COGENT: Co-design of Robots with GFlowNets

Kishan Reddy Nagiredla
Applied Artificial Intelligence Initiative
Deakin University
Australia
knagiredla@deakin.edu.au

Arun Kumar Anjanapura Venkatesh
Applied Artificial Intelligence Initiative
Deakin University
Australia

Thommen George Karimpanal
Schoool of Information Technology
Deakin University
Australia

Kevin Sebastian Luck
Computational Intelligence Group
Vrije Universiteit Amsterdam
Netherlands

Santu Rana
Applied Artificial Intelligence Initiative
Deakin University
Australia

Abstract

Co-design of robots involves optimizing 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. We introduce COGENT, a novel framework that leverages a graph synthesis technique named GFlowNet, to enhance search space traversal in robotic co-design. To increase sample efficiency, the proposed framework introduces a cost/performance-aware design prioritization mechanism that learns a design generator policy by carefully sampling the design space. Our experiments show the effectiveness of the proposed framework in various robot co-design tasks. Evaluations performed on a wide range of agent design problems demonstrate that our method significantly outperforms baselines. We show that COGENT produces a suite of diverse designs achieving better task objectives across all evaluated design problems.

1 Introduction

Designing robots involves numerous challenges, from defining the robot's structure to integrating various components and developing control algorithms. Hence, the development of robot platforms has mainly been a task of human engineers thus far. However, the development of optimization- and learning-driven methods capable of co-designing robot behavior and embodiment has received increased attention in recent years. Early approaches, starting with the seminal work by Sims (1994) to the more recent works like Michalewicz (2013), Doncieux et al. (2015), Lipson and Pollack (2000), Gupta et al. (2021) address the problem of robot co-design primarily through the lens of (biologically-inspired) evolution. These evolutionary methods apply the principles of selection, variation, and inheritance to automate the robot creation process, enabling candidate robot designs to independently learn and adapt by engaging with their surroundings similar to living organisms.

Recent advances in computational efficiency have propelled deep learning-based methods such as Wang et al. (2018), Ha (2019) and Luck et al. (2020), demonstrating early success in co-optimizing robot design and control by leveraging (deep) reinforcement learning (RL). This resulted in more recent advances including Yuan et al. (2021), Dong et al. (2023), Fan et al. (2024), and Lu et al. (2025), all focusing on the generation of novel task-tailored robotic systems through RL traversing the underlying complex search space of possible robot embodiments. More specifically, these approaches model each robot as a graph comprised of discrete components (links & joints, with actuators at each joint) and frame the co-design problem as a joint optimization over a combinatorial graph space and continuous closed-loop control policies, constrained usually by a finite evaluation budget. However, all methods discussed thus far focus on producing a single "best" robot design that is incrementally transformed given an initial template. We hypothesize that a major pitfall to such incremental design alterations is sub-optimal convergence in the highly multimodal, rugged design landscapes typical of robot co-design. This focus on single, incrementally derived designs highlights two fundamental challenges for co-design algorithms:

(C1) Generating a single optimal design is insufficient for real-world applications requiring diverse alternatives to evaluate multi-objective performance trade-offs. Efficiently exploring vast design spaces for multiple high-performing candidates remains an open challenge.

(C2) Determining candidate fitness is difficult under limited computational or real-world resources, as training behavior policies for each varying prototype is resource-intensive. This is compounded by designs of differing complexity requiring disparate training budgets.

A promising framework for generating diverse, high-quality graphs to address **C1** is the Generative Flow Network (GFlowNet) (Bengio et al., 2021). GFlowNets learn a probability distribution over graph structures by modeling the generation process as a flow network, where paths to high-reward terminal states receive higher probability. This allows for efficient sampling of diverse, high-reward graphs, distinct from single-optimum methods, and has seen success in areas like drug discovery (Jain et al., 2022) and material design (Cipcigan et al., 2024). However, applying GFlowNets to robot co-design is significantly challenged by **C2**, due to the computational cost of evaluating the fitness of each potential design candidate.

In this paper, we introduce **Co**-design of Robots with **GE**nerative Flow **NeT**works (**COGENT**), a novel GFlowNet-based co-design framework. COGENT addresses the limitations highlighted by **C2** through two key innovations: (i) a performance rate-based prioritization method implementing an early stopping mechanism based on current performance and its improvement rate during training, and (ii) a cost-aware design sampling method that optimizes training budget allocation by learning to prioritize resources for promising designs. We evaluate COGENT in MuJoCo (Todorov et al., 2012) and EvolutionGym (Bhatia et al., 2021) environments to demonstrate its effectiveness in generating diverse, task-specific robot co-designs, outperforming existing methods within a constrained computational budget. Notably, COGENT consistently generated multiple high-performing robot designs exhibiting distinct structural characteristics.

2 Related Works

Robot co-design, which aims to simultaneously optimize a robot's physical structure (embodiment) and its control policy (behavior), has roots in several distinct research areas.

