Learning Scalable Salp-Inspired Locomotion

Manuel Agraz Vallejo
Collaborative Robotics and Intelligent Systems Institute
Oregon State University
Corvallis, United States of America
agrazvam@oregonstate.edu

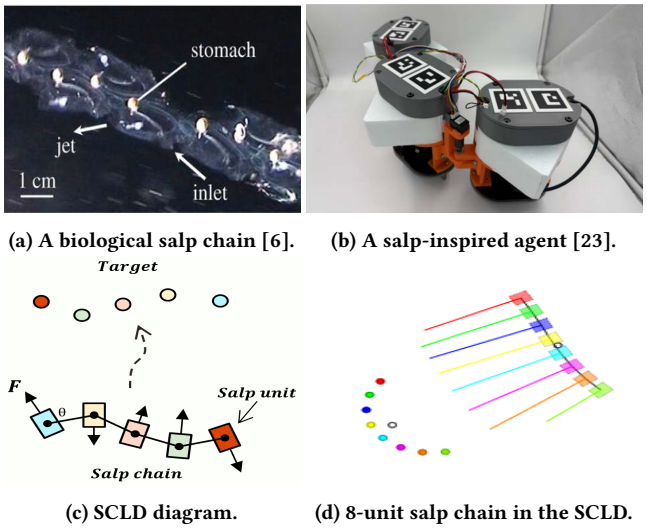
Kagan Tumer
Collaborative Robotics and Intelligent Systems Institute
Oregon State University
Corvallis, United States of America
Kagan.Tumer@oregonstate.edu

ABSTRACT

Salp-inspired agents (simple thrust-based agents that can form chains for locomotion) have the potential to monitor marine life and collect scientific samples in topologically intricate underwater habitats, such as caves, overhangs, and confined openings. For example, salp agents can modify the size of their chain structure to allow themselves to navigate through narrow bottlenecks and reach areas that would otherwise be inaccessible. However, the redundancy, non-linear characteristics, and chain structure make designing modular salp chain locomotion controllers a challenging task. Current approaches focus on controlling single salp units or small salp chains that are limited to operating within a fixed structure. In this work, we introduce a set of graph-based neuro-controllers whose structures directly map onto salp chains of arbitrary length. We develop a Salp Chain Locomotion Domain for learning the controllers' parameters and evaluating their zero-shot performance across different graph structures and aggregation mechanisms. By representing salp units as nodes in a graph, we can express the chain as a variable-sized input that naturally fits graph-based controllers regardless of chain length. This shift in perspective integrates the chain's structure directly into the controller's architecture, eliminating the need for a new controller whenever the chain grows or shrinks. Our results show that our graph-based controllers maintain strong zero-shot performance when generalizing to new chain lengths up to twice the length of the trained models.

KEYWORDS

Reinforcement Learning, Graph Neural Networks, Continuous Control



(a) A biological salp chain [6]. (b) A salp-inspired agent [23].

(c) SCLD diagram. (d) 8-unit salp chain in the SCLD.

Figure 1: From top to left: 1a shows a biological salp chain, while 1b shows a 3-unit salp chain agent capable of moving on the water's surface. Figure 1c shows a diagram of the Salp Chain Locomotion Domain (SCLD) we introduce in this work. The goal is to control each salp unit in the salp chain to reach the target pose. Each salp unit has a target pose assigned, as shown by their colors. Finally, 1d shows an 8-unit salp chain operating inside the SCLD. The projected lines from each salp unit represent the magnitude and direction of their generated force.

1 INTRODUCTION

Salps are marine organisms that propel themselves using multi-jet propulsion, and self-assemble into large chains to move efficiently underwater [3, 23]. Salp-inspired agents [7, 14, 30] offer similar advantages for long-range underwater motion, making them well-suited for extended tasks such as ocean monitoring or environmental surveys [10]. Additionally, salp *units* can be attached and detached from salp chains to reconfigure their structure, making them adaptable to dynamic environments. Two main limitations stem from the modular design of salp chain agents. The first is that the complete state of the salp chain consists of the combined limited perspective of each salp unit. The second is that the use of multi-jet propulsion leads individual salp units to have limited locomotion, as each one can only generate motion in the direction it is facing. These two limitations, added to the varying degrees of freedom of adding or removing salp units, make designing controllers for salp chain agents challenging.

Seminal research in salp-inspired agent locomotion focuses on the control of single salp units [7, 14, 31] and small salp chains (two to three units) [30, 32]. While these controllers are effective for small chains, the primary benefits of salp locomotion come from their ability to operate as much larger chains. The challenge is how to maintain effective control of the salp chain as more units are added. Each unit's dynamics are tightly coupled to every other unit, so a small perturbation from one unit can affect the entire chain.

