Towards Learning Scalable Agile Dynamic Motion Planning for Robosoccer Teams with Policy Optimization

Brandon Ho*, Batuhan Altundas, Matthew Gombolay

***Abstract*—In fast-paced, ever-changing environments, dynamic Motion Planning for Multi-Agent Systems in the presence of obstacles is a universal and unsolved problem. Be it from path planning around obstacles to the movement of robotic arms, or in planning navigation of robot teams in settings such as Robosoccer, dynamic motion planning is needed to avoid collisions while reaching the targeted destination when multiple agents occupy the same area. In continuous domains where the world changes quickly, existing classical Motion Planning algorithms such as RRT* and A* become computationally expensive to rerun at every time step. Many variations of classical and well-formulated non-learning path-planning methods have been proposed to solve this universal problem but fall short due to their limitations of speed, smoothness, optimally, etc. Deep Learning models overcome their challenges due to their ability to adapt to varying environments based on past experience. However, current learning motion planning models use discretized environments, do not account for heterogeneous agents or replanning, and build up to improve the classical motion planners' efficiency, leading to issues with scalability. To prevent collisions between heterogenous team members and collision to obstacles while trying to reach the target location, we present a learning-based dynamic navigation model and show our model working on a simple environment in the concept of a simple Robosoccer Game.**

***Index Terms*—Motion Planning, Deep Learning, Path Planning**

I. Introduction

A fast dynamic Motion Planning for Multi-Agent Systems in the presence of obstacles is a universal and unsolved problem [1], [2] applicable to robotics, planning, and optimization. In many industrial applications, users are interested in trajectories with minimum time to achieve the highest productivity [3]. Especially in athletics, Agile Motion Planning becomes an important factor for fast reactions for both gameplay quality and safety [4], especially when collaborating with humans. Our work can be used to provide an agile motion planning policy for Robosoccer, an environment that aims to promote AI and robotic research, specifically in a soccer domain [5].

In continuous domains where the world changes quickly, existing classical Motion Planning algorithms such as RRT* [6] and A* [7] become computationally expensive to rerun at every time step. The classical algorithms cannot capture the real-world dynamics to produce a non-collision path at once for all future time steps. Many variations of classical and well-formulated non-learning path-planning methods have been proposed to solve this universal problem [6] [8] [9]. Those methods are generalized in the following categories: sampling-based, graph-based, geometric-based, and optimization-based. However, these methods have their disadvantages. Sampling-based methods lack in producing consistent trajectories, and the produced trajectories are not smooth [1], [2].

Deep learning models' ability to adapt based on past experiences to varying environments is more appealing to use than classical models with fixed parameters [10]. Currently, the application of deep learning in motion planning is unmatured and will continue to grow, especially in Multi-Agent Reinforcement Learning (MARL) [2], [10], [11]. Many current learning motion planners have similar characteristics: discretized environment, single- or multi-agent, non-generative, and no replanning. A discretized environment limits the learning algorithm's efficiency and cost-effectiveness based on the granularity of the discretization. Many learning models do not account for heterogeneous agents. Learning-based approaches such as Neural-RRT [12] or Neural-A* [13] utilize a Deep Learning method to provide candidate locations or probability distributions for classical algorithms to leverage instead of generating a path themselves. Thus, they still run the baseline algorithms to generate the paths, making them inefficient. These methods, while shown to be faster than simple RRT* and A*, respectively, are still slow compared to pure learning-based approaches.

Using a Learning-based Motion Planner with symbolic knowledge of the agents provides a new path-finding method that can be trained to maximize the number of targets reached while avoiding collisions. In addition, our method minimizes the computational cost of recalculating a new non-colliding path in a dynamic environment.

We present the following in this paper:

- Present a Dynamic Motion Planning Environment for Heterogenous Teams that can be used for Robots in Sports.
- Provide an End-to-End Trainable Method that can be run in a Decentralized Setting for Agile-Motion Planning.
- Discuss future work to make the models scalable through the use of Graph Neural Networks to allow for the use of the model in different team compositions.