Evolutionary Co-design. Pioneering work of Sims (1994) demonstrated the potential to evolve both morphology and controllers for virtual creatures in simulated physics environments. This spurred further research exploring diverse representations and evolutionary algorithms for co-design including Stanley and Miikkulainen (2002); Michalewicz (2013); Chu et al. (2011); Michalewicz (2013); Doncieux et al. (2015); Wang et al. (2019); Gupta et al. (2021). These methods generate increasingly complex robots through effective traversal of the complex search space through iterative mutation and selection, aiming to find a single optimal design for a given task. However, premature convergence to locally optimal solutions (Gupta et al., 2021) is a deterrent to employing these methods in complex search spaces. There is also significant work in quality-diversity optimization methods, for example MAP-Elites (Mouret and Clune, 2015) to discover solutions that not only perform well, but also exhibit diverse behavioral characteristics. However, these methods need a good descriptor to guide their diversification and are not easily scalable to the more complex setting of joint optimization of morphology and control, where there is a possibility of wasting a lot of evaluations when poorly initialized. Additionally, the effectiveness of early stopping as a means to improve sample efficiency while preserving behavioral diversity has been demonstrated by Veenstra and Glette (2020) and further generalized by Arza et al. (2024). However, as noted in their work, the proposed stopping criteria offer limited advantages over the default termination mechanisms already implemented in most simulators, which are designed to prevent prolonged execution in undesirable states.

Deep-RL based Co-design. Recent advances in differentiable physics and scalable RL have inspired methods that treat morphology choice as part of the decision process. Approaches like Wang et al. (2018) and Ha (2019) demonstrated how RL could be used to co-optimize modular robot designs, often represented as graphs, alongside their controllers. Subsequent works were built on this approach, by transforming initial robot structures through expressive graph encodings (Yuan et al., 2021), incorporating symmetry priors (Dong et al., 2023), incorporating memory to reuse good designs to balance exploration with exploitation (Fan et al., 2024), developing specific graph-based RL frameworks for co-design (Luck et al., 2020) and introducing global message passing and reward-balancing (Lu et al., 2025). Despite their differences, these algorithms still return a single "best" design after an incremental hill-climbing procedure, making them prone to local convergence in a complex design landscape.

GFlowNets for Design. Generative Flow Networks (GFlowNets) (Bengio et al., 2021) have been proposed as a promising solution to generate diverse candidates in complex design spaces. They learn a policy to construct objects sequentially with probabilities proportional to a given reward function for a more global exploration of the design space. Initially applied to molecular design (Jain et al., 2022) and material discovery (Cipcigan et al., 2024), GFlowNets show potential for robot co-design, but are hindered by challenges in sample efficiency and the exploration-exploitation balance.

To the best of our knowledge, COGENT is the first method to generate multiple high-performance robot designs through the combination of a novel early-stopping mechanism and an intelligent resource-allocated mechanism to train RL-based controllers. Through this, COGENT explicitly learns a policy to generate heterogeneous, high-reward morphologies addressing the first challenge (C1) and introduces a performance rate-based prioritization and a cost-aware sampling strategy to address the evaluation-cost bottleneck C2 overlooked by previous approaches.

3 Preliminaries

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, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ 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 $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the scalar-valued reward-producing function for taking action a in state s.

The learning problem is to find an optimal policy that maximizes returns $\mathbb{E}_{\tau \sim \pi}[r(\tau)]$, where τ is a trajectory sampled from policy π, and $r(\tau) = \sum_{(s_i, a_i) \in \tau} R(s, a)\gamma^i$ is the shorthand for the sum of the rewards over trajectory τ with a discount factor $\gamma \in [0, 1]$.

Generative Flow Networks.

GFlowNets (Bengio et al., 2021) learn a stochastic, step-wise construction policy that samples terminal objects $\bar{s}_T$ with probability proportional to a reward $\bar{R}(\bar{s}_T)$:

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

By explicitly spreading probability mass over many high-reward states, GFlowNet produces diverse candidates. GFlowNets have already been effective in settings that demand both quality and variety, such as novel drug candidates (Jain et al., 2022) and material discovery (Cipcigan et al., 2024).
MDP formulation Starting from the initial (empty) state $\bar{s}_0$, a policy $\bar{\pi}_\theta(\bar{a}|\bar{s})$ either stops or attaches a component, yielding a trajectory $\bar{\tau}$ = ($\bar{s}_0$, $\bar{a}_0$, .. , $\bar{s}_T$). Each state $\bar{s}$ carries a non-negative *flow* $F(\bar{s})$ that satisfies the conservation rule:

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

where $P(\bar{s}' \mid \bar{s}, \bar{a})$ is the forward transition probability. When this condition holds, the induced terminal-state distribution matches the target in Eq. (1) while allocating flow across multiple construction paths. This ensures: 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 in each state to ensure that it is consistent with both the reward and flow conservation property. In this way, the model learns a probability distribution over the trajectories that encourages diverse, high-reward graphs.

To ensure that the sampling model learns a policy that can generate samples according to a target distribution, GFlowNets uses the Trajectory Balance (TB) objective (Malkin et al., 2022), given by:

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

where θ, ψ denote the learnable parameters and Z_ψ is the partition function that ensures consistency between state transitions. TB aims to ensure that the marginal likelihood of a trajectory becomes proportional to the reward $\bar{R}(\bar{s}_T)$ through efficient credit assignment.

Application to Co-design. We hypothesize that GFlowNet can generate diverse designs, and avoid *mode collapse* Bengio et al. (2021) observed when applying standard RL approaches, leveraging a carefully crafted reward function $\bar{R}(\bar{s}_T)$ that can reflect task performance, and balance exploration-exploitation through flow consistency and probabilistic diversity in high-dimensional spaces.

