

Graph Neural Networks for Decentralized Path Planning

Extended Abstract

Qingbiao Li
Dept. of Computer Science and
Technology, University of Cambridge
United Kingdom
ql295@cam.ac.uk

Fernando Gama,
Alejandro Ribeiro
Dept. of Electrical and Systems
Engineering, University of
Pennsylvania, USA
{fgama,aribeiro}@seas.upenn.edu

Amanda Prorok
Dept. of Computer Science and
Technology, University of Cambridge
United Kingdom
asp45@cam.ac.uk

ABSTRACT

We propose a combined model that automatically synthesizes local communication and decision-making policies for agents navigating in constrained workspaces. Our architecture is composed of a convolutional neural network (CNN) that extracts adequate features from local observations, and a graph neural network (GNN) that communicates these features among agents. We train the model to imitate an expert algorithm, and use the resulting model online in decentralized planning involving only local communication and local observations. We evaluate our method in simulations involving teams of agents in cluttered workspaces. We measure the success rates and sum of costs over the planned paths. The results show a performance close to that of our expert algorithm, demonstrating the validity of our approach. In particular, we show our model’s capability to generalize to previously unseen cases (involving larger environments and larger agent teams).

KEYWORDS

Graph neural networks, decentralized multi-agent path planning, inter-robot communication

ACM Reference Format:

Qingbiao Li, Fernando Gama, Alejandro Ribeiro, and Amanda Prorok. 2020. Graph Neural Networks for Decentralized Path Planning. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020)*, Auckland, New Zealand, May 9–13, 2020, IFAAMAS, 3 pages.

1 INTRODUCTION

Efficient and collision-free navigation in multi-agent systems is fundamental to advancing mobility, where collision-free paths are generated to lead agents from their origins to designated destinations. Current approaches can be classified as either *coupled* or *decoupled*, depending on the structure of the state space that is searched. While coupled approaches are able to ensure the optimality and completeness of the solution, they involve *centralized* components, and tend to scale poorly with the number of agents [7, 8]. Decoupled approaches, on the other hand, compute trajectories for each agent separately, and re-plan only in case of conflicts [10–12]. This can significantly reduce the computational complexity of the planning task, but generally produces sub-optimal and incomplete solutions. The application of learning-based methods to multi-robot motion planning has attracted particular attention due to their capability

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand. © 2020 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

of handling high-dimensional joint state-space representations, by offloading the online computational burden to an offline learning procedure [1, 3, 4, 9]. Notably, Sartoretti et al. [6] propose a hybrid learning-based method called PRIMAL for multi-agent path-finding that uses both imitation learning and multi-agent reinforcement learning. In contrast to our work, however, the latter approach does not learn explicit inter-agent communication policies.

2 PROPOSED FRAMEWORK

We formulate the multi-agent path planning problem as a sequential decision-making problem that each agent solves at every time instant t , with the objective of reaching its destination. All components are described as following:

Partial observation processed by CNN: In an environment $(W \times H)$ with static obstacles, each agent has a local field-of-view (FOV) defined by the radius r_{FOV} , beyond which it cannot ‘see’ anything. The input map Z_t^i for agent i with size $W_{\text{FOV}} \times H_{\text{FOV}}$ is fed into a CNN that is run internally on each agent. This results in a vector $\tilde{\mathbf{x}}_t^i \in \mathbb{R}^F$ containing F observations, $\tilde{\mathbf{x}}_t^i = \text{CNN}(Z_t^i)$. These observations can then be communicated to nearby agents.

Communication via GNN: The agents can communicate with each other as determined by the communication network defined by a graph $\mathcal{G}_t = (\mathcal{V}, \mathcal{E}_t, \mathcal{W}_t)$ at time t . Note that $\mathcal{V} = \{v_1, \dots, v_N\}$ is the set of N robots, $\mathcal{E}_t \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges and $\mathcal{W}_t : \mathcal{E}_t \rightarrow \mathbb{R}$ is a function that assigns weights to the edges. Robots v_i and v_j can communicate with each other at time t if $(v_i, v_j) \in \mathcal{E}_t$, i.e. if they are within a communication radius r_{COMM} of each other. The corresponding edge weight $\mathcal{W}_t(v_i, v_j) = w_t^{ij}$ can represent the strength of the communication.

Graph Neural Network: We define a *graph convolution* [2] as linear combination of shifted versions of the signal

$$\mathcal{A}(\mathbf{X}_t; \mathbf{S}_t) = \sum_{k=0}^{K-1} \mathbf{S}_t^k \mathbf{X}_t \mathbf{A}_k \quad (1)$$

where $\{\mathbf{A}_k\}_k$ is a set of $F \times G$ matrices representing the filter coefficients combining different observations. $\mathbf{S}_t^k \mathbf{X}_t = \mathbf{S}_t (\mathbf{S}_t^{k-1} \mathbf{X}_t)$ is computed by means of k communication exchanges with 1-hop neighbors, and is actually computing a summary of the information located at the k -hop neighborhood. The operation $\mathbf{S}_t \mathbf{X}_t$ represents a linear combination of neighboring values of the signal due to the sparsity pattern of \mathbf{S}_t .