This work was supported by the Office of Naval Research under grant N00014-22-1-2834

II. Background

	Discrete/Continuous Space	Multi-Agents	Scalability	Generative Motion Planner	Replanning	Dynamic Obstacles	Heterogeneous Agents
Keselman et al. 2018 [14]	**D**						
Wang et al. 2020 [12]	**D**						
Yonetani et al. 2021 [13]	**D**						
Lv et al. 2019 [15]	**D**			✓			
Ichter et al. 2018 [16]	**C**						
Khan et al. 2020 [17]	**C**		✓				
Yu, Gao, et al, 2021 [18]	**C**		✓	✓			
Li et al. 2020 [19]	**D**	✓		✓	✓		
Liu et al. 2020 [20]	**D**	✓			✓	✓	
Zhang et al. 2022 [21]	**C**		✓	✓		✓	
Ours	**C**	✓		✓	✓	✓	✓

TABLE I: Related Works in Motion Planning and their applications of Domain, Scalability, Methods and Team Compositions.

Table I represents recent work on motion planning. The existing work indicate that there is a need for heterogenous multi-agent motion planner for dynamic environments in both discrete and continuous space. While this paper focuses on the learning of a dynamic motion planner, we plan to address scalability, which accounts for either multiple obstacles or multiple agents, in our future work.

A. Motion Planning

The simplest motion planning goes by in a straight line from a start location to the end location with no consideration of obstacles. Non-learning motion planning algorithms, including RRT, RRT*, and RRG, account for obstacle avoidance but have poor sample efficiency. Learning motion planning algorithms, such as a learning RRT, can improve the sample efficiency of its non-learning counterpart. However, learning motion planner algorithms still require non-learning motion planners to generate their path, so they have the same time complexity as their non-learning counterparts.

III. Problem Statement and Setup

A. Environment

We present a simple motion planning environment that allows for the representation of team members, opponents, and goal locations. The environment can be translated into the context of Robosoccer Environment [5] as shown in Fig. 1. The agent is tasked with moving to the target location with the ball, intercepting it, and moving it to another series of locations while avoiding getting close to the opponent players who aim to intercept the player or avoid colliding with the other teammates. Both other teammates and opponent players are represented as static or moving obstacles, making the model decentralized. The training was done on a single agent, and therefore each agent can also be trained decentralized.

- **Agents:** Heterogenous Agents with different speeds and different collision radius that are assigned a different sequence of target locations

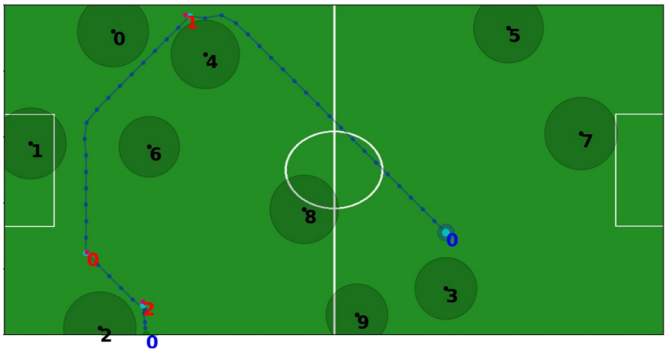


Fig. 1: Sample policy of the Trained Model blue agent avoiding static black Obstacles while moving from red Targets 1 to 0 to 2 in Robosoccer Domain with 0 collisions.

- **Targets:** Target locations with a circular keep-out radius that the agents are assigned to. The model takes in the sequence of target locations for each agent. This approach allows for the integration of a higher-level centralized controller that handles the Task Allocation and Scheduling [22] while allowing the Motion Planner to be run decentralized.
- **Obstacles:** Obstacles or opponents that the agents need to avoid with a circular keep-out radius. These opponents can be moving or static. At each time step, the model takes in their current location and generates a new action for the agents.

B. Problem Formulation

We formulate the motion planning problem as a fully observable Markov Decision Process (MDP) using the five-tuple $\langle S, A, T, R, \gamma \rangle$ shown below:

- States: The problem state S in Motion Planning Problems is a joint state consisting of all the agent, target, and obstacle information.
- Actions: Actions at time-step t within the Motion Planning Domain refers to a complete set of single-step motion plans for each agent, denoted as $A_t = [\langle \Delta x_{i_t}, \Delta y_{i_t} \rangle, \langle \Delta x_{i+1_t}, \Delta y_{i+1_t} \rangle, ...]$, to be executed at the same time in time-step t for all of agents i.
- Transitions: T corresponds to executing the action in Motion Planner and proceeds to the next time step.
- Rewards: R_t is based on the scheduling objective the user wants to optimize and the number of collisions that the agents have collided. In Section III-D we show how to compute R_t when optimizing motion plan to avoid collision with obstacles.
- Discount factor: γ.

