RoboNet: A Sample-Efficient Robot co-design Generator

Kishan Reddy Nagiredla, Arun Kumar AV and Santu Rana
Applied Artificial Intelligence Institute
Deakin University
Melbourne, Australia
Email: knagiredla@deakin.edu.au

Thommen George Karimpanal
School of Information Technology
Deakin University
Email: thommen.karimpanalgeorge@deakin.edu.au

***Abstract*—Co-design of robots involves designing the control mechanism and physical form together. This intertwined design process is inherently challenging and sample-inefficient because of the large design and control search spaces. Our key idea is to navigate this combinatorial search space more intelligently than earlier methods to reduce dependence on excessive samples while still generating capable, complex designs. Our proposed framework RoboNet, leverages a GFlowNet-based approach, a recent advancement in graph synthesis to explore the vast, high-dimensional space of robot co-designs efficiently. RoboNet learns to generate robots from scratch and evaluates designs based only on partially trained control policies. Specifically, we use GFlowNet in combination with a rate-based design prioritizing and cost-aware sampling strategy to evaluate robots based on the future promise under full-training and across different complexities. Our experiments show the proposed framework's utility in various robot design tasks.**

I. Introduction

Designing robots involves numerous challenges, from defining the robot's structure to integrating various components and developing control algorithms. Traditional approaches often tackle these challenges sequentially, but recent advancements have led to more integrated methodologies commonly referred to as co-design methods [10]. Co-design proposes simultaneously optimizing a robot's physical structure and control signals to achieve a target behavior [5]. However, the vast search space of possible configurations and the interdependence of design and control behavior make co-design costly and complex.

To address these challenges, researchers have increasingly turned to machine learning (ML) techniques. Relying on learning from data has shown promise in several areas of robotics including morphology and control optimization [11], robotic manipulation [9], locomotion [7], and perception and decision-making [14]. Integrating co-design principles with machine learning techniques promises to create bespoke robots that are truly novel [2]. Recent work from Ha [5] optimizes an initial design by parameterizing parts of the design as part of the environment and uses a genetic algorithm to learn a policy-parameter combination. Other notable work such as Transfrom2Act [19], learned a parameter-attribution policy using RL in the outer loop by employing a graph neural network capable of making design and control decisions over a given initial design. To address the inherent sample-inefficiency of Transform2Act, SARD [4] identifies symmetrical designs in the design space to traverse a much smaller space and applies a similar learning strategy as Transform2Act. While this may limit their designs in challenging environments, they can handle simpler environments sample-efficiently. Unfortunately, most methods do not optimize across different morphologies, and the few that do so [5], [19], [4] often produce a single optimal design. Single optimal design is often less useful in practice where multi-objective optimisation dominates necessiating discovery of several design options for enforcing trade-offs later on. Hence, efficiently exploring the vast design space to identify such diverse design candidates remains a significant challenge in co-design methods. In this context, a recently developed probabilistic machine learning framework known as GFlowNet (Generative Flow Network) has emerged as a promising approach to address these limitations [1].

GFlowNets represent a recent advancement in the generative modeling of graphs where a graph is constructed step-by-step, deciding at each step whether to stop the construction or to grow by selecting the node of the existing graph and the component to attach. In GFlowNets, a trajectory (or sequence of decisions) from an initial state (source) to a terminal state (sink) is akin to a path in a flow network (imagine a water pipe network). The model assigns a flow (probability) to each possible path, with paths that lead to a higher reward receiving greater flow. This ensures that: a) paths leading to high-reward terminal states are sampled more frequently, and b) flow is spread across different trajectories, allowing for diverse solutions rather than a single optimal one. During training, the policy network adjusts the flow at each state to ensure that it is consistent with both the reward and the flow conservation property. This way, the model learns a probability distribution over trajectories that encourages diverse, high-reward graphs. GFlowNets have already been explored in tasks requiring diverse and high-quality solutions, such as drug discovery [8] and material design [3]. We believe it can also offer significant benefit when adopted appropriately for robot design.

