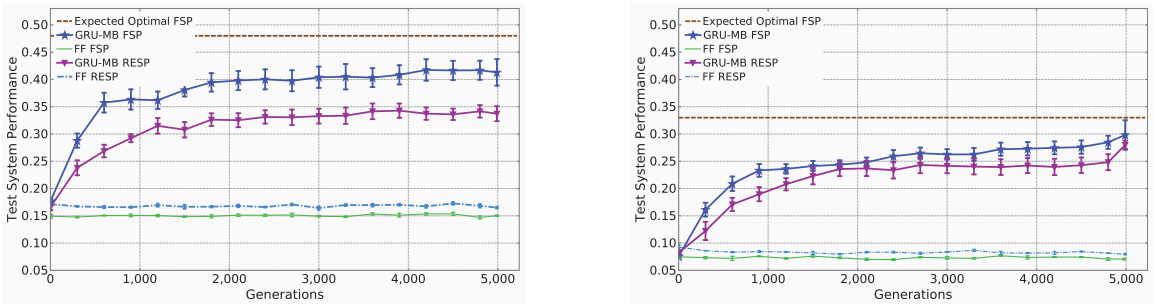
b) Robot Type: We have two categories of robots working together in a team. The first type of robots are **learning robots** that use either a GRU-MB or feedforward neural network as their policy, and are trained. The secondary category of robots are **static robots** that use the static policy as described in Section III. The robot under the static policy explores the endpoints in a specific order, returns to an endpoint if it finds a reward there, and resumes exploring as the reward location is switched. This strategy is the optimal strategy for the T-Maze in a single robot case, and represents a hard-coded optimal policy that is already operational on site in a real world setting. The learning robot's goal is to augment the static robot by adapting to work alongside it.

c) Static Robot Type: We test two types of static robots based on their exploration policy. The first type termed **fixed static policy (FSP)** will be a static robot that has a fixed endpoint it starts with and a fixed routine of exploration. The second type termed the **randomly exploring static policy (RESP)** will vary the endpoint it starts with, and the pattern in which it explores. The RESP represents an added challenge to the learning robots as they have to additionally infer the type of static robot they are teamed with, by observing them during the first couple of trials.

The protocol for computing the results were kept consistent across all the experimental variations. During each generation of training, the individual robot with the highest fitness from its population was selected to be part of a champion team. The champion team was then tested on a separate test set of 15 experiments, generated randomly for each generation. The average reward achieved by the champion team normalized by the number of trials, on this test set was then logged and reported as the **Test System Performance** for that generation. This protocol of a test set was implemented to shield the reported metrics from any bias of the distribution of random reward locations, depth of the task, or the size of the population. Ten independent statistical runs were conducted, and the average with error bars reporting the standard error in the mean were logged.

VI. Experiment 1: Converge

This set of experiments test the multi-robot teams consisting of one static robot and one learning robot, under the converge objective function as explained in Section III. This is a tightly coupled task definition where all the robots are required to be alongside the reward to observe it.



(a) (b)

Fig. 3. GRU-MB and FF control policies are compared for a 2-deep (a) and 3-deep (b) tightly coupled Extended T-Maze, where both the static and learning robot have to be at the same endpoint alongside the reward to observe it.

Figure 3(a) shows the results for a 2-deep Extended T-Maze domain tested against robots with both fixed (FSP) and randomly exploring static policies (RESP). The brown dashed line represents the expected system performance for an optimal joint policy between the learning robot and the FSP robot. The expected value for optimal joint policy alongside the RESP is not easily computable, but is provably lower than the FSP case. This is apparent as FSP is simply a special case of RESP where the type of robot exploration routine is fixed. The robot using the GRU-MB network significantly outperforms the robot using the feedforward network as its policy across both types of static robot types. The GRU-MB control policy achieves a system performance of $41.3 \pm 2.4\%$ and $33.7 \pm 1.4\%$ while the FF control policy achieves a system performance of $15.0 \pm 0.1\%$ and $16.5 \pm 0.2\%$ for the FSP and RESP variations respectively.