4 Co-Design Problem Statement

We follow the general co-design setup (Luck et al., 2020; Yuan et al., 2021) and define the optimization of robot embodiment and behavior as a bi-level optimization problem (Bracken and McGill, 1973), (Liu et al., 2021). Given a set of robot embodiments in their graph representation $\hat{\mathbf{G}}(V, E)$ and set of policy/behavior parameters $\mathbf{\Phi}$ we aim to solve:

$$\max_{\mathbf{G}\in\hat{\mathbf{G}}} \bar{R}(\mathbf{G}, \phi^*), \text{ s.t. } \phi^* = \arg\max_{\phi\in\Phi} R(\mathbf{G}, \phi), \tag{4}$$

where $\bar{R}(\cdot,\cdot)$ and $R(\cdot,\cdot)$ are fitness functions. The outer loop optimizes robot embodiment design, while the inner loop finds the optimal behavioral policy π_ϕ for a given design. We further specify that both optimization problems are posed as Markovian, and thus can be modeled as an MDP, and solved via (deep) RL.

The outer optimization problem models the robot embodiment design search as a sequential graph construction process represented as an MDP and presented as $\bar{\pi}_\theta$ with a maximization objective:

$$\mathbb{E}\left[\sum_{t=0}^{T} \bar{\gamma}^t \bar{r}(\bar{s}_i, \bar{a}_i) \mid \bar{a} \sim \bar{\pi}_\theta(\bar{s}_i), \bar{s}_i = \bar{d}(\bar{a}_{i-1}, \bar{s}_{i-1})\right], \tag{5}$$

in which the state $\bar{s}_i$ corresponds to the graph $\mathbf{G}_i$ constructed up to step i. Actions $\bar{a}_i$ involve adding components (like nodes or edges) to the graph, governed by deterministic dynamics $\bar{s}_i = \bar{d}(\bar{a}_{i-1}, \bar{s}_{i-1})$. The final graph $\mathbf{G}_T$ generated upon reaching a terminal state $\bar{s}_T$ (e.g., via a STOP action) serves as the static structural description of a robot, convertible to formats like the Universal Robot Description File (URDF) format (Kang et al., 2019). This relation precisely defines how the graph representation is transformed from $\mathbf{G}_{i-1}$ to $\mathbf{G}_i$ under the influence of the action $\bar{a}_{i-1}$. Within this MDP framework, we employ GFlowNets (Sec. "Preliminaries") to learn a stochastic policy $\bar{\pi}_\theta$ for graph construction. GFlowNets sample terminal graphs $\mathbf{G}_T$ with probability proportional to a non-negative terminal reward $\bar{R}(\mathbf{G}_T)$. The outer loop is thus optimized using only this terminal reward, with a discount factor $\bar{\gamma} = 1$, aligning with the GFlowNet objective of matching the terminal reward distribution. Moreover, in this bi-level framework, this terminal reward for the outer loop, $\bar{R}(\mathbf{G}_T)$, is defined by the embodiment fitness obtained from the inner policy optimization given by $\bar{R}(\mathbf{G}_T) = \max_\phi R(\mathbf{G}_T, \phi)$. This fitness is determined by training a policy for the constructed embodiment $\mathbf{G}_T$ using classic (deep) RL (e.g., PPO (Schulman et al., 2017)) on the task-specific reward $R(\cdot,\cdot)$. In this inner loop MDP, the robot's dynamics $s_i \sim d(s_i|s_{i-1}, a_{i-1}, \mathbf{G})$ depend explicitly on the embodiment $\mathbf{G}$, and has the standard RL discount factor $\gamma < 1$, contrasting with the outer loop's $\bar{\gamma} = 1$.

Crucially, evaluating $\bar{R}(\mathbf{G})$ for each embodiment requires executing the computationally expensive inner policy optimization. Given the vastness of $\hat{\mathbf{G}}$ and a finite computational budget, the practical challenge is to efficiently explore the graph space $\hat{\mathbf{G}}$ and discover a diverse set of high-performing designs $\{\mathbf{G}_1, \ldots, \mathbf{G}_N\}$ and their corresponding policies, while simultaneously addressing the limitations outlined as challenges (**C1**) and (**C2**) in Sec. "Introduction".

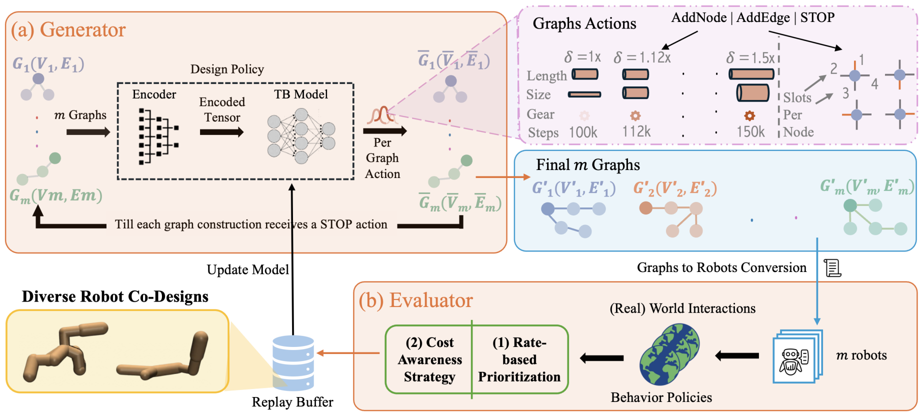

Figure 1: An illustration of COGENT framework with the two key stages: (a) **Generator** generates final graphs, by encoding graph objects into tensors to pass through the TB Model to compute graph actions i.e. AddNode, AddEdge or STOP (top-right purple box) until STOP action is received and (b) **Evaluator**, evaluates the robots constructed from final graphs and applies performance rate-based prioritization and cost-awareness techniques before storing these ranked co-designs in the replay buffer; which then helps calculate model loss and updates model parameters.