In this study, we introduce RoboNet, a novel sample-efficient co-design strategy that uses GFlowNets at its core. In a co-design process, one must deal with the core issue of finding the utility of a design especially because the budget for learning control mechanisms for each design may be small to make it feasible to evaluate a large number of co-designs. In such a scenario the control mechanism would be only partially learned and we have to estimate the utility of the design under a fully-learned control policy from the performance of the design under the partially-learned control policy. In addition, as we deal with robots having different morphological structures some more complex (i.e. more nodes) than others we need to provide them not with equal control learning budgets but equivalent budgets. The complex design should use higher resources so that the degree of maturity of control learning is similar to that of a simpler design so that they can be compared against each other. We solve these issues in the following ways (also illustrated in Fig. 1):

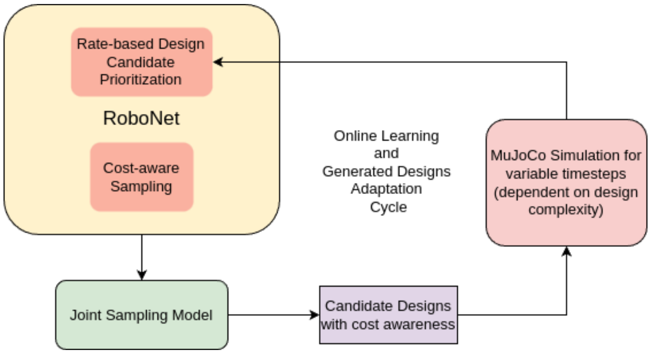
Rate-based Design Candidate Prioritization
RoboNet
Cost-aware Sampling
Online Learning and Generated Designs Adaptation Cycle
MuJoCo Simulation for variable timesteps (dependent on design complexity)
Joint Sampling Model
Candidate Designs with cost awareness

Fig. 1: RoboNet Framework illustrating the ideas of Joint Sampling, Rate-based Design Candidate Prioritization, and Cost-aware sampling strategies.

- Rate-Based Prioritization: When providing a suitable reward to GFlowNet, it may be tempting to provide the current sum of rewards value of the partially-trained robots, however, this value may not be indicative of future performance with full training. A good future performance is often indicated by the rapid rate of growth in reward instead of saturation to a high value. Thus, RoboNet employs a rate-based approach to prioritize design candidates that showcase a steep learning curve.
- Cost Aware Designs: RoboNet employs a cost coding mechanism where the robots are optimized in the joint space of performance and the training budget used. In essence, the use of higher cost-coded components are allowed only if the penalty of using them is outweighed by the marginal gain in performance.

We implement our method in the Ant-v5 OpenAI MuJoCo environment [16] where we modify the terrains to make the walking task challenging. We show how our contributions help produce robots that learn to walk sample-efficiently while the designs from other methods struggle.

II. Related Works

The field of robot co-design has advanced significantly, with various approaches emerging to optimize both morphology and control policies simultaneously. Early work by Von Neumann, in 1966, laid the foundation for co-design through evolutionary mechanisms, which Karl Sims extended on in 1994 to demonstrate how evolutionary strategies could generate intelligent virtual agents. While some subsequent works retained the essence of Sims' approach, focusing on skeletal structure optimization, others optimized over pre-selected skeletons. However, evolutionary methods such as those in Howard et al. [6] and Yu et al. [18] are highly sample-inefficient, as each design in the population requires evaluation.

To address this inefficiency, recent works have proposed more sample-efficient strategies. David Ha's work [5] introduced joint learning of policy and physical structure using a parameterized environment and the implicit function theorem, allowing task-specific design adaptations. This approach highlights the adaptability of co-designed agents but still faces challenges when exploring vast design spaces. Other methods, such as Schaff et al. [15] and Luck et al. [11], used policy transfer mechanisms to reduce the evaluation cost of evolving new designs, transferring learned policies from previous designs to new ones. However, these strategies still struggle with generalization across different designs and dynamic environments.