Action Policy: A local MLP (weight-sharing) is trained to predict action $\tilde{\mathbf{u}}_t^i$ taken by robot i , which is computed by a softmax over the probability distribution of motion primitives, includes up, left,

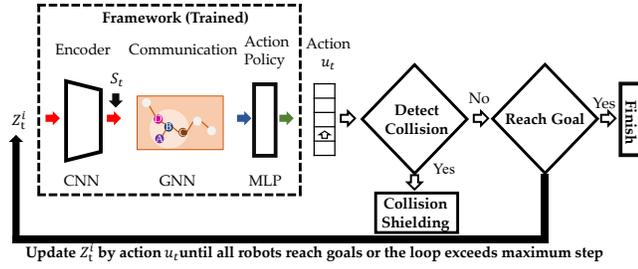


Figure 1: Illustration of the inference stage: for each robot, the input map Z_t^i with the same format as in training, is fed to the trained framework to predict its action; collisions are detected and prevented by collision shielding. The input map Z_t^i is continuously updated until the robot reaches its goal or exceeds maximum steps $T_{max} = 3T_{MP}^*$. Here, T_{MP}^* is the makespan of the solution generated by the expert algorithm.

down, right and idle. The final path is represented by a series of sequential actions.

Learning from Expert Data: To train our models, we propose an imitation learning approach based on expert data [5]. At training time, we have access to an optimal trajectory of actions $\{U_t^*\}$ for all the robots, and the corresponding maps obtained for this trajectory $\{Z_t^i\}$, collected in a training set $\mathcal{T} = \{(\{U_t\}, \{Z_t^i\})\}$. Then, we train the mapping \mathcal{F} so that the output is as close as possible to the corresponding optimal action U^* using cross entropy loss $\mathcal{L}(\cdot)$. For the mapping \mathcal{F} involving the CNN, GNN and action policy parametrizations just described, then this optimization problem becomes

$$\min_{\text{CNN}, \{A_{\ell k}\}, \text{MLP}} \sum_{(\{U_t\}, \{Z_t^i\}) \in \mathcal{T}} \sum_t \mathcal{L}(U_t^*, \mathcal{F}(\{Z_t^i\}, \mathcal{G}_t)). \quad (2)$$

We are optimizing over the filters in the CNN required to process the map and the set of matrices $\{A_{\ell k}\}$ that contains the $\sum_{\ell=1}^L K_{\ell} F_{\ell-1} F_{\ell}$ learnable parameters of the communication GNN. Note that the number of parameters is independent of the size of the network N .

Policy Execution with Collision Shielding: Fig. 1 summarizes the inference process with a protective mechanism, called *collision shielding*: robots’ actions are replaced by idle actions if the robot reaches obstacle or edge, and an inter-robot collision happens. To overcome such deadlocks caused by collision shielding, a dataset aggregation method inspired by DAGger [5] is designed to leverage an online expert to resolve hard cases.

3 PERFORMANCE EVALUATION

Metrics: 1) *Success Rate* (α) = $\frac{n_{\text{success}}}{n}$, is the proportion of successful cases over the total number of tested cases n . A case is considered successful (*complete*) when *all* robots reach their goal prior to the timeout; 2) *Flowtime Increase* (δ_{FT}) = $\frac{FT - FT^*}{FT^*}$, measures the difference between the sum of the executed path lengths (FT) and that of expert (target) path (FT*). We set the length of the predicted path $T^i = T_{max}$ (Fig. 1), if the robot i does not reach its goal.

Experimental Setup: We instantiate 600 different maps of size 20×20 with obstacle density 10%, of which 420 are used for training, 90 for validation, and 90 for testing. We generate 50 cases for each map. We set $r_{FOV} = 4$ and $r_{COMM} = 5$. Our simulations were conducted using a 12-core, 3.2Ghz i7-8700 CPU and an Nvidia

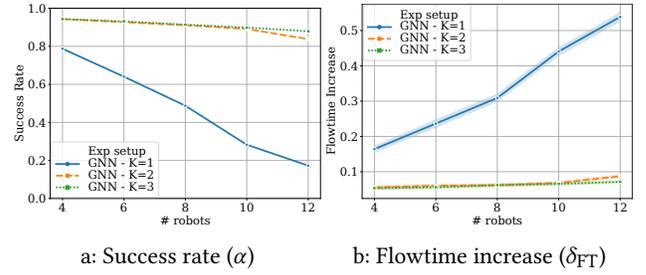


Figure 2: Results for success rate (α) and flowtime increase (δ_{FT}), as a function of the number of robots. For each panel, we vary the number of communication hops ($K \in \{1, 2, 3\}$), where $K = 1$ corresponds to no communication

	4	6	8	10	12	14
4	0.9431	0.8518	0.7531	0.5980	0.4909	0.3642
6	0.9638	0.9313	0.8747	0.7916	0.7058	0.6227
8	0.9676	0.9442	0.9140	0.8547	0.7842	0.7042
10	0.9647	0.9520	0.9249	0.8984	0.8544	0.8136
12	0.9696	0.9542	0.9347	0.9082	0.8791	0.8302

Figure 3: Success rate α (trained with $K = 3$). The rows represent the number of robots on which each model was trained, and columns represent the number of robots at test time. The generalization performance of the network is visualized by a heatmap, which maps performance values into a color range from purple to red, where purple indicates the best performance and red indicates the worst performance.