This is where Reinforcement Learning (RL) methods have shown promising results, as they can learn controllers that handle tightly coupled, non-linear dynamics [12, 16, 21, 29]. Graph neural networks (GNNs) [11, 27] in particular are well suited for controlling intricate structures with many interconnected joints [2, 13, 28]. They aggregate information from neighboring joints to determine the control action for a single joint. However, applying them to a

Proc. of the Adaptive and Learning Agents Workshop (ALA 2026), Aydeniz, Delgrange, Mohammedalamen, Yang (eds.), May 25 – 26, 2026, Paphos, Cyprus, https://alaworkshop2026.github.io/. 2026.

salp chain introduces a unique challenge because the joints connecting individual salps are passive, and the reconstruction of the salp chain state requires accurate processing of each salp unit's information. This means that selecting how the information between units will be shared, through graph topology or aggregation mechanism, can have a significant impact on the salp chain's behavior.

In this paper, we introduce a set of graph-based controllers that embed the structure of the salp chain into a controller's architecture. Additionally, we analyze how these controllers perform in zero-shot experiments across varying salp chain sizes. To train the controllers, we introduce the Salp Chain Locomotion Domain (SCLD), a novel RL environment that simulates salp chain locomotion (see Figure 1). In the SCLD, introducing a new salp unit requires the policy to control the new unit and coordinate with the rest of the chain to maintain performance. The key insight in this work is that the graph structure captures the relationships among the independent states of the salp units, enabling the graph neural network controller to operate on arbitrary salp chain lengths.

The contributions of this work are:

(1) A set of graph-based neural network controllers that achieve scalable salp chain locomotion, enabled by a representative state space that captures the independent salp unit's state while capturing the joint locomotion potential of the salp chain.
(2) A novel learning environment, the Salp Chain Locomotion Domain (SCLD), that simulates salp chain locomotion and allows for scaling the salp chain size.

Empirical results in the SCLD environment show that graph-based controllers maintain their performance on salp chains of up to twice the length used during training; beyond that, their performance decays linearly. Furthermore, we find that both local and global aggregation mechanisms can produce graph-based controllers with strong zero-shot performance. The key is ensuring important data is not lost during aggregation.

2 BACKGROUND

2.1 Salp-Inspired Agents

Work on single salp units started with the emulation of salp jet propulsion by Bujard et al. [5]. They built a pulse-jet swimming agent whose cost-of-transport matches the ones seen in efficient pulse-jet swimmers such as jellyfish and salps. Tackling the problem of linking multiple salp units into a chain, the authors [30] developed a ground-based version of a 3-unit salp chain and developed a controller using geometric mechanics. Finally, iterating on their origami-inspired salp unit, Yang et al. [32] connected multiple salp units into a 2-unit chain, and showed that a 2-unit chain has a significant increase in speed underwater than a single unit. To control the salp chain, they use a fixed jet pulse sequence alternating at a known rate.

2.2 Reinforcement Learning

To leverage reinforcement learning methods [24], we model the salp chain locomotion problem as a Markov Decision Process (MDP). An MDP is a sequential decision-making process defined by five components: a set of states $\mathcal{S}$, actions $\mathcal{A}$, a reward function $\mathcal{R}(s)$, a transition function $\mathcal{P}(s'|s,a)$ and a policy $\pi(a|s)$. At each time step, given a state $s \in \mathcal{S}$, the agent takes an action $a \in \mathcal{A}$ following policy π. Afterwards, it receives the next state s' from the transition function $\mathcal{P}$ and finally obtains a reward from $\mathcal{R}(s') \rightarrow r$. The use of policy π induces a value function $V(s_t) = \mathbb{E}[R_t|s_t]$, where $R_t = \sum_{i=0}^{\infty} \gamma^i r_{i+1}$ is the discounted return from the episode. In this work, we utilize the Proximal Policy Optimization (PPO) algorithm to optimize the MDP's policy.

2.2.1 Proximal Policy Optimization. Proximal Policy Optimization (PPO) [20] is a reinforcement learning method that is widely used in locomotion learning tasks [12, 16, 29]. PPO collects a series of rollouts from the environment and uses them to optimize the policy using the following clipped loss function:

$$L(\theta) = \hat{\mathbb{E}}_t[\min\Big(r_t(\theta)\,\hat{A}_t, \text{clip}\big(r_t(\theta),\,1-\epsilon,\,1+\epsilon\big)\,\hat{A}_t\Big)] \tag{1}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ represents the probability ratio between the current policy and a previous copy, and $\hat{A}_t$ represents the advantage estimation using Generalized Advantage Estimation [17].

2.3 Graph Neural Networks

Graph Neural Networks (GNN) are a class of neural network architectures that can operate on any graph, both directed and undirected [11]. A graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of all nodes, and $\mathcal{E}$ is the set of all edges. The set of nodes neighboring node i is represented as $\mathcal{N}(i)$. When it comes to locomotion tasks, often the structure of the agent is partitioned into nodes, and the edges between them define their kinematic relation [2, 18, 28]. For example, in [18], the authors partitioned a legged agent such that the state of each joint became a node and each connection between joints an edge in a graph.

Graph Convolutional Networks (GCN) [15] are a type of GNN that leverages graph convolution to aggregate neighboring node features into each node in the graph. They work by updating the features h_i of each node in the graph. This update aggregates information from neighboring nodes' features x_j where $j \in \mathcal{N}(i)$ using a weight matrix $\mathbf{W}$. The following equation shows how the node features are calculated:

$$h_i = \frac{1}{\sqrt{deg(i)}\sqrt{deg(j)}} \sum_{j \in \mathcal{N}(i)} \mathbf{W} x_j \tag{2}$$

where $deg(i)$ and $deg(j)$ represent the number of connections node i and j have respectively, also called the *degree*. The degree is used to normalize the features of nodes with a high neighbor count, as they are shown to propagate more easily than those with a lower neighbor count.