C. Model

We utilize a simple Neural Network Model that takes in the following information to generate an output of $\langle \delta x, \delta y \rangle$.

- **Agent Information:** Current location $\langle x_{u,t}, y_{u,t} \rangle$ at time-step t for agent u.
- **Target Information:** Current location, $\langle x_{v,t}, y_{v,t} \rangle$ at time-step t for target v.

- **Obstacle Information:** A set of the current location, the keep-out radius, and distance from the agent's current location to its current location for each obstacle, $[\langle x_{o_i,t}, y_{o_i,t}, r_{o_i}, d_{o_i,t}\rangle, \langle x_{o_{i+1},t}, y_{o_{i+1},t}, r_{o_{i+1}}, d_{o_i,t}\rangle, ...]$ at time-step t for all N closest obstacles o_i.

a) Baseline: We compare the performance of our results to a Target-to-Target Heuristic that moves to the location of the given target in a straight line without knowledge of the obstacle locations. This baseline is also used in the Dynamic Environment for the Obstacles, as they move to the current observed location of the Agents. It serves as an upper bound for our model in the number of target location it can reach.

D. Policy Optimization

Our reward policy is trained using a Reward based on the distance to target, and the distance to the obstacles [23] and is formulated in Equation 1.

$$R = -\alpha D(u, v) + \sum_{i=1}^{N} g(D(u, o_i), r_{o_i}) \tag{1}$$

$$g(D, r) = \begin{cases} \beta_1(D - r), & \text{if } D - r \geq 0 \\ \beta_2(D - r) & \text{otherwise} \end{cases} \tag{2}$$

where α, β_1, and β_2 are constants set to 10, 1, 100, respectively, and D is the distance function to the obstacle's center and r is the radius of the keep out range from the center where the unit is 'threatened' and collides with the obstacle.

We train our model in using Policy Gradient methods that seek to directly optimize the model parameters based on the present reward received from the environment [24].

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi(\sum_{t}^{T} A_t^{\pi_\theta}(s_t, a_t) \nabla_\theta log \pi_\theta(a_i|s_t)) \tag{3}$$

In Equation 3, the advantage term, A_t, is estimated by subtracting the mean reward from a batch of policies from the reward of each policy. Each decision is sampled from a policy generated from a single observation, and an action is chosen randomly across them to perpetuate the environment in the next time step. We use the gradients calculated from Equation 3 to update the model weights.

IV. Experimental Results

We train our model in a static environment for a single agent, 3 targets, and 10 obstacles with knowledge of $N = 10$ closest obstacles for 4000 training steps. Our observation for the model is limited to observing the closest 10 obstacles. This is the limitation of the model being unable to scale as the number of obstacles increases.

We set γ = 0.99, batch size = 8 and used Adam optimizer [25] with a learning rate of 8×10^{-3}, and a weight decay of 1×10^{-4}. Specifically, we compute the gradient of the model using the log-likelihood at each stage for each agent, as shown in Equation 3: We train our model using static obstacles for 1 agent and 3 targets. All models are trained and evaluated on a Mac Studio 2022 with an Apple M1 Ultra Chip.

We test our model on different seeds (10, 11, 12) and compare against the Baseline on 3 domains for 3 independent sets of 100 test problems as follows:

- **Static Simple Domain:** 1 Agent assigned to 3 Targets in random order with static 10 Obstacles.
- **Multi-Agent Dynamic Domain:** 3 Agents assigned to 3 Targets in random order, with mobile 10 Obstacles moving to one of the 3 Agents at random in a straight line, at the same speed as Agents.
- **Multi-Agent Dynamic Domain (Large):** 3 Agents assigned to 10 Targets in random order, with 10 mobile Obstacles that are moving to one of the 3 Agents at random in a straight line, at the same speed as Agents.

We evaluate the performance of a Motion Planning Policy using 3 metrics:

- Number of Collisions over the entire timeline of each problem instance, with the maximum performance based on the number of time steps.
- Number of Targets Reached over the entire runtime of the problem.
- Weighted Score based on:

$$S = \alpha \times n_\tau - n_c \tag{4}$$

where α is a constant set to 10, n_τ is the number of targets reached, and n_c is the number of collisions over the entire runtime instance.