5 COGENT: End-to-End Architecture

This section outlines the key components in our approach coined COGENT, to traverse the complex robot co-design space. This implementation can be broadly divided into two main stages, as depicted in the flow diagram in Fig. 1:

Generator. The Generator (a), depicted in Fig. 1, is responsible for proposing a batch of m diverse robot designs, each represented as a graph $\mathbf{G}$. This is achieved using a GFlowNet Transformer with a design policy $\bar{\pi}_\theta$. Starting from initial states, the generator incrementally constructs each of the m graphs by iteratively applying Graph Actions (AddNode, AddEdge, STOP) selected from a defined action space. These actions are guided by the learned forward policy $\bar{\pi}_\theta$, which is influenced by our annealing exploration strategy (SubSec. “Annealing Exploration”). The process continues independently for each graph in the batch until a STOP action is sampled, yielding m complete robot design graphs ($\bar{s}_{T_1} = \mathbf{G}'_1$ to $\bar{s}_{T_m} = \mathbf{G}'_m$). To promote diversity within the generated batch, the Gumbel trick (Gumbel, 1954) is applied when sampling actions.

Evaluator. The Evaluator (b) (Fig. 1) takes the batch of m generated robot design graphs ($\mathbf{G}'_1$ to $\mathbf{G}'_m$) and assesses their potential task performance. Firstly, these graph representations are converted into a format suitable for simulation, such as XML, representing the robot's structure. These robots are then instantiated within a physics simulator, where their behavior policies are trained using an policy optimization algorithm (like PPO) for a specified number of timesteps (ζ_i for each design d_i). The performance achieved during this training ($R(\bar{s}_{T_i})$) is then processed by our proposed methods: Performance Rate-based Prioritization and Cost-Aware Sampling. These methods modify the raw simulation reward to produce a composite reward $\bar{R}(\bar{s}_{T_i})$ (Eq. 8) that incorporates factors beyond just final performance, reflecting the potential for improvement and the computational cost of evaluation. This composite reward is then used to update the Generator's design policy $\bar{\pi}_\theta$ and the partition function Z_ψ via the TB objective (Eq. 3), closing the loop.

5.1 COGENT Contributions

To tackle the challenges of sample inefficiency and evaluating diverse designs in robot co-design with GFlowNets, COGENT introduces key contributions in both the Generator and Evaluator stages. The goal remains to learn a design policy $\bar{\pi}_\theta$ that generates high-performing robot designs $\bar{s}_T$, optimized via the TB objective (Eq. 3). The following subsections outline these contributions:

Rate-based Prioritization. To address sample inefficiency of standard GFlowNets, we introduce performance rate-based prioritization (RBP), an early stopping heuristic that is part of the Evaluator (Fig. 1 (b)). In the context of robot co-design, this rate indicates how quickly the robot's performance improves during training with a sampled budget of ζ timesteps immediately before the cutoff budget, denoted by η. A higher rate combined with good reward accumulation performance indicates that the robot co-design has a comparatively greater potential for rapid improvement given more training timesteps. Therefore, in COGENT, we prioritize such quick and stable learners by ranking them higher relative to the other co-designs. This can be formalized as:

$$\hat{R}_p(\bar{s}_T) = \hat{R}(\bar{s}_{T_\zeta}) + \frac{\hat{R}(\bar{s}_{T_\zeta}) - \hat{R}(\bar{s}_{T_\eta)}}{\zeta - \eta} \tag{6}$$

where $\hat{R}(\bar{s}_{T_\zeta})$ is the performance after a budget of ζ timesteps and $\hat{R}(\bar{s}_{T_\eta})$ is the performance observed at an intermediate budget of η timesteps.

Cost Aware Sampling. Allocating a fixed amount of training resources to all candidate co-designs can lead to two issues: (i) If the allocated timestep budget is too low, the Generator (Fig. 1 (a)) may become biased towards producing simpler designs that underperform when given more resources, due to learning behaviors that are only beneficial in the short term. (ii) If the allocated budget is too high, unpromising designs receive excessive resources, resulting in sample inefficiency and potential difficulties in convergence. It is challenging to pin down the right level of resources for fair evaluation per task. Hence, we propose a flexible budgeting mechanism wherein designs may utilize evaluation resources exceeding a base level, contingent on performance gains justifying the imposed resource cost penalty. COGENT learns a joint sampling strategy to select both design nodes (robot links) with parameters (length, size etc.) and an associated budget factor δ for each node. The evaluation budget per node is calculated as $b_{v\in V} = \delta \cdot b_0$, where b_0 is the base budget allocated for the selected node and δ is a multiplicative factor learned by the design policy. After all the nodes are sampled with their individual node resource (b_v), the total budget (ζ timesteps) is calculated as an aggregate of individual node budgets. However, allocating maximum budget to each node i.e., $b_v = \max(\delta) \cdot b_o$ leads to an undesirably large total budget, thus we restrict such allocations by introducing a cost metric $C(\bar{s}_T)$ that quantifies the excess cost of the robot design relative to a baseline budget (across all the nodes V in graph $\mathbf{G}$). Therefore, cost-aware sampling (CS) can be formalized as:

Algorithm 1 COGENT Algorithm

```
Initialize: Design Policy $\bar{\pi}_\theta$, Behavior Policy $\pi_\theta$, Partition function $Z_\psi$, Batch Size $m$, Maximum Iterations $K$, and Maximum Training Budget $\mathcal{B}$    ▷ Design Generator Loop
while iter $< K$ do
    Sample batch $\bar{\tau}_i \ \forall i \in \{1, \cdots, m\}$ using $\bar{\pi}_\theta$ and $\beta$ (Eq. 9)
                                            ▷ Behavior Evaluator Loop
    for $\bar{\tau}_i = (\text{design } d_i, \text{budget } \zeta_i, \text{reward } r_i)$ in batch $i$ do
        Train $\pi_\phi$ of $d_i$ for $\zeta_i$ steps and collect reward, $R(\bar{s}_{T_i})$.
        Update Reward using Eq. 8 to get $\bar{R}(\bar{s}_{T_i})$
    end for
    Calculate loss $\mathcal{L}(\bar{\tau}_i)$ using TB objective (Eq. 3)
    Update $\theta$ and $\psi$ via gradient descent on $\mathcal{L}(\bar{\tau}_i)$ using Eq. 3
end while
Sample final batch $\bar{\tau}_f$ using $\bar{\pi}_\theta$
Train all designs in final batch using maximum budget $\mathcal{B}$
Result $\bar{\pi}_\theta$ and final designs from $\bar{\tau}_f$
```

$$\hat{R}_c(\bar{s}_T) = w_c C(\bar{s}_T) \tag{7}$$

where w_c is a hyperparameter that controls the trade-off between task performance and evaluation cost. Mathematically,

$$C(\bar{s}_T) = \zeta - |V| \times b_0 \quad \text{and} \quad \zeta = \sum_{v\in V} b_v$$

Finally, the combination of $\hat{R}(\bar{s}_T)$, $\hat{R}_p(\bar{s}_T)$ and $\hat{R}_c(\bar{s}_T)$ constitutes $R(s_T)$, that is,

$$\bar{R}(s_T) = \hat{R}(\bar{s}_T) + \hat{R}_p(\bar{s}_T) + \hat{R}_c(\bar{s}_T) \tag{8}$$

Annealing Exploration. Efficiently exploring the vast robot design space in the early stages of training and then converging on high-performing regions is crucial for effective co-design. In GFlowNets, the inverse temperature parameter β in the forward policy $\bar{\pi}_\theta$ controls this trade-off (Bengio et al., 2021), where higher β promotes uniform sampling (exploration), while lower β sharpens the distribution toward high-reward actions (exploitation). Standard implementations often fix β or apply basic annealing schedules. To smoothen this transition, COGENT introduces an annealing strategy given by:

$$\beta_{decay}(t) = \rho \cdot \beta_{init} + (1 - \rho \cdot \beta_{init} \cdot (1 - d_v)^{(t-d_d)/d_s} \tag{9}$$

where,

$$\rho(d_d, t) = \begin{cases} 0 & \text{if } t \geq d_d \\ 1 & \text{if } t < d_d \end{cases} \tag{10}$$

wherein t is the current iteration, d_v is the percentage reduction in exploration every d_s number of iterations and d_d determines the window after which this exploration annealing begins. The gradual decay in β ensures COGENT's Generator smoothly transitions its design policy from exploration to exploitation, avoiding abrupt changes in the sampling distribution that could destabilize training.

In summary (also presented in Alg. 5.1), COGENT addresses key challenges in applying GFlowNets to robot co-design through several novel contributions. The Generator's (Fig. 1 (a)) exploration is strategically managed by the annealing exploration strategy (SubSec. "Annealing Exploration"), while the Evaluator's (Fig. 1 (b)) efficiency is significantly improved by incorporating rate-based prioritization (Eq. 6) and cost-aware sampling (Eq. 7) into the reward computation. By enhancing both the design proposal and evaluation phases, these combined strategies allow COGENT to overcome the sample inefficiency and evaluation complexity often faced by standard GFlowNets and traditional techniques in this domain, facilitating the discovery of effective robot co-designs.

6 Experiments and Results

6.1 Experiment Setup

We use the Gymnasium MuJoCo environments (Todorov et al., 2012) and EvolutionGym, (Bhatia et al., 2021) environments where we retain all original physics settings. These simulators allow us to embed agents with different embodiments proposed by COGENT and baseline methods. While COGENT can handle any number of robot links (or voxels), we limit the maximum number to create environments of different complexity as follows:

MuJoCo Environments. We use *3D Locomotion* (based on Ant-v5 (Schulman et al., 2015)), *2D Locomotion* (based on Hopper-v5 (Erez et al., 2012)), *Gap Environment* (our modified terrain based on Hopper-v5) and *Swimmer* (based on Swimmer-v5 (Coulom, 2002)) environments. The maximum nodes (or body parts) in each of these environments is limited to 10, 6, 8 and 6 respectively. Each of their body parts are cost coded to $1.5e4$ timesteps. A robot, for example, in the 3D Locomotion environment can by default be allocated a base total budget, i.e. $|\mathbf{V}(\mathbf{G})| \times b_0$ which can vary between $1.5e4$ to $1.5e5$ timesteps depending on the number of nodes being 1 to 10 respectively. The goal of all robots in these environments is to walk to collect more rewards.