Another type of GNN is the Graph Attention Network (GAT) [27]. In this work, we implement the GATv2 model [4] since it's a direct improvement over the original GAT model. The GATv2 network makes use of a simplified attention mechanism that calculates the attention score e_{ij} using the following equation:

$$e_{ij} = a^\top \text{LeakyReLU}(\mathbf{W}_l \overrightarrow{h_i} + \mathbf{W}_r \overrightarrow{h_j}) \tag{3}$$

where i is the query node, j is the key node, a is the adjacency matrix, and $\mathbf{W}_l$ and $\mathbf{W}_r$ are the query and key weight matrices respectively. This equation enables the dynamic aggregation of neighboring nodes' features.

Similar to GATv2, the Graph Transformer (GT) [8, 22] is a GNN architecture that leverages the full self-attention mechanism used in the Transformer architecture [26]. Where the attention weight between two nodes i and j is calculated with the following equation:

$$w_{ij} = \text{softmax}(\frac{Qh_i \cdot Kh_j}{\sqrt{d}}) \tag{4}$$

where h_i and h_j are the features of their respective nodes, d is the feature embedding dimension, and Q and K are the query and key embedding matrices. These weights function as a similarity measurement between node features and can capture more complex patterns between node interactions.

3 METHOD

3.1 Salp Chain Locomotion Domain

The Salp Chain Locomotion Domain (SCLD) simulates a 2D chain of linked salp units. The main goal in this domain is to translate the entire chain to a target pose that changes in shape and position each episode. In the SCLD, each salp unit can only generate forces to propel itself along one degree of freedom through forward or backward force. The links between salps connect them in a chain through revolute joints. Each salp unit is assigned a reachable individual target position that is part of the complete target pose. The main locomotion challenge comes from the restricted motion of each salp unit and the requirement to coordinate the salp units' forces to produce both translation and rotation of the entire chain. A diagram of the learning environment is shown in Figure 1c.

3.1.1 Reward Structure. SCLD's reward structure consists of a sparse goal-based reward and a dense distance-based reward. Dividing up the reward into these two components is common in continuous control tasks [1, 19, 25]. The sparse goal-based reward is not given every episode, but only at the end of episodes where the target pose is reached within a threshold. This reward allows the policy to differentiate between behaviors that reached the goal but couldn't match the target pose correctly, and behaviors that were able to do both navigation and pose adjustment. The use of a dense reward provides the policy with a learning signal even when the target pose is not fully reached. A key aspect of the implemented dense reward is that it does not reward or punish the salp chain for not moving. The reason is that in every episode, the initial salp chain pose and the target vary. This leads to the starting state having different rewards for most episodes, increasing the chances of biasing the model positively or negatively for taking no action. Thus, the dense reward is provided at every timestep as the difference between the distance to the target pose in the current timestep and the previous timestep, providing a small reward every time the salp chain moves towards (positive) or away (negative) from the target pose.

To calculate the rewards, we define the following notation, where n is the number of salp units in the chain, $\mathbf{s} \in \mathbb{R}^{n\times 2}$ represents all salp units' positions in the chain, and $\mathbf{u} \in \mathbb{R}^{n\times 2}$ represents all target poses of the goal pose. The dense reward is provided by calculating the negative mean distance error between the target and the salp chain at the current and previous timestep, and calculating their difference:

$$R_{dist}(t) = -\frac{\sum_{i=0}^{n} ||u_i^t - s_i^t||}{n} \tag{5}$$

$$R_{dense} = R_{dist}(t) - R_{dist}(t-1) \tag{6}$$

The sparse goal-based reward utilizes the discrete Fréchet distance [9] as a similarity measure between the curves defined by the target pose and the salp chain. The discrete Fréchet distance is the greatest distance between any salp unit position and the closest point in the target pose. It's worth noting that this metric can be replaced with any metric that captures curve similarity. The goal-based reward is given once the salp chain reduces the Fréchet distance between target and salp chain to fulfill this equation:

$$R_{goal} = \begin{cases} 0 & 0.95 > \frac{1}{\exp(\text{fréchet}(\mathbf{u},\mathbf{s}))} \\ 1 & 0.95 \leq \frac{1}{\exp(\text{fréchet}(\mathbf{u},\mathbf{s}))} \end{cases} \tag{7}$$

The 0.95 threshold effectively means that once the salp chain reduces the discrete Fréchet distance to a value $\epsilon \approx 0$ the task is considered complete. The final reward value per time step is calculated as the sum of both rewards:

$$R_{SCLD} = R_{dense} + R_{goal} \tag{8}$$

3.1.2 State Space. The state space in SCLD is composed of the concatenation of salp units' state information. The state of each salp unit can be seen in Table 1, which is composed of 24 different kinematic and dynamic metrics. The complete salp chain state dimension is $n \times 24$, and it scales linearly with the number of salp units in the chain.