GTX 1080Ti GPU with 32 and 11GB of memory, respectively. The proposed network was implemented in PyTorch v1.1.0, and was accelerated with Cuda v10.0 APIs. We used the Adam optimizer with momentum 0.9. The learning rate γ was scheduled to decay from 10^{-3} to 10^{-6} within 150 epochs, using cosine annealing. We set the batch size to 64, and L2 regularization to 10^{-5} .

Results: Figures 2a and 2b show results for the success rate and flowtime increase, respectively, as a function of the number of robots. We train a model for $N \in \{4, 6, 8, 10, 12\}$, and test it on instances of the same robot team size. In each experiment, we vary the number of communication hops ($K \in \{1, 2, 3\}$). Note that for $K = 1$ there is no communication involved. In both figures, we see a drop in performance for larger teams, but this drop is much more pronounced for the non-communicative GNN ($K = 1$). There is a small but noticeable improvement as we increase the communication hop count from $K = 2$ to $K = 3$.

Fig. 3 summarizes the generalization capability of our model for success rates. By comparing across rows under the same column, we see that when tested on the same number of robots, the network trained on a larger number of robots tends to perform better (even better than on the same instance size as trained on), leading to higher success rates. Also, comparing performance across columns for a fixed row, we observe that as we train networks with increasingly large robot teams, the network tends to generalize better across any unseen instances (larger as well as smaller robot teams).

4 ACKNOWLEDGMENTS

We gratefully acknowledge the support of ARL grant DCIST CRA W911NF-17-2-0181. A. Prorok was supported by the Engineering and Physical Sciences Research Council (grant EP/S015493/1). We gratefully acknowledge their support.

REFERENCES

- [1] M. Everett, Y. F. Chen, and J. P. How. 2018. Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Madrid, Spain, 3052–3059. <https://doi.org/10.1109/IROS.2018.8593871>
- [2] F. Gama, A. G. Marques, G. Leus, and A. Ribeiro. 2019. Convolutional Neural Network Architectures for Signals Supported on Graphs. *IEEE Trans. Signal Process.* 67, 4 (Feb. 2019), 1034–1049.
- [3] A. Khan, E. Tolstaya, A. Ribeiro, and V. Kumar. 2019. Graph Policy Gradients for Large Scale Robot Control. *arXiv preprint arXiv:1907.03822* (2019).
- [4] A. Prorok. 2018. Graph Neural Networks for Learning Robot Team Coordination. *Federated AI for Robotics Workshop, IJCAI-ECAI/ICML/AAMAS 2018*; *arXiv:1805.03737 [cs]* (May 2018). [arXiv: 1805.03737](https://arxiv.org/abs/1805.03737).
- [5] S. Ross, G. Gordon, and D. Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. Journal of Machine Learning Research (JMLR), Fort Lauderdale, FL, USA, 627–635.
- [6] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. K. Kumar, S. Koenig, and H. Choset. 2019. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *IEEE Robotics and Automation Letters* 4 (2019), 2378–2385. Issue 3.
- [7] D. Silver. 2005. Cooperative Pathfinding. *Artificial Intelligence and Interactive Digital Entertainment* 1 (2005), 117–122.
- [8] T. Standley and R. Korf. 2011. Complete Algorithms for Cooperative Pathfinding Problems. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One (IJCAI'11)*. AAAI Press, Barcelona, Catalonia, Spain, Article 1, 6 pages. <https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-118>
- [9] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro. 2019. Learning Decentralized Controllers for Robot Swarms with Graph Neural Networks. In *Conf. Robot Learning 2019*. Int. Found. Robotics Res., Osaka, Japan.
- [10] J. van den Berg, M. Lin, and D. Manocha. 2008. Reciprocal velocity obstacles for real-time multi-agent navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, Pasadena, CA, USA, 1928–1935.
- [11] J. van den Berg and M. H. Overmars. 2005. Prioritized motion planning for multiple robots. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Edmonton, Alta., Canada, 430–435. <https://doi.org/10.1109/IROS.2005.1545306>
- [12] W. Wu, S. Bhattacharya, and A. Prorok. 2020. Multi-Robot Path Deconfliction through Prioritization by Path Prospects. *IEEE International Conference on Robotics and Automation (ICRA)* (2020). [arXiv: 1908.02361](https://arxiv.org/abs/1908.02361).