EvolutionGym Environments. We use the 3 locomotion environments Walker, BridgeWalker and BidirectionalWalker introduced in EvoGym (Bhatia et al., 2021). We limit the maximum voxels for each robots generated across these environments to 16, and each robot across the mentioned environments are allocated a base budget that could vary between $6.25e3$ to $1e5$ timesteps depending on the voxels sampled in the final robot. The goal of these agents is also to move forward and collect more rewards when various terrain conditions are presented in each environment.

COGENT Hyperparameters. The hyperparameters corresponding to RBP and CS components are set to $b_0 = 2e4$, $\eta = 1.5e4$, $\delta = \{1, 1.12, 1.25, 1.37, 1.5\}$ and $w_p = 1$, $w_c = -0.002$ across all experiments. For annealing exploration, we set $\beta_{init} = 10$ $d_v = 0.5$, $d_s = 16$ and $d_d = 25$.

Table 1: **MuJoCo Envs.** Average Rewards (as mean $\pm$ standard deviation) for the best design generated by each of the methods in different MuJoCo environments when run for 5 independent trials and a training budget of $2e6$ timesteps. These best designs are selected after training all co-design methods to an equivalent timesteps budget of $1.5e8$.

MuJoCo Environments	3D Locomotion	2D Locomotion	Gap Terrain	Swimmer
ESS	823.7 ± 65.7	925.9 ± 55.2	569.3 ± 78.3	198.5 ± 49.1
NGE	1023.8 ± 82.7	1002.7 ± 58.8	656.2 ± 88.2	159.3 ± 35.1
T2ACT	1563.7 ± 46.2	1639.3 ± 47.3	1116.5 ± 142.7	190.8 ± 52.4
EDiSon	1890.3 ± 35.3	1923.8 ± 80.8	1592.5 ± 75.4	233.7 ± 80.6
BodyGen	2393.3 ± 35.3	2234.3 ± 28.4	1605.1 ± 32.1	242.5 ± 43.4
COGENT	**2830.3 ± 59.1**	**2612.7 ± 27.2**	**1862.3 ± 42.9**	**320.6 ± 12.5**

Table 2: **EvoGym Envs.** Average Rewards (as mean $\pm$ standard deviation) for the best design generated by each of the methods in different EvoGym environments when run for 5 independent trials and a training budget of $2e6$ timesteps. These best designs are selected after training all co-design methods to an equivalent timesteps budget of $1.5e8$.

EvoGym Environments	Walker	BridgeWalker	BidirectionalWalker
GA	3.8 ± 0.6	4.8 ± 0.5	4.2 ± 0.5
BO	2.1 ± 1.1	5.1 ± 0.8	4.8 ± 0.8
CPPN-NEAT	5.6 ± 0.2	3.3 ± 0.3	4.3 ± 0.4
COGENT	**7.3 ± 0.5**	**6.6 ± 0.4**	**7.1 ± 0.5**

Baselines. Across MuJoCo experiments we use two evolutionary methods i.e. ESS (Chu et al., 2011) and NGE (Wang et al., 2019) and three Deep-RL based Co-Design methods i.e. T2ACT (Yuan et al., 2021), EDiSon (Fan et al., 2024) and BodyGen (Lu et al., 2025) as baselines. Across EvolutionGym environments, we use GA (Michalewicz, 2013), BO (Kandasamy et al., 2018), and CPPN-NEAT (Stanley and Miikkulainen, 2002) as baselines. For a fair evaluation, we use the same initial robot design across all MuJoCo and EvoGym experiments for baselines and COGENT.

Evaluation Budget. We allocate all evolutionary methods and COGENT $1.5e8$ timesteps to train and collect 3 top designs. We compare these to the designs produced by 3 independent runs (different seeds) of all Deep-RL co-design methods trained for $5e7$ timesteps. To disentangle the quality of design from policy, we independently train the design(s) produced from each method using common PPO hyperparameters for $2e6$ timesteps and report the results for the designs produced by all methods.

6.2 Results

Sample Efficiency Analysis. We present our results in Table 1 comparing COGENT with both Deep-RL based policy gradient methods and evolutionary methods across MuJoCo environments. Since all baselines optimize a single fitness function that combines design and behavior effectiveness, their final robot design given the same amount of environment interaction budget as COGENT is still low performing when the influence of their learned behavior policy is removed. The experimental results show that COGENT is significantly $(p << 0.05)$ better than the next-best baseline for each environment confirmed using a student t-test between COGENT and next best baseline results. This effectiveness is because COGENT focuses on a design-first approach, where more resources are allocated to identify the best embodiment rather than training one behavior policy until saturation which might forgo the need for good-quality designs. Additionally, COGENT's Performance Rate-Based Prioritization and Cost-Aware Sampling methods combined with the Annealing Exploration technique, help ensure the Generator effectively uses timesteps compared to other methods, thus arriving at multiple best performance designs faster while learning from good and bad performing robot designs throughout the training process. This ability of generating/discovering novel heterogeneous embodiments and locomotion behaviors, allows engineers to select from multiple top-performing designs rather than just one, given any task. Notably, the COGENT's designs conform to the established constraints of evaluation budget, design, and behavior trade-offs yet perform better than the baselines, evident from Fig. 2.