The state space captures the salp unit's local, neighboring, and individual target information. We include local kinematic data such as linear and angular position and velocity. Additionally, we include data relative to the overall chain and target, such as positions and velocities, which helps locate the salp unit in the context of the task. Finally, the state includes data from the salp unit's neighbors, such as their relative angle and forces being generated, which gives information about the local shape of the salp chain. This effectively means that the centralized controller can only see the local view of each salp unit. By having the state space be a concatenation of local states, introducing a new salp unit only requires the concatenation of a new local state.

3.1.3 Action Space. The action space of the salp chain is composed of the possible actions of all salp units, and it has a dimension of $n \times 2$. The action space of each salp unit is composed of two actions: $a_{forward}$ determines how much the salp unit wants to go forward, and $a_{backward}$ determines how much it wants to move backward. These two action values are then used to generate the resulting force magnitude F_{unit} of a single salp unit using the following equation:

$$F_{unit} = a_{forward} - a_{backward} \tag{9}$$

The forces of all salp units determine the motion and shape of the salp chain. The translation and rotation of the salp chain

Table 1: The state space information of a single salp unit.

Salp unit state information	Dim
Salp unit binary ID	$\mathbb{Z}^4$
Position	$\mathbb{R}^2$
Velocity	$\mathbb{R}^2$
Angular position	$\mathbb{R}^1$
Angular velocity	$\mathbb{R}^1$
Relative velocity to salp chain centroid	$\mathbb{R}^2$
Relative position to salp chain centroid	$\mathbb{R}^2$
Relative position to target pose centroid	$\mathbb{R}^2$
Relative position to assigned target position	$\mathbb{R}^2$
Relative angles to salp unit neighbors	$\mathbb{R}^2$
Salp unit neighbors' forces	$\mathbb{R}^4$

is calculated at every timestep by the rigid body physics engine used by VMAS [1]. We use VMAS due to its high customizability and accessible/user-friendly API to create new continuous control learning environments.

3.2 Training a Scalable Locomotion Policy

We utilize the SCLD to train graph-based locomotion policies that can handle the growing state and action space resulting from the addition of more salp units to the chain. To train GNNs, we apply a transformation that translates the state information of the salp chain into a graph.

3.2.1 Turning states into graphs. The SCLD environment provides the state as a vector of dimension $n \times 24$. Therefore we consider each salp unit's 24-dimensional state vector as a node in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of all nodes (salp units), and $\mathcal{E}$ is the set of all edges (links). GNN implementations require all edges to have a direction, and additional edges to create self-loops. This results in the graph having a total of $n+2(n-1)$ edges for the mixed graph structure, and n^2 for the fully connected structure shown in Figure 2. The edges are only used to determine the adjacency between nodes and contain no state information.

3.2.2 The impact of modeling the salp chain as a graph. The connectivity of the salp chain graph structure determines how each node processes the data of other salp units' features. By using the mixed graph structure shown in Figure 2, each node only aggregates features from its neighbors and itself; this keeps the action selection local to each salp unit's point of view. Whether using attention or graph convolution in the GNN, the mixed graph structure reduces the number of trainable parameters required and eliminates the need to aggregate features from every node in the graph. Thus, for every new salp unit added to the chain, only its direct neighbors have to aggregate the new features. In contrast, when using a fully-connected structure as shown in Figure 2, all salp units can aggregate features from all other units. This means that when a new salp unit is added, every other salp unit has to decide how much information to aggregate from it. When using an attention mechanism such as the one used by the Graph Transformer model, a fully connected structure can help the model capture more complex interactions between all nodes. However, when using the same fully connected structure, other aggregation mechanisms, such as graph convolution in a GCN or the simpler attention mechanism of GATv2, can lead to the node features collapsing onto a similar value in a phenomenon called oversmoothing. This leads to a reduction in the overall expressiveness of the model.

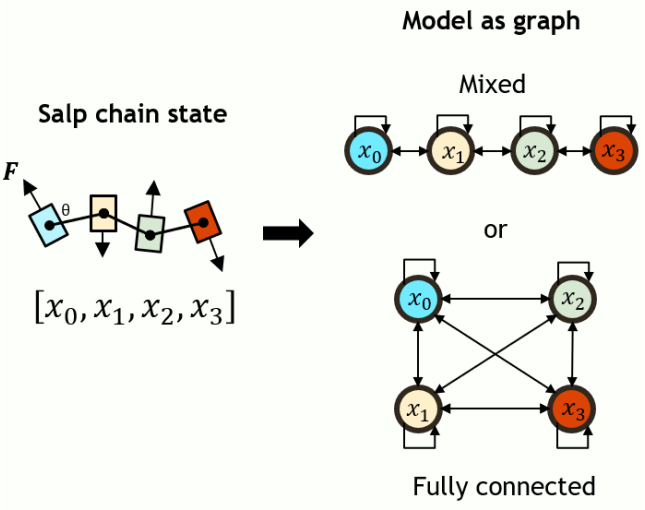


Figure 2: The mixed graph structure shown only allows salp units to aggregate information from their direct neighbors. On the other hand, the fully connected structure allows every salp unit to aggregate information from every other salp unit in the chain to make a decision.