A notable alternative is Transform2Act [19], which uses graph neural networks to encode robot designs and RL to optimize control policies. Although Transform2Act effectively integrates morphology and control, it remains computationally demanding and struggles with sample efficiency in high-dimensional spaces. Model-based methods [17, 13] also offer solutions but lack robustness when faced with changes to design parameters or environment dynamics, emphasizing the need for more generalizable approaches. Another work Symmetry-aware design framework (SARD) [4] has introduced additional innovations by leveraging structural properties to simplify optimization, improving sample efficiency. While effective in structured domains, SARD is limited by it's dependence on symmetry. GFlowNets [1] have been proposed as a promising solution to generate diverse candidates in complex design spaces by addressing mode collapse. Initially applied to molecular design [8], GFlowNets have shown potential for robot co-design but are hindered by challenges in sample efficiency and the exploration-exploitation balance.

RoboNet builds upon these foundational works by introducing rate-based design candidate prioritization and cost-aware sampling, significantly improving sample efficiency compared to methods like Transform2Act. RoboNet's adaptive sampling strategies, integration of domain-specific heuristics, and novel reward formulation enable more effective exploration of the design space while balancing performance and resource constraints. Unlike SARD, which is constrained by symmetry, RoboNet offers a flexible framework adaptable to a broader range of design challenges. By addressing key limitations in previous co-design methods, RoboNet paves the way for more efficient and diverse robot designs in scenarios where unconventional configurations may offer significant advantages.

III. Background

A. Reinforcement Learning

Reinforcement Learning (RL) problems are modeled as a Markov Decision Process (MDP), represented as $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, R \rangle$, where $\mathcal{S}$ is the state-space, $\mathcal{A}$ is the action-space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the transition function governing the next state reached by taking an action $a \in \mathcal{A}$ in a state $s \in \mathcal{S}$, and $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the scalar-valued reward producing function for taking action a in state s.

The learning problem is to find the optimal policy that maximizes the returns $\mathbb{E}_{\tau \sim \pi} r(\tau)$, where π is the policy, τ is a trajectory sampled from π, and $r(\tau) = \sum_{(s,a) \in \tau} R(s, a)$ is shorthand for sum of the rewards over the trajectory τ.

B. Generative Flow Networks

GFlowNets aim to generate diverse, high-reward candidates in structured spaces by learning a policy that samples from a distribution proportional to a reward function $\bar{R}(\bar{s}_T)$. This approach is distinct from traditional generative models, as it explicitly seeks diversity, critical in robot co-design where multiple design candidate solutions could be possible, and is beneficial to explore.

1) Definition: A GFlowNet defines a trajectory $\bar{\tau} = (\bar{s}_0, \bar{a}_0, \ldots, \bar{s}_T)$ in a Markov decision process (MDP) with states $\bar{s} \in \bar{\mathcal{S}}$ and actions $\bar{a} \in \bar{\mathcal{A}}$. The objective is to sample terminal states $\bar{s}_T$ from policy space P according to:

$$P(\bar{s}_T) \propto \bar{R}(\bar{s}_T) \tag{1}$$

The *flow consistency condition* ensures balance across states:

$$F(\bar{s}) = \sum_{\bar{a} \in \bar{\mathcal{A}}} P(\bar{s}' \mid \bar{s}, \bar{a}) F(\bar{s}') \tag{2}$$

where $F(\bar{s})$ is the flow through state $\bar{s}$.