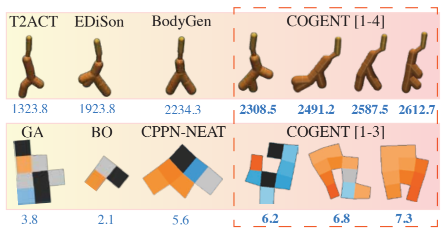

Figure 2: COGENT's diverse designs and single best design output by each baseline methods in 2D Loco. (top) and EvoGym Walker (bottom) envs. with performance values (in blue).

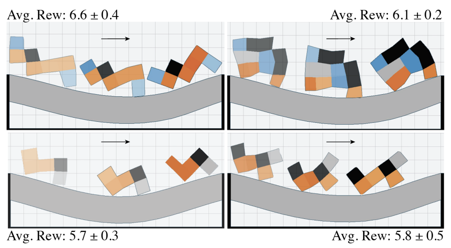

Figure 3: Top-4 diverse designs produced by COGENT in the BridgeWalker environment, moving left to right to traverse the suspended bridge.

We also observed that with the same COGENT hyperparameters, our contributions are applicable to tasks from a different simulator altogether. More specifically, across all EvoGym locomotion tasks, COGENT was able to generate compellingly diverse designs performing better than baselines. Their performance compared to baselines is presented in Table 2 and final designs are presented in Fig. 3.

Design Diversity Analysis. We quantified morphological diversity using Average Pairwise Distance (APD). Each robot was characterized by its number of components (voxels or links) and the set of unique component types (e.g., voxel types in EvoGym or parameter types in MuJoCo). The pairwise distance between two robots was calculated by averaging two values: the difference in their component counts and the Jaccard distance between their sets of component types (i.e., 1 minus the ratio of shared to total unique types). APD is defined as the average of these pairwise distances across all robot pairs and serves as a measure of overall structural diversity. Table 3 reports the APD results for both EvoGym and MuJoCo environments.

Table 3: APD values across 2 MuJoCo (top) and 2 EvoGym environments (bottom) compared to next best relevant baselines. A higher APD suggests more diversity in the final batch of robots and lowest APD possible is 0.

Environments	COGENT	BodyGen	CPPN-NEAT
3D Loco.	**2.1**	0.2	-
Gap	**0.3**	0.2	-
Walker	**2.5**	-	1.2
BridgeWalker	**3.2**	-	1.4

Table 4: Sensitivity analysis of COGENT in 2D Loco. environment, showing diversity and average performances of top-3 designs for different parameter settings of w_p and w_c.

Parameters	Diversity	Performance		
w_p, w_c	(in APD)	Top-1	Top-2	Top-3
0.5, -0.001	1.1	2670.7	2635.4	2608.2
0.5, -0.004	1.2	2877.6	2780.0	2716.5
2, -0.001	1.2	2787.4	2707.8	2666.3
2, -0.004	1.2	2552.8	2518.8	2504.8

Table 3 highlights the design diversity achieved by COGENT across environments of varying complexity, from the simpler Gap/Walker to the more complex 3D Locomotion/BridgeWalker environments. COGENT demonstrates a strong capacity to explore and generate diverse solutions in high-dimensional design spaces. However, in baseline methods such as BodyGen and CPPN-NEAT, the optimization is centered around improving a single design trajectory, thereby restricting their ability to capture a diverse range of viable morphologies.

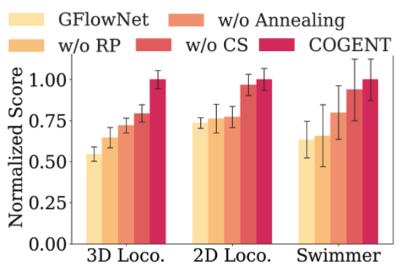

Figure 4: Normalized performance per million steps of top-performing designs by COGENT and its ablated versions.

6.3 Ablation Study

We performed ablations to show the impact of the individual components of the overall framework, Fig. 4). We compare COGENT with naive GFlowNet, its variants without Performance Rate-based Prioritization (w/o RP), Cost-aware Sampling (w/o CS) and Annealing Exploration (w/o Annealing). RP showed the highest significance on the performance of COGENT with CS and Annealing contributing significantly in the more complex 3D environment.

Sensitivity Analysis. Reward function in Eq. 8 is composed of three parts: (i) the task reward, (ii) R_p (proposed rate-based prioritization to detect good designs early), and (iii) R_c (proposed cost based penalization to give higher training budget *only* to deserving designs). We set the weight corresponding to (i) as 1 and vary the others ($w_p \in [0.5, 2.0]$ and $w_c \in [-0.001, -0.004]$) corresponding to R_p and R_c respectively and present these results in Table 4. Notably, across all parameter sets the performance of our co-designs is higher than the next best baseline which has a performance value of 2234.3.