4 EXPERIMENTS

We use the SCLD to test how different graph-based models perform when evaluated on unseen salp chain lengths. We evaluate a total of four models: an MLP to serve as a baseline for performance, a GCN, GATv2, and GT. The following list describes the set of experiments:

(1) Training of graph-based policies (GCN, GATv2, GT) using the mixed and fully connected graph structure, and an MLP policy as a performance baseline using PPO. The training is conducted using an 8 and 16 salp-unit chain.
(2) Zero-shot evaluation using policies trained on 8 and 16 salp-unit chains and executing on salp chain lengths of up to 64 salp units.

4.1 Training the graph-based controllers

We use PPO to train the four different policy architectures over five statistical runs. Both MLP and GCN policies utilize two layers with a hidden dimension of 128, for both the actor and critic. The GATv2 and GT policies use two attention heads and two layers of size 128 for the actor and critic. The GCN policy has a total of 102804 learnable parameters, the GATv2 policy has 162452, and the GT policy has 318740. All graph-based policies use attention pooling using all the graph's nodes to produce the joint state's value. The reward used during training is the SCLD reward from Equation 8. Additionally, the PPO parameters used for training can be seen in Table 2 and the detailed model parameters can be seen in Table 3.

4.2 Zero-shot experiments on unseen salp chain lengths

Using the trained policies, we conduct 50 zero-shot trials for 15 different salp chain lengths. Each size increment consists of adding 4 salp units to the chain, starting at 4 and up to a length of 64. These experiments aim to demonstrate the relevance of the aggregation mechanism (attention or convolution) and graph connectivity to the scaling performance of the tested policies. To maintain consistency across experiments as new salp chain units are added to the chain, we also increase the size and spawn distance of the target pose. Additionally, we use the distance reward from Equation 5 as it has a clear upper bound of zero, which is only achieved when the salp chain fully matches the target pose, facilitating comparison between policies.

Table 2: PPO Hyperparameters

Hyperparameter	Value
Epochs	10
Training episodes	60000
Batch size	5120
Minibatch size	256
Epsilon clip	0.2
Gradient clip	0.5
Discount factor γ	0.99
GAE λ	0.95
Entropy coefficient	0.001

Table 3: Graph Neural Networks Properties

Model	Properties	Value
All	Hidden Layers	2
	Hidden Size	128
	Learning rate	5e-5
GCN	Learnable Parameters	102804
	Aggregation Mechanism	Eq. 2
GATv2	Learnable Parameters	162452
	Attention Heads	2
	Aggregation Mechanism	Eq. 3
GT	Learnable Parameters	318740
	Attention Heads	2
	Aggregation Mechanism	Eq. 4

5 RESULTS

5.1 Training performance

In the first set of experiments, we train the MLP, GCN, GATv2, and GT policies using salp chains of 8 and 16 units. The learning curves can be seen in Figure 3. We can see that different graph models combined with different graph structures achieve different levels of training performance. The Graph Transformer policy using both graph structures was the best performer when training with both 8 and 16 salp units. A close second was the GCN and GATv2 policies when just using the mixed graph structure. This is surprising considering that both policies have a lower parameter count and are aggregating data from only the neighboring nodes. Unsurprisingly, when using the fully connected structure, both GATv2 and GCN policies struggle in both 8 and 16 salp unit cases, underperforming when compared to the MLP policy. We attribute this to the over-smoothing of node features caused by the graph convolution operator and the GATv2 attention mechanism when operating on dense graphs.

5.2 Zero-shot experiment

The second set of experiments utilizes zero-shot evaluations to analyze how the performance of each policy decays as the salp chain size increases. The results in Figure 4 show increments of 4 salp units in a range of 4 up to a 64 salp-unit chain. Since the MLP policy is restricted by the salp chain size it was trained on, it is not included in the evaluation.

Focusing on the policies trained on an 8-unit salp chain, we see similar performance decay from the GATv2, GT, and GCN policies using the mixed graph structure, and the GT policy using the fully connected structure. These policies share a performance breakpoint at the 16-unit salp chain; beyond this point, the performance decay becomes linear as the controller no longer effectively controls the entire chain. It's worth noting that while the GATv2 policy shows middling performance during training, it shows good zero-shot performance. This is caused by the GATv2 policy learning to approach the goal pose, but not being able to consistently obtain the goal reward.

The best performers from the 16-unit policies are the same as with the 8-unit policies. Observing the policies trained on the 16-unit chain in Figure 4, we see a significant difference in the performance breakpoint when compared to the 8 salp-unit policies. The best-performing 16-unit policies show a performance breakpoint at 32 units, after which decay becomes linear. This behavior indicates that by training on larger salp chains, the graph-based policies allow us to extend the performance breakpoint to twice the trained salp chain length while retaining performance on shorter chains.