2) Learning Objective : To ensure the sampling model learns a policy that can generate samples according to a target distribution, GFlowNets uses the Trajectory Balance (TB) objective [12], given by:

$$\mathcal{L}_{\text{TB}}(\bar{\tau}; \theta) = \left(\log \left(\frac{Z_\theta \prod_{\bar{s} \rightarrow \bar{s}' \in \bar{\tau}} P_{F_\theta}(\bar{s}' \mid \bar{s})}{\bar{R}(\bar{s}_T)} \right) \right)^2 \tag{3}$$

where Z_θ is the partition function ensuring consistency across state transitions. TB aims to ensure that the marginal likelihood of a trajectory terminating at a terminal state $\bar{s}_T$ becomes proportional to the reward $\bar{R}(\bar{s}_T)$ through efficient credit assignment and robustness to long trajectories.

3) Application to co-design: In robot co-design, GFlowNets can generate diverse designs, avoiding the mode collapse typical of RL approaches. The reward $\bar{R}(\bar{s}_T)$ can reflect task-specific performance, and GFlowNets effectively balance exploration and exploitation via flow consistency, optimizing sampling in high-dimensional spaces.

Our method, RoboNet, extends GFlowNet by introducing rate-based boosting and cost-aware sampling, improving sample efficiency and addressing the challenges of high-dimensional co-design spaces in the robotics domain.

IV. Methodology

We aim to search a discrete space of robot designs $\mathcal{X}$ to find designs $\mathbf{x}_i \in \mathcal{X}$ that can perform a given task effectively. Given the complexity of the robot design search space, a naive random search for $\mathbf{x}_i$ can be impractical, thus, advanced search techniques must be employed. This section outlines the key components and innovations in our approach to tackle this high-dimensional search space.

A. RoboNet Framework and Contributions

In RoboNet, we build upon the GFlowNet framework, which aims to sample from a distribution proportional to a given reward function. In the context of robot co-design, we define our state space $\mathcal{X}$ as the set of all possible robot configurations, including both morphology and control parameters. The action space $\bar{\mathcal{A}}$ represents the design decisions that can be made at each construction process step. This decision/action space is a high-dimensional space composed of multiple lengths, sizes, joint gear ratios, and slot options for the insertion of each new link.

The core objective of RoboNet is to learn a policy that generates terminal states $\bar{s}_T$ (complete robot designs) according to $P(\bar{s}_T) \propto \bar{R}(\bar{s}_T)$, where $P(\bar{s}_T)$ is the policy and $\bar{R}(\bar{s}_T)$ is the reward function evaluating the performance of the complete robot design. To train RoboNet, we employ the Trajectory Balance (TB) objective (described in 3). Following are the contributions RoboNet proposes for co-designing robots:

1) Rate Based Design Candidate Prioritization: To address the sample inefficiency of standard GFlowNets, we introduce rate-based prioritization. This prioritization module generates an initial set of candidate designs, evaluates their performance over a limited number of timesteps in a simulator, and updates the RoboNet policy to bias towards designs with higher performance rates during the training period. This approach can be formalized as:

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \mathbb{E} \bar{s}_T \approx P\theta[\nabla_{\bar{s}_T} \bar{R}(\bar{s}_T)] \tag{4}$$

where α is the learning rate, P is the policy space and $\nabla_{\bar{s}_T} \bar{R}(\bar{s}_T)$ is the rate of change of the reward with respect to the terminal state $\bar{s}_T$. In the context of robot co-design, this rate indicates how quickly the robot's performance improves during training specifically immediately before the cutoff timesteps window (i.e. the evaluation during the last n timesteps). A steeper rate indicates that the robot design has a comparatively greater potential for rapid improvement over the long run. Thus, through RoboNet, we prioritize and bias policy updates toward robot designs that have the potential to keep learning and reach higher reward eventually.