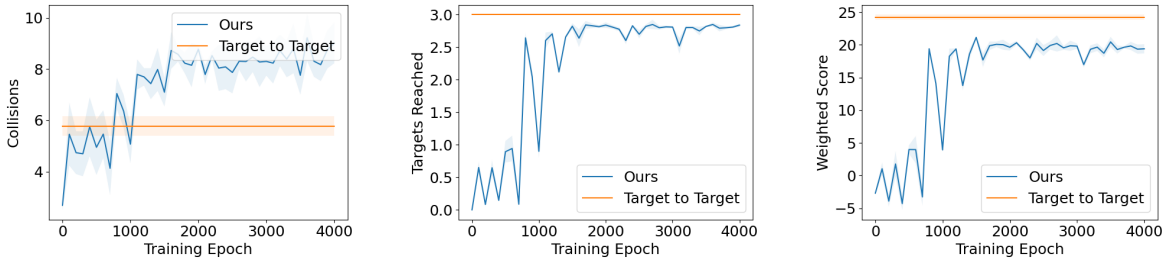
We show the performance of the trained obstacle avoidance policy compared to the baseline solution of going directly to the target, without any knowledge of the obstacles in Figures 2, 3 and 4.

The baseline is always able to reach the provided targets due to taking the shortest path to the location in the continuous domain. The performance of our model in Figs. 2(b), 3(b) and 4(b) show that our model learns to reach the target locations.

In the **Simple Domain** in Figs. 2, the performance of the trained model is less than the performance of the baseline, as there are more collisions and the weighted score shows a similar pattern to the number of targets reached. The initial low number of collisions is due to the untrained policy, where the agent is not moving to the target and therefore not moving within the keep-out area of the obstacles.

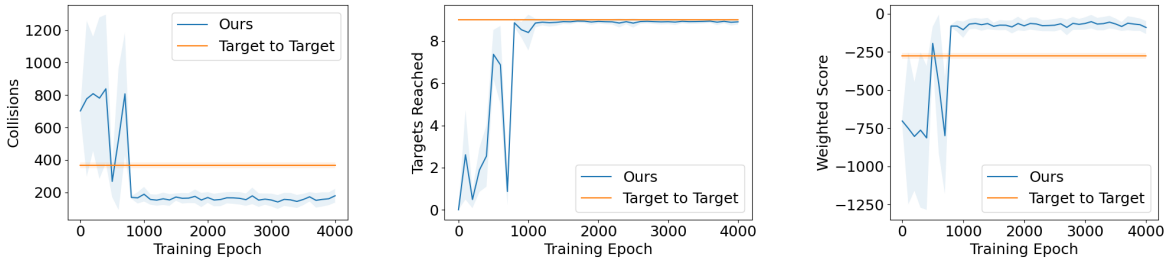
In the **Multi-Agent Dynamic Domain** in Figs. 3, the number of collisions decreases over time as the mobile obstacles move to the location of the agents. This forces the agents to learn to move away. The learned performance shows that training learned on simple path planning in static environments is transferable to dynamic domains.

The main challenges of **Multi-Agent Dynamic Domain (Large)** are the number of Targets being too large for even the fastest policy (baseline) to complete, and the lack of full-observability for our model. The trained model takes in the 10 closest obstacles, treating Targets that it is not currently assigned to as obstacles as well. The limited observability of a larger number of obstacles leads to a decrease in the performance of the trained model as seen in Figs. 4.



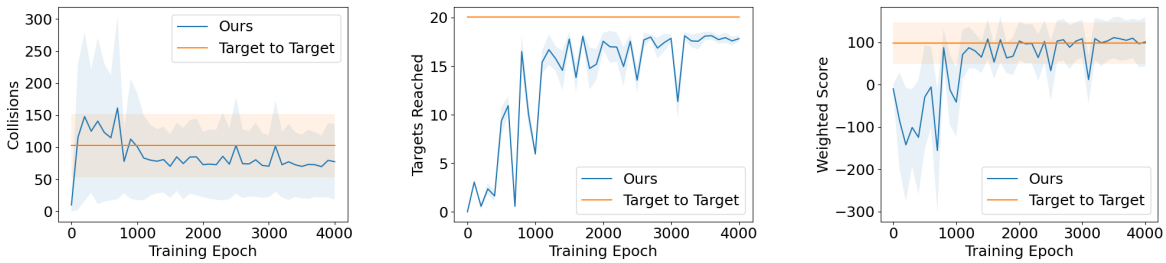
(a) (b) (c)

Fig. 2: Single-Agent Static Domain with 3 Targets, and 10 Obstacles Performance for Collisions (lower is better), Number of Targets reached, and Weighted Score based on Eq. 4 (higher is better).