Figure 3(b) shows the results for a 3-deep Extended T-Maze domain tested against robots with both fixed (FSP) and randomly exploring static policies (RESP). The brown dotted line represents the expected system performance for an optimal joint policy between the learning robot and the FSP. The robot using the GRU-MB network significantly outperforms the robot using the feedforward network as its control policy across both types of static robot types. The GRU-MB control policy achieves a system performance of $29.8 \pm 2.7\%$ and $28.0 \pm 1.5\%$ while the FF control policy achieves a system performance of $7.1 \pm 0.2\%$ and $8.0 \pm 0.1\%$ for the FSP and RESP variations respectively. The GRU-MB robots' performance alongside the FSP static robot variation has also not fully converged and seems to be approaching the expected optimal joint policy value of 33%.

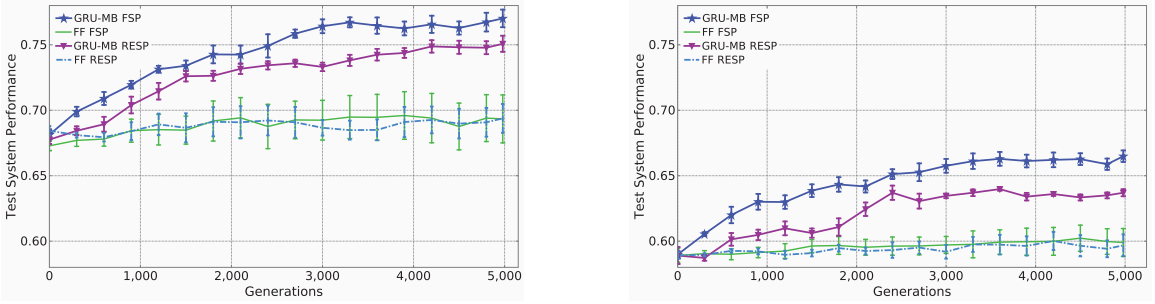
The results in Figure 3 demonstrate that the robot using the GRU-MB network is able to learn effective policies that let it adapt to the varying types of static robots. The robot using the FF network however fails to learn any intelligent joint policy at all. The critical factor behind this disparity is memory, which is the primary point of variation between the GRU-MB and FF network architecture. The FF robots do not have enough input information to form an effective strategy that can learn to exploit the reward when its location is known, and explore when it is unknown. Additionally, the FF robots also do not have enough information to model their teammates' policies and built a strategy around it. An immediate mapping between their current input measurements and an effective exploration strategy or joint action policy is not existent. The GRU-MB robots on the other hand have access to the same information (input) as the FF robots, but have the additional capability of memory. This ability to memorize past events and process temporal dependencies in their input channels forms a type of information bridge that can be leveraged to form effective exploration strategies that can adapt to the change in reward location. Additionally, the availability of memory also allows the GRU-MB robot to model its teammates' policy and adapt alongside it to maximize system performance. This is especially apparent in the RESP case where in order to succeed, the GRU-MB robot is required to first samples its colleague static robots' actions, identify its exploration strategy and adapts its behavior to best supplement that strategy. This behavior highlights the crucial role memory plays in learning adaptive behaviors that facilitate tight multi-robot coordination.

VII. Experiment 2: Spread

This set of experiments test the multi-robot teams under the spread objective function as explained in Section III. This is a loosely coupled task definition where a minimum of one robot is required to be alongside the reward to observe it.

Figure 4(a) shows the results for a 2-deep Extended T-Maze tested against both FSP and RESP robots. The expectation for the lower bound of performance (not plotted) is 48% and represents the worst possible system performance achievable by the learning robot. This is the expected system performance achieved simply by the hard coded policy of the static robot. The robot using GRU-MB network outperforms the robot using the feedforward network as its policy across both types of static robot types. The GRU-MB control policy achieves a system performance of $77.0 \pm 0.7\%$ and $75.1 \pm 0.6\%$ while the FF control policy achieves a system performance of $69.3 \pm 2.0\%$ and $69.4 \pm 1.2\%$ for the FSP and RESP variations respectively.