2) Cost Aware Sampling: Allocating a fixed amount of training resources to all candidate designs can lead to two issues: (i) If the allocated timesteps window is too low, the generator may become biased toward producing simpler designs that underperform when given more resources, due to learning behaviors only beneficial in the short-term. (ii) If the allocated window is too high, unpromising designs receive excessive resources, resulting in sample inefficiency and potential difficulties in convergence. It is also challenging to pin down what the right resource level might be for fair evaluation per task. Hence, we present our cost-aware sampling strategy which is formalized as:

$$P_{sample}(\bar{s}_T) \propto \bar{R}(\bar{s}_T) - wC(\bar{s}_T) \tag{5}$$

where $C(\bar{s}_T)$ is a cost function reflecting the computational complexity of evaluating design $\bar{s}_T$, and w is a hyperparameter controlling the trade-off between performance and evaluation cost. Through this formulation, we enforce the idea that designs will be sampled proportionally to their performances, with a penalty associated with larger resource consumption. Thereby, we integrate the $C(x)$ into RoboNet's sampling space. Thus, we encourage RoboNet to learn a joint sampling strategy to optimize effectively over the robot design search space and cost for each evaluation.

Algorithm 1 RoboNet co-design Algorithm

```
Init: Policy Parameters $\theta$, Partition function $Z$, Design Space $\mathcal{X}$, Policy Space $P$, Environment $E$, Batch Size $M$, Max Training timesteps $K$
while not converged do
    Sample batch $\{\bar{\tau}_i\}_{i=1}^{M}$ using Eq. 1 and Eq. 6
    for $\bar{\tau}_i = (d_i, p_i, r_i)$ in batch do
        // $d_i$: design, $p_i$: policy, $r_i$: reward
        Update Reward: $\bar{R}(\bar{s}_T) = f(\text{Performance in } E(d,p), \text{complexity strategy}(d))$ using Eq. 4 and Eq. 5
        Calculate loss $\mathcal{L}(\bar{\tau}_i)$ using TB objective (Eq. 3)
    end for
    Update $\theta, Z$ via gradient descent on $\mathcal{L}(\bar{\tau}_i)$
    Adjust sampling probabilities for promising design regions
    if converged then
        Sample final batch using the updated sampler
        Train batch using timesteps $K$
    end if
end while
return top performance designs $d^*$ and policies $p^*$
```

B. Exploration vs Exploitation

To balance exploration and exploitation, we use the following technique that adjusts the sampling distribution based on the number of designs sampled:

1) Annealing Exploration: In the standard GFlowNet framework, an inverse temperature parameter β is employed to control the balance between exploration and exploitation in the forward policy [1]. Higher values of β lead to more exploitative behavior, focusing on high-reward states, while lower values encourage more exploration of the overall state space. In RoboNet, we introduce a novel formulation for β that adapts over the course of its training:

$$\beta_{decay} = \gamma\beta_{init} + (1-\gamma)\beta_{init}(1-d_v)^{(t-d_d)/d_s} \tag{6}$$

where,

$$\gamma(d_d, t) = \begin{cases} 0 & \text{if } t \geq d_d \\ 1 & \text{if } t < d_d \end{cases} \tag{7}$$

where t is the current iteration, d_v is the selected window after which the annealing starts, d_s is the decay step size, and d_d is the decay delay parameter which controls when the gradual decrease in exploration starts. The gradual decay in this formulation ensures a smooth transition from exploration to exploitation, avoiding abrupt changes in the sampling distribution that could destabilize training.

By incorporating this adaptive temperature schedule, RoboNet aims to efficiently explore the vast robot design space in the early stages of training, while progressively focusing on high-performing regions as training progresses.

This temperature-controlled forward policy, combined with our rate-based candidate prioritization and cost-aware sampling strategies, enables RoboNet to navigate the complex robot design space more effectively than standard GFlowNets or traditional optimization approaches to address the robot co-design problem.