7 Discussion & Conclusion

COGENT advances robot co-design by leveraging GFlowNets for sample-efficient, multi-modal distribution learning. Unlike traditional single-solution optimizers, COGENT generates diverse, high-performing designs, reducing sensitivity to initialization and enhancing robustness. Future work could improve further on sample efficiency by enabling knowledge transfer across robot topologies during policy (behavior) learning, transferring knowledge while preventing negative interference or catastrophic forgetting could significantly reduce the computational cost of evaluating candidate designs, thus enhancing scalability.

We presented COGENT, a novel framework for sample-efficient robot co-design that uses GFlowNets to explore diverse morphological and control configurations. Our key contributions include performance rate-based candidate prioritization, cost-aware sampling, and annealing exploration. These strategies enable COGENT to balance performance gains against computational costs, while gradually shifting from broad exploration to focused exploitation making the whole process extremely sample efficient. Our experimental results demonstrate COGENT's superior performance in diverse design generation compared to baseline methods in different environments.

References

Arza, E., Le Goff, L. K., and Hart, E. (2024). Generalized early stopping in evolutionary direct policy search. *ACM Transactions on Evolutionary Learning*, 4(3):1–28.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. (2021). Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394.

Bhatia, J., Jackson, H., Tian, Y., Xu, J., and Matusik, W. (2021). Evolution gym: A large-scale benchmark for evolving soft robots. *Advances in Neural Information Processing Systems*.

Bracken, J. and McGill, J. T. (1973). Mathematical programs with optimization problems in the constraints. *Operations research*, 21(1):37–44.

Chu, W., Gao, X., and Sorooshian, S. (2011). A new evolutionary search strategy for global optimization of high-dimensional problems. *Information Sciences*, 181(22):4909–4927.

Cipcigan, F., Booth, J., Ferreira, R. N. B., dos Santos, C. R., and Steiner, M. (2024). Discovery of novel reticular materials for carbon dioxide capture using gflownets. *Digital Discovery*, 3(3):449–455.

Coulom, R. (2002). *Reinforcement learning using neural networks, with applications to motor control*. PhD thesis, Institut National Polytechnique de Grenoble-INPG.

Doncieux, S., Bredeche, N., Mouret, J.-B., and Eiben, A. E. (2015). Evolutionary robotics: what, why, and where to. *Frontiers in Robotics and AI*, 2:4.

Dong, H., Zhang, J., Wang, T., and Zhang, C. (2023). Symmetry-aware robot design with structured subgroups. In *International Conference on Machine Learning*, pages 8334–8355. PMLR.

Erez, T., Tassa, Y., and Todorov, E. (2012). Infinite-horizon model predictive control for periodic tasks with contacts. *Robotics: Science and systems VII*, 73.

Fan, J., Tang, H., Przystupa, M., Phielipp, M., Miret, S., and Berseth, G. (2024). Efficient design-and-control automation with reinforcement learning and adaptive exploration. In *AI for Accelerated Materials Design-NeurIPS*.

Gumbel, E. J. (1954). *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office.

Gupta, A., Savarese, S., Ganguli, S., and Fei-Fei, L. (2021). Embodied intelligence via learning and evolution. *Nature communications*, 12(1):5721.

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

Jain, M., Bengio, E., Hernandez-Garcia, A., Rector-Brooks, J., Dossou, B. F., Ekbote, C. A., Fu, J., Zhang, T., Kilgour, M., Zhang, D., et al. (2022). Biological sequence design with gflownets. In *International Conference on Machine Learning*, pages 9786–9801. PMLR.

Kandasamy, K., Krishnamurthy, A., Schneider, J., and Póczos, B. (2018). Parallelised bayesian optimisation via thompson sampling. In *International conference on artificial intelligence and statistics*, pages 133–142. PMLR.

Kang, Y., Kim, D., and Kim, K. (2019). Urdf generator for manipulator robot. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 483–487. IEEE.

Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978.

Liu, R., Gao, J., Zhang, J., Meng, D., and Lin, Z. (2021). Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10045–10067.

Lu, H., Wu, Z., Xing, J., Li, J., Li, R., Li, Z., and Shi, Y. (2025). Bodygen: Advancing towards efficient embodiment co-design. In *The Thirteenth International Conference on Learning Representations*.

Luck, K. S., Amor, H. B., and Calandra, R. (2020). Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning. In *Conference on Robot Learning*, pages 854–869. PMLR.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. (2022). Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967.

Michalewicz, Z. (2013). *Genetic algorithms+ data structures= evolution programs*. Springer Science & Business Media.

Mouret, J.-B. and Clune, J. (2015). Illuminating search spaces by mapping elites. *arXiv e-prints*, pages arXiv–1504.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sims, K. (1994). Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372.

Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE.

Veenstra, F. and Glette, K. (2020). How different encodings affect performance and diversification when evolving the morphology and control of 2d virtual creatures. In *Artificial Life Conference Proceedings 32*, pages 592–601. MIT Press.

Wang, T., Liao, R., Ba, J., and Fidler, S. (2018). Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*.

Wang, T., Zhou, Y., Fidler, S., and Ba, J. (2019). Neural graph evolution: Towards efficient automatic robot design. In *International Conference on Learning Representations*.

Yuan, Y., Song, Y., Luo, Z., Sun, W., and Kitani, K. M. (2021). Transform2act: Learning a transform-and-control policy for efficient agent design. In *International Conference on Learning Representations*.