6 CONCLUSION

In this work, we introduced a set of graph-based controllers for scalable salp chain locomotion using RL. Current salp-inspired agents use controllers that are limited by the fixed configuration of the hardware they were developed for. One of the main advantages of salp-inspired agents is their ability to form chains to adapt to different tasks. To learn scalable salp chain locomotion controllers, we modeled the state of the salp chain as a graph where each node represents an individual salp unit state. Using the salp chain graph model, we trained multiple graph-based controllers on a novel Salp Chain Locomotion Domain. The main task in the SCLD is to control each salp-unit in a chain to reach a desired target pose. The challenge comes from the fact that when a unit is added or removed from the chain, the controller has to adapt in order to efficiently control the entire chain.

Our results show that on zero-shot evaluations on different salp chain lengths, a Graph Convolutional Network using a mixed graph

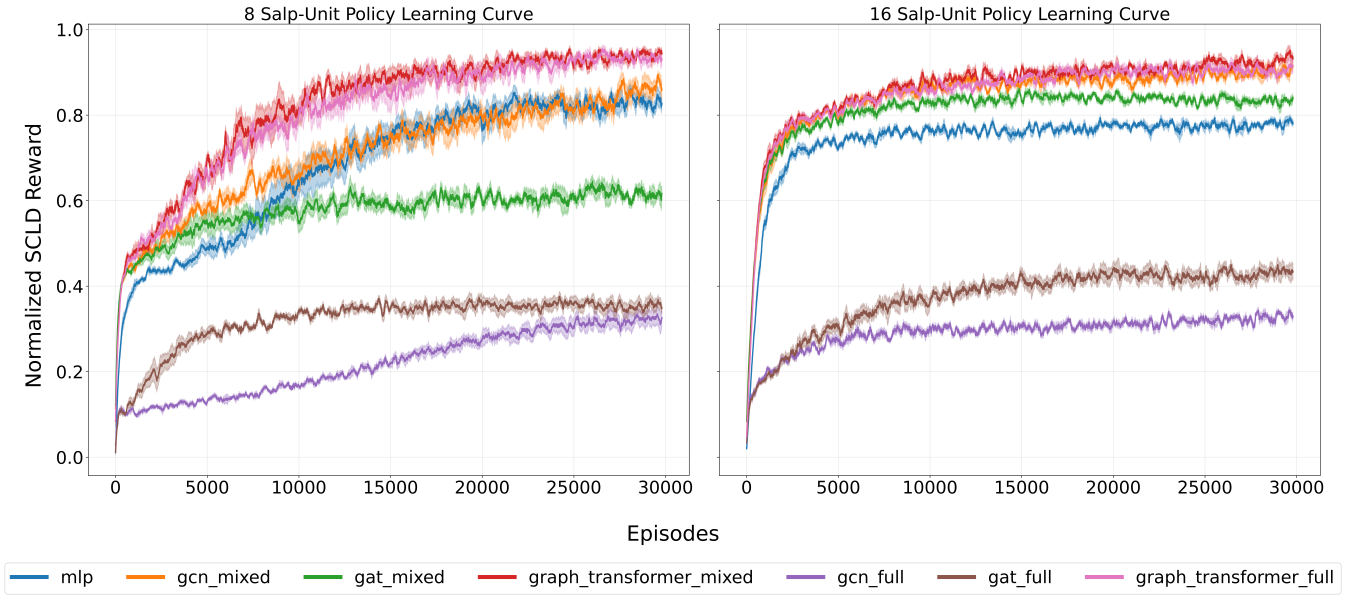


Figure 3: Learning curves with standard mean error of policies trained with 8 and 16 salp-unit chains. We use a moving average of 200 data points and show the standard mean error over five statistical runs. The plots show the normalized rewards across both salp chain lengths. We can see that there's a wide performance gap between the GCN and GAT models using the fully connected topology when compared to the rest of the models. This empirically shows how the graph structure can negatively affect the controlability of the salp chain when using simpler aggregation mechanisms.