V. Toy Experiment

In the toy experiment, instead of using a simulator, we use a simple reward function that incentivizes RoboNet to generate the desired robot designs. The problem is setup such that the action space includes 5 different lengths, and 5 different size options, totalling to a 25-dimensional space. Specifically, the objective is to model robots with the longest chain of interconnected bodies. To achieve this, RoboNet must select actions that progressively increase the overall length of the robot. The action space includes discrete length options for each selected body, requiring RoboNet to choose robot bodies with the highest length values and form the graph such that the graph has a long branch. In each iteration, the observed lengths of each design candidate are collected and averaged to calculate the mean value and is plotted against the longest possible robot length in Fig. 2. The reward is given as 10 times the length of the robot.

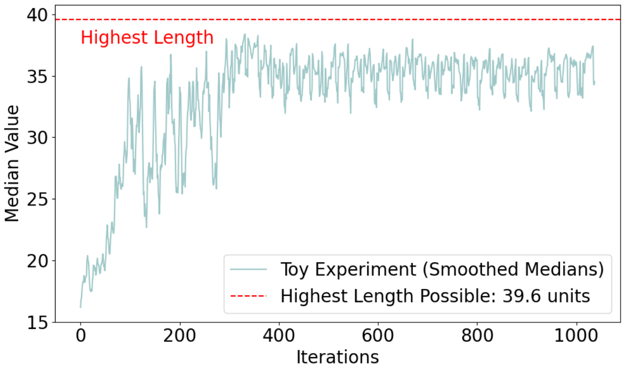

Fig. 2: RoboNet training curves for the toy experiment, where Median length (per batch) of the robots is plotted against the iteration. The dotted red line shows the maximum robot length possible.

Fig. 2 shows the median robot length per batch (batch size = 32) after 1000 iterations. Remarkably, RoboNet was able to sample 14 distinct longest (i.e. 39.6 units) robots as shown in Fig. 3. Hence, instead of converging onto one design that follows the reward function for reward maximization, we have

14 distinct top performers which vary in other properties like size and angle of connection. In the following section, we present complex experiments that use a simulator for feedback on the quality of designs instead of a toy reward function.

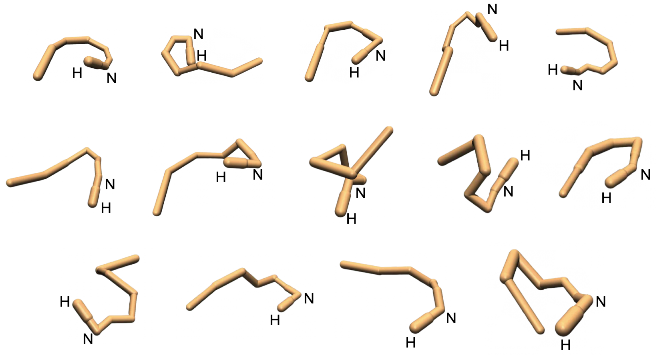

Fig. 3: Top fourteen generated robots in the last iteration of the toy experiment. All these designs have two initial links i.e. head(H) and neck(N), and six-links added by RoboNet, and have the highest possible length of 39.6 units.

VI. Experiments and Results

A. Environment

We use the OpenAI Ant-v5 Mujoco environment where we retain all the original physics settings. However, instead of using the default Ant robot, we allow our algorithm to construct robots of its choice using monolithic geometries called capsules. The maximum number of capsules RoboNet can use is limited based on the type of terrain the agent is tested in. The goal of these candidate-design robots is to move forward and collect rewards while ensuring they do not lose balance and hit their head on the ground, upon which the episode terminates. We also modified the environment reward for staying alive to 0.5 instead of 1.0, and there is no control cost associated with joint movements. This expands RoboNet's action space to 625 dimensions, incorporating 5 different options for link-cost (to employ the cost-aware sampling strategy), as well as 5 distinct choices for length, size, and gear ratio.

B. Baselines

We evaluate our method against the following baselines.

1) Transform2Act: Transform2act uses graph-based representation for agents with limbs represented as edges and joints as nodes in a 3-stage policy optimization - first for skeleton design, second for parameter learning, and then for learning control policy. In our implementation, we present their algorithm with the same problem of the high-dimensional and open design space RoboNet solves.