Figure 4(b) shows the results for a 3-deep Extended T-Maze domain tested against both FSP and RESP robots. The expectation for the lower bound of performance (not plotted) is 33%. The robot using the GRU-MB network outperforms



(a) (b)

Fig. 4. GRU-MB and FF control policies are compared for a 2-deep (a) and 3-deep (b) Extended T-Maze domain, using the Spread objective where any one robot is sufficient in observing the reward.

the robot using the feedforward network as its policy across both types of static robot policies. The GRU-MB control policy achieves a system performance of $67.5 \pm 0.4\%$ and $63.7 \pm 0.3\%$ while the FF control policy achieves a system performance of $60.0 \pm 1.0\%$ and $60.0 \pm 0.8\%$ for the FSP and RESP variations respectively.

Similar to the results in Section VI, the results in Figure 4 demonstrate the GRU-MB robot learning to form effective joint policies alongside both types of static robots. The FF robots however fail to learn and simply picks an endpoint to stick to. It fails to adapt to the task where the reward location changes periodically, and to the nature of the static robot's exploration strategy. The GRU-MB robot however is able to leverage its memory and adapt to both the task, and the varying exploration strategies of its teammates, effectively augmenting the static robots in the task.

A. Multiple Learning Robots

To further probe our approach on the axis of **multi-robot adaptation**, we ablate the robot with static policy from our multi-robot team, and experiment with learning robots of varying team sizes.

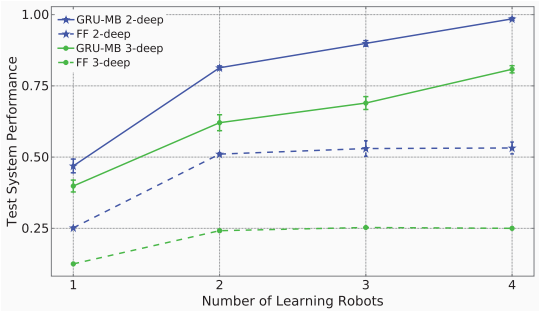


Fig. 5. GRU-MB and FF control policies are compared for a 2-deep and 3-deep Extended T-Maze, with Spread objective tested for varying number of learning robots.

Figure 5 shows the results for a 2-deep and 3-deep Extended T-Maze tested for varying number of learning robots. The robots using GRU-MB network significantly outperform the robots using the feedforward network as its control policy across all values for team size and task depth. The system performance for the GRU-MB robots improves with increasing number of robots for both variations of depth. This demonstrates that the GRU-MB team is able to leverage memory and form coordinated exploration strategies that explore different sections of the Extended T-Maze, benefiting from larger team sizes. The FF team however fails to form effective joint exploration strategies and thus fails to fully benefit from increasing team sizes. This result highlights the critical role memory plays in facilitating the learning of effective coordination policies in adaptive tasks like the Extended T-Maze.

VIII. Conclusion

In this work, we designed a diverse set of experiments across multiple axes of difficulty in the flexible Extended T-Maze domain. Our results demonstrated that the multi-robot teams trained with the GRU-MB network perform significantly better in all tasks as compared with multi-robot teams trained with a traditional feedforward neural network, acting with the same set of information. These results highlight the critical role memory plays in developing effective multi-robot coordination strategies that can adapt to changing environmental factors and teammates' policies.

The Extended T-Maze domain formulated in the paper is designed to be a general framework for testing adaptive tasks alongside the difficulties of a multi-robot approach. The framework is also devised to be modular and flexible such that each axis of difficulty can be regulated somewhat independently of the other. Future work will look to retain these properties while transcribing the domain into a ROS environment. This is prescribed with the objective of integrating the noise and constraints of a robot hardware within the domain, and facilitating benchmarks for performance with real robots. We will start with the GRU-MB based controllers developed in this paper.