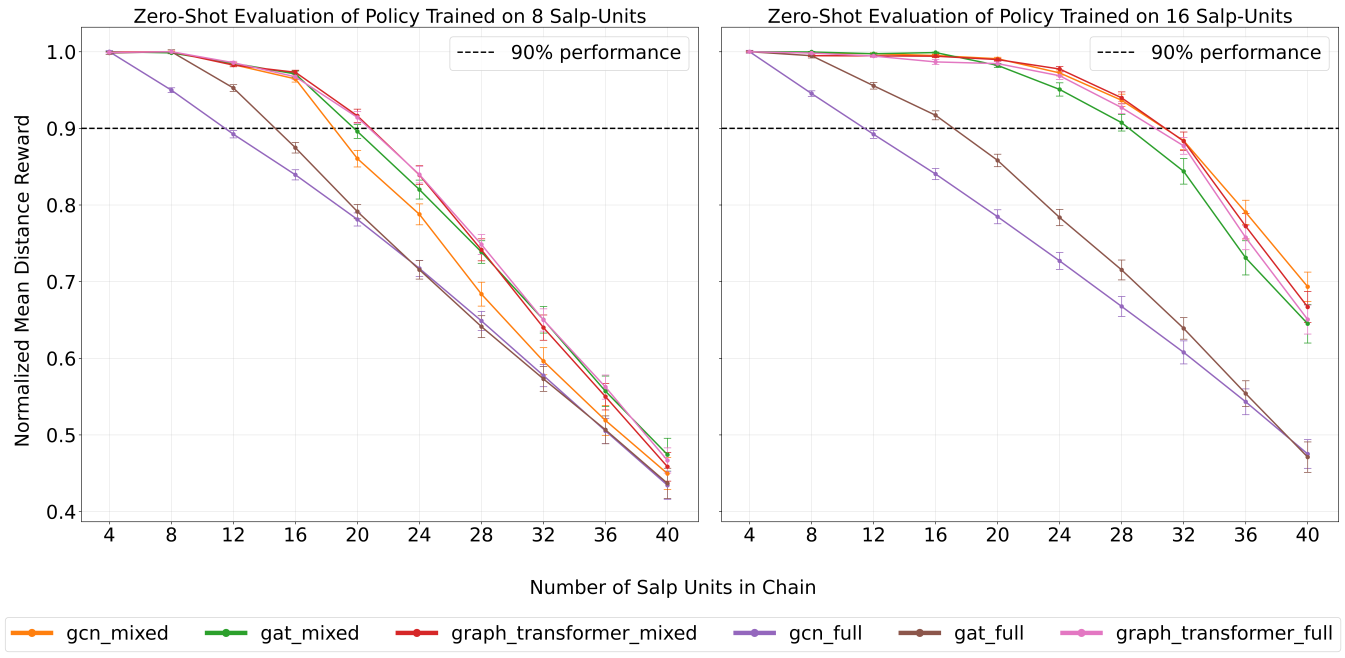


Figure 4: Zero-shot evaluations of policies trained on 8 and 16 salp-unit chains across varying salp chain lengths. The plots show the mean and standard deviation over 50 trials for every salp chain length. A normalized mean distance reward of 1.0 implies that the salp chain was able to match the target pose fully. We use a threshold line to determine how many salp units each controller can support while maintaining above 90% of the maximum reward. We can see that the best performing model was the graph transformer using both the mixed and fully connected topologies. It was able to retain above 95% of the maximum performance while operating on double the chain size seen during training.

structure reaches similar performance breakpoints when compared to higher complexity models such as the Graph Transformer using both a mixed and fully connected structure. Additionally, we find that the best-performing graph-based controllers can maintain performance during zero-shot evaluation in salp chain lengths of up to twice the length used for training. Furthermore, we show that by training on longer salp chains, we can achieve high zero-shot performance in shorter salp chains. These results highlight the value of graph-based models for salp chain locomotion, as they open the path for training controllers on specific salp chain configurations and generalizing to a much broader range of chain lengths.

ACKNOWLEDGMENTS

I would like to thank my advisor, Kagan Tumer; my friends at OSU, and my labmates at the AADI Lab

REFERENCES

[1] Matteo Bettini, Ryan Kortvelesy, Jan Blumenkamp, and Amanda Prorok. 2024. VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning. In *Distributed Autonomous Robotic Systems : 16th International Symposium (DARS 2022) (Springer Proceedings in Advanced Robotics, Vol. 28)*. Springer, Montbéliard, France, 42–56. https://doi.org/10.1007/978-3-031-51497-5_4

[2] Charlie Blake, Vitaly Kurin, Maximilian Igl, and Shimon Whiteson. 2021. Snowflake: Scaling GNNs to High-Dimensional Continuous Control via Parameter Freezing. In *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*. 23983–23992. https://proceedings.neurips.cc/paper/2021/file/c952ce98517ac529c60744ac28364b03-Paper.pdf

[3] Q. Bone and E. R. Trueman. 1983. Jet propulsion in salps (Tunicata: Thaliacea). *Journal of Zoology* 201, 4 (1983), 481–506. https://doi.org/10.1111/j.1469-7998.1983.tb05071.x

[4] Shaked Brody, Uri Alon, and Eran Yahav. 2022. How Attentive are Graph Attention Networks?. In *Proceedings of the 10th International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=F72ximsx7C1 Poster.

[5] Thierry Bujard, Francesco Giorgio-Serchi, and Gabriel D. Weymouth. 2021. A Resonant Squid-Inspired Robot Unlocks Biological Propulsive Efficiency. *Science Robotics* 6, 50 (January 2021), eabd2971. https://doi.org/10.1126/scirobotics.abd2971

[6] Jinwoo Choi. 2025. *Gait-family-based hierarchical control of kinematic locomoting systems*. Ph.D. Dissertation. Oregon State University.

[7] Xue Dong, Hao Chen, Zhitong Zhou, Chen Ouyang, Lei Hu, Fang Zhang, Bo Chen, and Zhen Gan. 2024. Salpot: A Jet Propulsion Swimmer with Scissor Structure and Bilateral Apertures. *IEEE Robotics and Automation Letters* 9, 8 (Aug. 2024), 7102–7109. https://doi.org/10.1109/LRA.2024.3418278

[8] Vijay Prakash Dwivedi and Xavier Bresson. 2020. A Generalization of Transformer Networks to Graphs. *arXiv preprint arXiv:2012.09699* (2020). arXiv:2012.09699 [cs.LG] https://arxiv.org/abs/2012.09699