2) GFlowNet: A naïve GFlowNet implementation, applied directly to solve the robot design problem without the rate based design prioritization and cost-awareness components.

We provide all baselines and our method a total timesteps budget of 5 billion steps to Co-Design.

C. Results

We set up 2 different types of terrain to walk along, these include a flat and a steep 15-degree inclined walking surfaces and are further described below:

- Flat Terrain: Our flat terrain is the standard Ant-v5 environment terrain with all physics conserved as the original Ant. RoboNet is allotted a maximum of 8 nodes to build with and the training budget per candidate design is limited to 450k timesteps.
- Inclined Terrain: Our inclined Terrain has similar dynamics to the Ant-v5 environment, except that the slope of the floor surface is tilted 15° to create a more challenging walking task for the generated robots. RoboNet is allotted a maximum of 10 nodes and the training budget is limited to 600k timesteps per candidate design.

	Flat Terrain	Inclined Terrain
Transform2Act [19]	416.1 ± 46.2	130.8 ± 37.3
GFlowNet [1]	590.3 ± 35.3	361.7 ± 80.8
RoboNet (Ours)	**925.3 ± 59.1**	**812.5 ± 71.8**

TABLE I: Average Rewards (as mean ± standard error) for the best-performing design after 400 GFlowNet iterations equivalent samples from each of the three methods across varying terrains when run for 5 independent trials and a training budget of 5 million timesteps is provided.

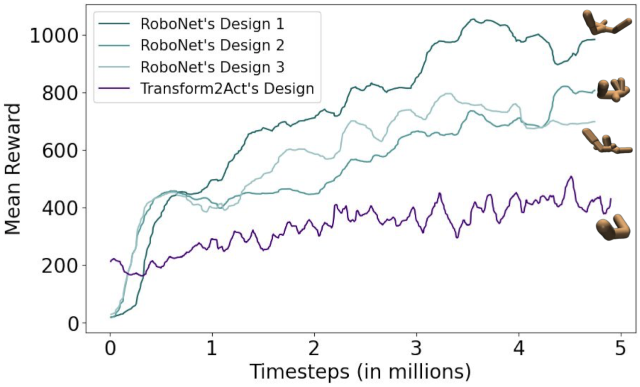

Fig. 4: The training curves of RoboNet's top 3 designs compared to Transform2Act's design when each robot design is independently trained for 5 million timesteps in the flat terrain. Similar behavior is also observed in the inclined terrain.

The results in Table I further prove the sample efficiency of our method. Since Transform2Act sequentially upgrades an initial design and focuses on training these intermediary designs to maximize task performance, their final design might still be too simple to handle difficult terrains similar to inclined terrain. RoboNet and GFlowNet instead focus on a design-first approach where more resources are allocated on identifying the best design rather than training each design to completion, thus saving up on a lot of samples. Though GFlowNet is better at handling samples compared to Transform2Act, it still struggles with distinguishing capable candidates amongst designs. This is because it relies only on the performance value but not the pace of learning, and does not take into account that more complex designs might need more resources than a simple design, like RoboNet does. Additionally, while Tranform2Act outputs a single design, RoboNet outputs a distribution of designs (as presented in Fig. 4).

VII. Discussion

RoboNet represents a significant step forward in the field of robot co-design, leveraging the power of GFlowNets to explore the vast space of possible robot configurations more efficiently. However, as with any novel approach, RoboNet is a work in progress with several challenges that need to be addressed to fully unlock its potential in robotics applications.