(a) (b) (c)

Fig. 3: Multi-Agent Dynamic Domain with 3 Agents, 3 Targets, and 10 Obstacles Performance for Collisions (lower is better), Number of Targets reached, and Weighted Score based on Eq. 4 (higher is better).



(a) (b) (c)

Fig. 4: Multi-Agent Dynamic Domain (Large) with 3 Agents, 10 Targets, and 10 Obstacles Performance for Collisions (lower is better), Number of Targets reached, and Weighted Score based on Eq. 4 (higher is better).

V. Conclusion and Future Work

We present a Multi-Agent Agile Motion Planning Model for navigation across an environment with static and moving obstacles. We show that our model is end-to-end trainable in a custom environment that is based on Robosoccer, with multiple agents required to avoid moving obstacles. We further show that our model is capable of handling navigation in continuous space dynamically and avoiding moving targets.

A key limitations of the current model is its scalability. Scalability in multi-agent teams is a challenge that is an open research problem [26]. We plan to integrate the use of Graph Neural Networks for the Motion Planner to address this need for scalability. While our model scales with the number of agents, it is unable to accurately represent the observation space leading to performance drop as seen in Figure 4. The scalability of graph-based models would allow us to address the limitations presented within our results [27], [28].

The environment that we have presented utilizes a fully observable MDP. We show that our model is able to perform under partial observability or stochastic behavior of team members or opponents. We plan to expand on our current work to account for different observability conditions.

With the presence of an adversary, the environment we have presented can be further developed into accounting for Adversarial games, using the Game Theoretic approaches for MARL [29]. We plan to further develop our environment to more accurately represent different adversarial gameplay scenarios.

References

[1] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A Review of Motion Planning Techniques for Automated Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016.

[2] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A Survey of Deep Learning Techniques for Autonomous Driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, Apr. 2020, arXiv:1910.07738 [cs]. [Online]. Available: http://arxiv.org/abs/1910.07738

[3] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[4] Z. Zaidi, D. Martin, N. Belles, V. Zakharov, A. Krishna, K. M. Lee, P. Wagstaff, S. Naik, M. Sklar, S. Choi, Y. Kakehi, R. Patil, D. Mallemadugula, F. Pesce, P. Wilson, W. Hom, M. Diamond, B. Zhao, N. Moorman, R. Paleja, L. Chen, E. Seraj, and M. Gombolay, "Athletic Mobile Manipulator System for Robotic Wheelchair Tennis," Feb. 2023, arXiv:2210.02517 [cs]. [Online]. Available: http://arxiv.org/abs/2210.02517

[5] H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada, "The RoboCup synthetic agent challenge 97," in *RoboCup-97: Robot Soccer World Cup I*, ser. Lecture Notes in Computer Science, H. Kitano, Ed. Berlin, Heidelberg: Springer, 1998, pp. 62–73.

[6] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," May 2011, arXiv:1105.1186 [cs]. [Online]. Available: http://arxiv.org/abs/1105.1186

[7] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path Planning with Modified a Star Algorithm for a Mobile Robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S187770581403149X

[8] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning." [Online]. Available: http://msl.cs.uiuc.edu/ lavalle/papers/Lav98c.pdf

[9] S. Karaman and E. Frazzoli, "Incremental Sampling-based Algorithms for Optimal Motion Planning," 2010. [Online]. Available: https://arxiv.org/pdf/1005.0416.pdf

[10] S. Teng, X. Hu, P. Deng, B. Li, Y. Li, Y. Ai, D. Yang, L. Li, Z. Xuanyuan, F. Zhu, and L. Chen, "Motion Planning for Autonomous Driving: The State of the Art and Future Perspectives," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 6, pp. 3692–3711, Jun. 2023.

