Graph-Structured Reinforcement Learning for Controlling a Transformable-Wheel Robot

Tom Tang, Neil Chulani, Anthony J. Clark

***Abstract*—Transformable-wheel robots occupy a middle ground between conventional wheeled and legged systems: they can roll efficiently on even terrain, yet reconfigure to better handle obstacles. In this work, we study reinforcement learning for controlling a transformable-wheel robot and investigate whether a graph-structured actor-critic policy provides an advantage relative to a flat multilayer perceptron (MLP). Our policy represents the robot as one body node and four corner nodes, which encodes the platform's symmetry and relational structure. We find in our experiments that graph-based policies improve early training behavior by helping the agent avoid certain local failure modes. Our results show that consistent forward locomotion can be learned and that both graph-based and MLP policies can perform well under the present task formulation, yet the graph-based policy is able to escape obstacle-relative local basins faster. We also identify the deep connection between reward task design and graph feature design for similar morphology dependent navigation tasks.**

I. INTRODUCTION

Transformable-wheel robots can roll efficiently on flat terrain while reconfiguring to handle obstacles, but learning when and how to transform remains challenging [5], [6]. In this work, we explore using reinforcement learning to control a transformable-wheel robot so that it can crawl over a step. Specifically, we investigate whether a graph-structured actor-critic policy provides an advantage over a flat multilayer perceptron (MLP) by encoding the platform's modular symmetry through message passing [1].

We represent the robot as a five-node graph (one body, four modules) where edges encode relational quantities such as wheel-speed and extension-length mismatches across modules (see Figure 2). Our experiments reveal that graph-based policies demonstrate faster early-stage coordination, avoiding symmetric failure modes (spinning, heaving) that temporarily trap flat MLPs. However, both architectures eventually achieve consistent forward locomotion under the proprioceptive-only observation space and reward formulation, suggesting the graph structure's value lies primarily in sample efficiency during exploration. These results highlight the interplay between morphology-aware policy design and task formulation for transformable robots.

II. METHODS

a) Robot and Task: The robot considered here, shown in Figure 1, consists of a central body and four wheel modules. Each module supports wheel rotation together with a rotating extension mechanism, so that the robot can transition between ordinary rolling and obstacle-climbing configurations. The control problem is to move forward and overcome a step-like obstacle by coordinating these two behaviors.

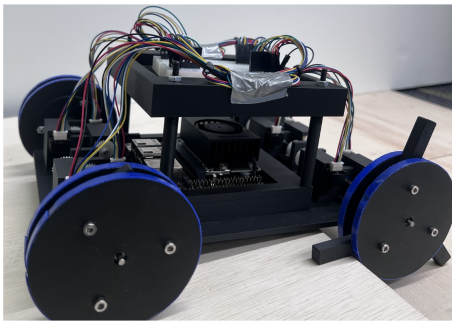

Fig. 1. Robot and obstacle setup.

b) Simulation Environment: We implemented our task in a custom MuJoCo/MJX environment. The action space corresponds to motor torques on the four wheels and four extensions. The observables include wheel speeds and body inclination. This choice provides the policy with proprioceptive measurements while not explicitly encoding features of the environment. Our reward is as follows:

$$
\begin{aligned}
r = {} & \alpha \ \texttt{Progress} \\
& + \beta \ \texttt{Velocity} \\
& + \gamma \ \texttt{Extension / Recovery} \\
& + \epsilon \ \texttt{Success} \\
& - \eta \ \texttt{control effort}
\end{aligned}
$$

c) Graph Policy: We model the robot as a graph: a central body coordinates four repeated wheel modules. Intuitively, the control rule that is useful for one corner module should be useful for the others as well, while still allowing each module to react to its own local state [3], [4], [7].

The graph structure is shown in Figure 2: one body node and four wheel nodes. The body node stores global information such as root pose and velocity. Each module node stores local information such as wheel motion and extension state. In addition, edges encode relational quantities between connected parts of the robot.

The graph-based policy can be presented as follows:

1) Parse an observation into node features h_i.
2) Encode relational quantities, such as wheel speed and extension length, as edges e_{ij}.

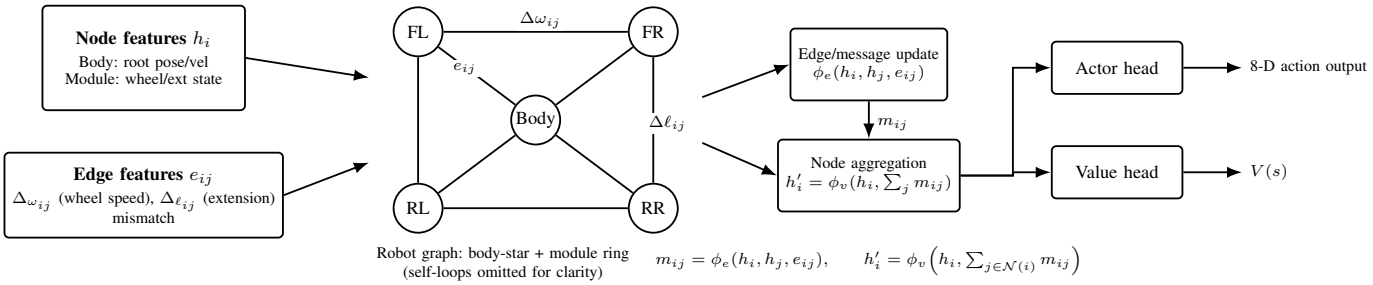

Fig. 2. A graph-structured actor-critic policy for the transformable-wheel robot. Each node carries local state features h_i. For each edge (i, j), we define relational features $e_{ij} = [\Delta\omega_{ij}, \Delta\ell_{ij}] = [\omega_i - \omega_j,\ \ell_i - \ell_j]$, where $\Delta\omega_{ij}$ is the wheel-speed mismatch and $\Delta\ell_{ij}$ is the extension-length mismatch between connected modules. Message passing first computes edge-level messages m_{ij} then aggregates them to update node embeddings h'_i.