A. Current Limitations

- Computational Complexity: While RoboNet demonstrates improved sample efficiency compared to traditional methods, the computational cost of evaluating complex robot designs remains a significant bottleneck. As the dimensionality of the design space increases, so does the computational burden of simulating and evaluating each candidate design.
- Exploration vs Exploitation: Ensuring sufficient exploration in a vast and heavily constrained robot design space is a huge challenge. This is because while good levels of exploration, in theory, have to be encouraged, the cost of evaluation prohibits this to a large extent. We only trained RoboNet with a batch size of 32 resulting in RoboNet sampling only close to 10,000 robot designs to deliver the results discussed, however, we believe exploring the design space more with the possibility of adding more robot-links will unlock truly unique design structures across multiple tasks.
- Knowledge Propagation: While several robot candidate designs are being evaluated, there is no knowledge accumulation currently in place that could help reduce the need for extensive evaluations. This is challenging because of the number of nodes each of these candidate design robot graphs may contain.

VIII. Conclusion

This paper introduces RoboNet, a novel sample-efficient framework for robot co-design that leverages Generative Flow Networks (GFlowNets) to explore the vast space of robot configurations. RoboNet's key innovations include rate-based design candidate prioritization, cost-aware sampling, and adaptive annealed exploration. These strategies enable RoboNet to balance performance gains against computational costs while gradually shifting from broad exploration to focused exploitation. Our experimental results demonstrate RoboNet's superior performance compared to baseline methods such as Transform2Act and vanilla GFlowNet in both flat and inclined terrain scenarios. Despite these promising results, challenges remain in reducing computational complexity and refining the balance between exploration and exploitation. Future work should address these limitations and explore RoboNet's application to more diverse and challenging robotic tasks.

References

[1] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

[2] Josh Bongard, Victor Zykov, and Hod Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

[3] Flaviu Cipcigan, Jonathan Booth, Rodrigo Neumann Barros Ferreira, Carine Ribeiro dos Santos, and Mathias Steiner. Discovery of novel reticular materials for carbon dioxide capture using gflownets. *Digital Discovery*, 3(3): 449–455, 2024.

[4] Heng Dong, Junyu Zhang, Tonghan Wang, and Chongjie Zhang. Symmetry-aware robot design with structured subgroups. In *International Conference on Machine Learning*, pages 8334–8355. PMLR, 2023.

[5] David Ha. Reinforcement learning for improving agent design. *Artificial life*, 25(4):352–365, 2019.

[6] David Howard, Agoston E Eiben, Danielle Frances Kennedy, Jean-Baptiste Mouret, Philip Valencia, and Dave Winkler. Evolving embodied intelligence from materials to machines. *Nature Machine Intelligence*, 1 (1):12–19, 2019.

[7] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[8] Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. Biological sequence design with gflownets. In *International Conference on Machine Learning*, pages 9786–9801. PMLR, 2022.

[9] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

[10] Hod Lipson and Jordan B Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799): 974–978, 2000.

[11] Kevin Sebastian Luck, Heni Ben Amor, and Roberto Calandra. Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning. In *Conference on Robot Learning*, pages 854–869. PMLR, 2020.

[12] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022.

[13] Chandana Paul, Francisco J Valero-Cuevas, and Hod

Lipson. Design and control of tensegrity robots for locomotion. *IEEE Transactions on Robotics*, 22(5), 2006.

[14] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *2017 ieee international conference on robotics and automation (icra)*, pages 1527–1533. IEEE, 2017.

[15] Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R Walter. Jointly learning to construct and control agents using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9798–9805. IEEE, 2019.

[16] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[17] Miguel G Villarreal-Cervantes, Carlos A Cruz-Villar, Jaime Alvarez-Gallegos, and Edgar A Portilla-Flores. Robust structure-control design approach for mechatronic systems. *IEEE/ASME Transactions on Mechatronics*, 18 (5):1592–1601, 2012.

[18] Wenhao Yu, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. Learning fast adaptation with meta strategy optimization. *IEEE Robotics and Automation Letters*, 5 (2), 2020.

[19] Ye Yuan, Yuda Song, Zhengyi Luo, Wen Sun, and Kris M Kitani. Transform2act: Learning a transform-and-control policy for efficient agent design. In *International Conference on Learning Representations*, 2021.