Additionally, in this work we used the system performance obtained for each multi-robot team as an evaluation for each of its member. Future work will look to integrate reward shaping techniques like the difference reward [22] that can better compute each robot's marginal utility and provide a cleaner signal to learn on.

REFERENCES

[1] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Evolving adaptive neural networks with and without adaptive synapses," in *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 4. IEEE, 2003, pp. 2557–2564.

[2] R. Dukas, "Evolutionary biology of insect learning," *Annu. Rev. Entomol.*, vol. 53, pp. 145–160, 2008.

[3] T. S. Collett, P. Graham, and V. Durier, "Route learning by insects," *Current opinion in neurobiology*, vol. 13, no. 6, pp. 718–725, 2003.

[4] B. Bakker, "Reinforcement learning memory," *Neural Information Processing Systems 14*, pp. 1475–1782, 2002.

[5] J. Bayer, D. Wierstra, J. Togelius, and J. Schmidhuber, "Evolving memory cell structures for sequence learning," *Artificial Neural Networks–ICANN 2009*, pp. 755–764, 2009.

[6] R. B. Greve, E. J. Jacobsen, and S. Risi, "Evolving neural turing machines," in *Neural Information Processing Systems: Reasoning, Attention, Memory Workshop*, 2015.

[7] S. Khadka, J. J. Chung, and K. Tumer, "Evolving memory-augmented neural architecture for deep memory problems," in *In Proceedings of the Genetic and Evolutionary Computation Conference 2017*. ACM, 2017.

[8] L. Grabowski, D. Bryson, F. Dyer, C. Ofria, and R. Pennock, "Early evolution of memory usage in digital organisms," *Procedings of the International Conference on Artificial Life*, 2010.

[9] P. Stone, G. A. Kaminka, S. Kraus, and J. S. Rosenschein, "Ad hoc autonomous agent teams: Collaboration without pre-coordination," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[10] M. Colby, L. Yliniemi, and K. Tumer, "Autonomous multiagent space exploration with high-level human feedback," *Journal of Aerospace Information Systems*, pp. 1–15, 2016.

[11] M. Knudson and K. Tumer, "Coevolution of heterogeneous multi-robot teams," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 2010, pp. 127–134.

[12] M. A. Hsieh, A. Cowley, J. F. Keller, L. Chaimowicz, B. Grocholsky, V. Kumar, C. J. Taylor, Y. Endo, R. C. Arkin, B. Jung, *et al.*, "Adaptive teams of autonomous aerial and ground robots for situational awareness," *Journal of Field Robotics*, vol. 24, no. 11-12, pp. 991–1014, 2007.

[13] K. Cho, B. v. M. C. Gulcehre, D. Bahdanau, F. B. H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1724–1734.

[14] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing machines," 2014, preprint at https://arxiv.org/abs/1410.5401.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] W. Spears, K. De Jong, T. Bäck, D. Fogel, and H. De Garis, "An overview of evolutionary computation," in *Machine Learning: ECML-93*. Springer, 1993, pp. 442–459.

[17] D. B. Fogel, *Evolutionary computation: toward a new philosophy of machine intelligence*. John Wiley & Sons, 2006, vol. 1.

[18] S. G. Ficici, O. Melnik, and J. B. Pollack, "A game-theoretic and dynamical-systems analysis of selection methods in coevolution," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 580–602, 2005.

[19] M. Colby, J. J. Chung, and K. Tumer, "Implicit adaptive multi-robot coordination in dynamic environments," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 5168–5173.

[20] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Efficient non-linear control through neuroevolution," in *European Conference on Machine Learning*. Springer, 2006, pp. 654–662.

[21] D. Floreano, P. Dürr, and C. Mattiussi, "Neuroevolution: from architectures to learning," *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008.

[22] A. Agogino and K. Tumer, "Efficient evaluation functions for evolving coordination," *Evolutionary Computation*, vol. 16, no. 2, pp. 257–288, 2008.