3) Process message-passing layers m_{ij} as

$$m_{ij} = \phi_e(h_i, h_j, e_{ij}), \qquad h'_i = \phi_v\left(h_i, \sum_{j\in\text{Nodes}} m_{ij}\right)$$

4) Map the resulting output embeddings to the an eight-dimensional action vector.

Note that message passing layers only partially share network parameters and the actor headers are still independent. This graph architecture provides a morphology-aware prior: it encourages shared relational computation across repeated modules, while still allowing different parts of the robot to respond differently when their local states differ [3].

d) Training Pipeline: The robot implemented in simulation is 5 by 8 cm in size, with a height of 2 cm when the legs are fully retracted. The targeted obstacle is 5 cm tall. Training was performed with PPO [2].

III. RESULTS

Frequent early failure modes in MLP-based policies include spinning and heaving as shown in 3. *Spinning* is caused by unequal wheel speeds and/or extensions on left and right sides cause high angular velocity. *Heaving* is caused by unequal wheel speeds and/or extensions on front and back wheels cause extra energy costs to raise the robot first and then move and drop.

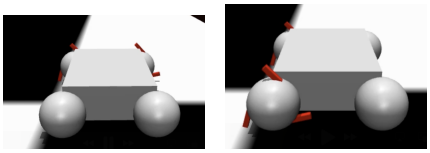

Fig. 3. Two failure modes: spinning (left) and heaving (right).

In our graph network architecture, once a favorable action is discovered in one module, it is partially shared with other modules through message passing layers. Thus, graph networks inherently avoid spinning and heaving.

This result is tightly related to reward design. We utilize dense forward rewards, constant penalties on leg extensions and stall steps, and a stall steps flag, above which an extension

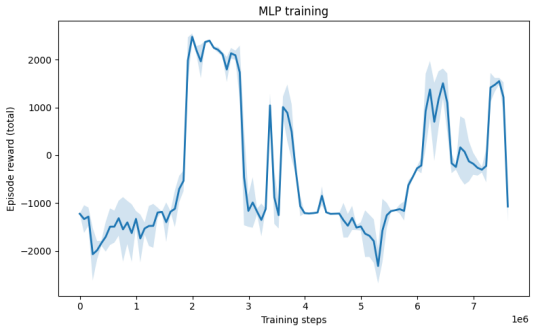

Fig. 4. Evaluated rewards during MLP training. The local basins can be understood as 1) initial coordination for forward locomotion; 2) regression after contacting the obstacle; 3) failure modes in local minimum of reward.

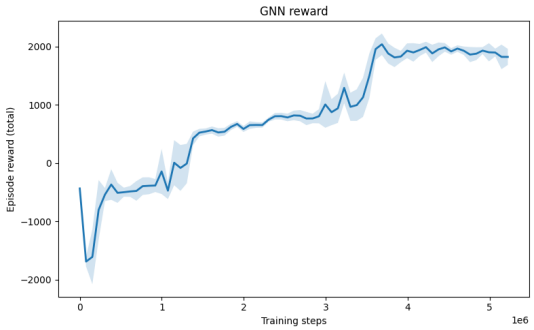

Fig. 5. Evalutation rewards of GNN policy during training.

reward turns on. The two failure modes occur when the robot learns to extend legs unevenly to overcome the obstacle. However, the message passing layer triggers connected nodes to extend legs when one node discovered so and is able to escape this basin faster.

IV. CONCLUSION

We investigated whether graph-structured policies provide advantages for transformable-wheel robots by exploiting their modular morphology. Our central finding is architectural: encoding repeated actions via graph message passing helps coordinate multi-module behaviors during early training. Com-

pared to flat MLPs, the GNN-based policy more readily escaped asymmetric failure modes and reached coordinated forward locomotion faster.

We also highlight that deeper connection between graph design and task design. The graph prior benefit could be achieved through terrain-aware observations or curriculum training. The next program is to systematically test how graph policy behaves in more complex, asymmetric terrains.

REFERENCES

[1] P. W. Battaglia et al. Relational inductive biases, deep learning, and graph networks, 2018.

[2] C. D. Lu et al. Brax: A differentiable physics engine for large scale rigid body simulation. In *NeurIPS Datasets and Benchmarks*, 2021.

[3] Deepak Pathak, Christopher Lu, Trevor Darrell, Phillip Isola, and Alexei A. Efros. Learning to control self-assembling morphologies: A study of generalization via modularity. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[4] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[5] Yu She, Carter J. Hurd, and Hai-Jun Su. A transformable wheel robot with a passive leg. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4165–4170, 2015.

[6] Tao Sun, Xu Xiang, Weihua Su, Hang Wu, and Yimin Song. A transformable wheel-legged mobile robot: Design, analysis and experiment. *Robotics and Autonomous Systems*, 98:30–41, 2017.

[7] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.