[11] Y. Huang and Y. Chen, "Autonomous Driving with Deep Learning: A Survey of State-of-Art Technologies," Jul. 2020, arXiv:2006.06091 [cs]. [Online]. Available: http://arxiv.org/abs/2006.06091

[12] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural RRT*: Learning-Based Optimal Path Planning," 2020. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=9037111

[13] R. Yonetani, T. Taniai, M. Barekatain, M. Nishimura, and A. Kanezaki, "Path Planning using Neural A* Search," Jul. 2021, arXiv:2009.07476 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2009.07476

[14] A. Keselman, S. Ten, A. Ghazali, and M. Jubeh, "Reinforcement Learning with A* and a Deep Heuristic," Nov. 2018, arXiv:1811.07745 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1811.07745

[15] L. Lv, S. Zhang, D. Ding, and Y. Wang, "Path Planning via an Improved DQN-Based Learning Policy," *IEEE Access*, vol. 7, pp. 67 319–67 330, 2019.

[16] B. Ichter, J. Harrison, and M. Pavone, "Learning Sampling Distributions for Robot Motion Planning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 7087–7094, iSSN: 2577-087X.

[17] A. Khan, A. Ribeiro, V. Kumar, and A. G. Francis, "Graph Neural Networks for Motion Planning," Dec. 2020, arXiv:2006.06248 [cs]. [Online]. Available: http://arxiv.org/abs/2006.06248

[18] C. Yu and S. Gao, "Reducing Collision Checking for Sampling-Based Motion Planning Using Graph Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 4274–4289.

[19] Q. Li, F. Gama, A. Ribeiro, and A. Prorok, "Graph Neural Networks for Decentralized Multi-Robot Path Planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 11 785–11 792, iSSN: 2153-0866.

[20] Z. Liu, B. Chen, H. Zhou, G. Koushik, M. Hebert, and D. Zhao, "MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 11 748–11 754, iSSN: 2153-0866.

[21] R. Zhang, C. Yu, J. Chen, C. Fan, and S. Gao, "Learning-based Motion Planning in Dynamic Environments Using GNNs and Temporal Encoding," Oct. 2022, arXiv:2210.08408 [cs]. [Online]. Available: http://arxiv.org/abs/2210.08408

[22] A. Messing, G. Neville, S. Chernova, S. Hutchinson, and H. Ravichandar, "GRSTAPS: Graphically Recursive Simultaneous Task Allocation, Planning, and Scheduling," *The International Journal of Robotics Research*, vol. 41, no. 2, pp. 232–256, Feb. 2022. [Online]. Available: http://journals.sagepub.com/doi/10.1177/02783649211052066

[23] M. Hausknecht and P. Stone, "Deep Reinforcement Learning in Parameterized Action Space," Feb. 2016, arXiv:1511.04143 [cs]. [Online]. Available: http://arxiv.org/abs/1511.04143

[24] J. Peters, "Policy gradient methods," *Scholarpedia*, vol. 5, no. 11, p. 3698, Nov. 2010. [Online]. Available: http://www.scholarpedia.org/article/Policy_gradient_methods

[25] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017, arXiv:1412.6980 [cs]. [Online]. Available: http://arxiv.org/abs/1412.6980

[26] M. Natarajan, E. Seraj, B. Altundas, R. Paleja, S. Ye, L. Chen, R. Jensen, K. C. Chang, and M. Gombolay, "Human-Robot Teaming: Grand Challenges," *Current Robotics Reports*, vol. 4, no. 3, pp. 81–100, Aug. 2023. [Online]. Available: https://link.springer.com/10.1007/s43154-023-00103-1

[27] B. Altundas, Z. Wang, J. Bishop, and M. Gombolay, "Learning Coordination Policies over Heterogeneous Graphs for Human-Robot Teams via Recurrent Neural Schedule Propagation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 11 679–11 686, arXiv:2301.13279 [cs]. [Online]. Available: http://arxiv.org/abs/2301.13279

[28] Z. Wang, C. Liu, and M. Gombolay, "Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints," *Autonomous Robots*, vol. 46, no. 1, pp. 249–268, Jan. 2022. [Online]. Available: https://link.springer.com/10.1007/s10514-021-09997-2

[29] A. Celli, M. Ciccone, R. Bongo, and N. Gatti, "Coordination in Adversarial Sequential Team Games via Multi-Agent Deep Reinforcement Learning," Dec. 2019, arXiv:1912.07712 [cs]. [Online]. Available: http://arxiv.org/abs/1912.07712