[9] Thomas Eiter and Heikki Mannila. 1994. *Computing the Discrete Fréchet Distance*. Technical Report CD–TR 94/64. Christian Doppler Laboratory for Expert Systems, Technische Universität Wien, Vienna, Austria. https://www.kr.tuwien.ac.at/staff/eiter/et-archive/cdtr9464.pdf

[10] Dennis P. Gordon, Jennifer Beaumont, Alison MacDiarmid, Donald A. Robertson, and Shane T. Ahyong. 2010. Marine Biodiversity of Aotearoa New Zealand. *PLoS ONE* 5, 8 (2010), e10905. https://doi.org/10.1371/journal.pone.0010905

[11] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A New Model for Learning in Graph Domains. In *Proceedings of the 2005 IEEE International Joint Conference on Neural Networks (IJCNN '05)*, Vol. 2. IEEE, Montreal, QC, Canada, 729–734. https://doi.org/10.1109/IJCNN.2005.1555942

[12] Yaqi Guo and Haijun Peng. 2021. Full-Actuation Rolling Locomotion with Tensegrity Robot via Deep Reinforcement Learning. In *2021 5th International Conference on Robotics and Automation Sciences (ICRAS)*. 51–55. https://doi.org/10.1109/ICRAS52289.2021.9476651

[13] Wenlong Huang, Igor Mordatch, and Deepak Pathak. 2020. One Policy to Control Them All: Shared Modular Policies for Agent-Agnostic Control. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 4455–4464. https://proceedings.mlr.press/v119/huang20d.html

[14] Ali Jones and J. R. Davidson. 2024. Underwater Salp-Inspired Soft Structure Contraction with Twisted Coiled Actuators. In *Proceedings of the 7th IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, San Diego, CA, USA, 504–510. https://doi.org/10.1109/RoboSoft60065.2024.10522044

[15] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=SJU4ayYgl

[16] Ashish Kumar, Zhongyu Li, Jun Zeng, Deepak Pathak, Koushil Sreenath, and Jitendra Malik. 2022. Adapting Rapid Motor Adaptation for Bipedal Robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 1161–1168. https://doi.org/10.1109/IROS47612.2022.9981091

[17] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. 1928–1937.

[18] Gaukhar Nurbek. 2024. *Exploring Graph Neural Networks in Reinforcement Learning: A Comparative Study on Architectures for Locomotion Tasks*. Master's Thesis. The University of Texas Rio Grande Valley, Edinburg, TX, USA. https://scholarworks.utrgv.edu/etd/1493

[19] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2021. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems* 34 (2021), 12208–12221.

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[21] Junyao Shi, Tony Dear, and Scott David Kelly. 2020. Deep Reinforcement Learning for Snake Robot Locomotion. In *21st IFAC World Congress*. 9688–9695. https://doi.org/10.1016/j.ifacol.2020.12.2581

[22] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjing Wang, and Yu Sun. 2021. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, Zhi-Hua Zhou (Ed.). International Joint Conferences on Artificial Intelligence Organization, 1548–1554. https://doi.org/10.24963/ijcai.2021/214 Main Track.

[23] Kelly R. Sutherland and Daniel Weihs. 2017. Hydrodynamic advantages of swimming by salp chains. *Journal of the Royal Society Interface* 14, 133 (2017), 20170298. https://doi.org/10.1098/rsif.2017.0298

[24] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems 12*. MIT Press, 1057–1063.

[25] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 5026–5033. https://doi.org/10.1109/IROS.2012.6386109

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, Vol. 30. 5998–6008.

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018). https://openreview.net/forum?id=rJXMpikCZ

[28] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. 2018. NerveNet: Learning Structured Policy with Graph Neural Networks. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=S1sqHMZCb

[29] Yuliu Wang, Ryusuke Sagawa, and Yusuke Yoshiyasu. 2024. Learning Advanced Locomotion for Quadrupedal Robots: A Distributed Multi-Agent Reinforcement Learning Framework with Riemannian Motion Policies. *Robotics* 13, 6 (May 2024), 86. https://doi.org/10.3390/robotics13060086

[30] Yanhao Yang, Nina L. Hecht, Yousef Salaman-Maclara, Nathan Justus, Zachary A. Thomas, Farhan Rozaidi, and Ross L. Hatton. 2025. Geometric Data-Driven Multi-Jet Locomotion Inspired by Salps. *arXiv preprint arXiv:2503.08817* (2025). https://doi.org/10.48550/arXiv.2503.08817 arXiv:2503.08817 [cs.RO]

[31] Zhiyuan Yang, Dongsheng Chen, David J. Levine, and Cynthia Sung. 2021. Origami-inspired robot that swims via jet propulsion. *IEEE Robotics and Automation Letters* 6, 4 (2021), 7145–7152. https://doi.org/10.1109/LRA.2021.3097757

[32] Zhiyuan Yang, Yipeng Zhang, Matthew Herbert, M. Ani Hsieh, and Cynthia Sung. 2025. Effect of Jet Coordination on Underwater Propulsion with the Multi-Robot SALP System. In *2025 IEEE 8th International Conference on Soft Robotics (RoboSoft)*. 1–8. https://doi.org/10.1109/RoboSoft63089.2025.11020967