MorphoGen: Free-Form XML Evolution for Robot Morphology Design

Hongyuan Su[1,2] **Yu Zheng**[1,3,*] **Yong Li**[1,2,*]

[1]Department of Electronic Engineering, BNRist, Tsinghua University, Beijing, China
[2]Zhongguancun Academy, Beijing, China
[3]Massachusetts Institute of Technology, Cambridge, MA, USA

[*]Corresponding authors: Yu Zheng (yu_zheng@mit.edu); Yong Li (liyong07@tsinghua.edu.cn)

Abstract

Designing effective robot morphologies is challenging because the design space is combinatorial, non-differentiable, and costly to evaluate. Existing methods often rely on predefined structural grammars, fixed morphology templates, or handcrafted edit operators, limiting the range of designs they can explore. We present **MorphoGen**, an LLM-driven evolutionary framework that directly modifies complete MuJoCo XML descriptions without a predefined structural grammar or fixed edit-action space. MorphoGen combines structure-aware directional feedback for guided mutations and crossovers with an island-based MAP-Elites archive and a hierarchical proxy evaluator for efficient search. On a suite of challenging locomotion benchmarks, MorphoGen discovers novel and high-performing morphologies, significantly outperforming strong baselines by 14.4% in downstream locomotion evaluation. Quantitative analyses further demonstrate the structural diversity of the high-performing morphologies it discovers. Code is available at https://github.com/tsinghua-fib-lab/MorphoGen.

1 Introduction

The functionality of a robotic agent is fundamentally determined by its physical form, or morphology [36, 19]. This morphology is often defined by a structured file, such as the XML format used by MuJoCo [31], which serves as the robot's genotype specifying the precise configuration of skeletons, joints, and their corresponding attributes. Automating the search for an optimal genotype is a highly valuable endeavor, as manually designing and adapting robot morphologies for different tasks is a laborious, intuition-driven process. However, this automation is a non-trivial challenge. The primary difficulty lies in the immense, combinatorial design space, and it is impossible to exhaustively iterate through all potential skeletal structures and attributes. This challenge is compounded by the high computational cost of evaluation, as each candidate morphology necessitates a costly inner loop of control policy to assess its fitness.

To manage this complexity, existing methods avoid operating directly on raw XML definitions, instead introducing an intermediate graph-based abstraction as a mediator. In this common paradigm, a robot's morphology is represented as a directed graph where topological elements (nodes and edges) denote body parts, joints, and connections with physical attributes [27]. Traditional evolutionary algorithms have long navigated this space by applying simplistic, hand-crafted genetic operators, such as adding or deleting nodes and edges [32, 27, 4]. However, these operations are semantically blind; they modify the graph's structure without any inherent understanding of the physical consequences, leading to inefficient exploration. More recent learning-based approaches also build upon this graph abstraction, using graph neural networks (GNN) and reinforcement learning (RL) to learn a policy over a set of predefined graph modification actions [36, 8].

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

While this learns a more intelligent strategy for applying edits, the edits themselves are still constrained to the same limited set of graph operations. Recent LLM-based robot-design systems reduce some manual engineering, but they still typically expose the model to a restricted design interface: RoboMorph uses predefined grammar rules [24], RoboMoRe focuses on parameter refinement within fixed skeletal templates [9], and LASeR introduces reflection over a structured morphology search process [29]. In essence, current paradigms sidestep the original, more challenging problem of direct XML generation. By operating within a constrained action space, they simplify the problem at the cost of sacrificing much of the expressiveness and higher degrees of freedom inherent to XML morphology definitions.

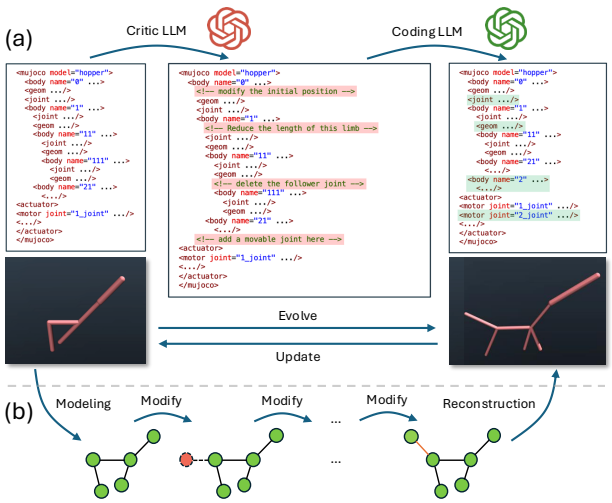

Figure 1: Comparison between paradigms of robot morphology design. (a) The MorphoGen framework, in which LLMs directly operate on the raw XML representation to generate complete robot morphologies in a single step. (b) The conventional approach, which relies on simplified intermediate models and requires multiple handcrafted modification steps.

Recent advancements in large language models (LLMs) have demonstrated their remarkable creative capabilities of generating raw forms of structured language [1, 10], such as computer codes [3]. In particular, powerful LLMs combined with automated evaluators have proven effective at evolving and optimizing complex algorithms, as highlighted by the groundbreaking AlphaEvolve project [22, 26]. This success unlocks a new possibility for robotic design: directly operating on XML definitions in their original, open form. Here we introduce MorphoGen, a framework that treats a robot's XML genotype as a form of specialized code and adapts LLM-based code optimization to morphology design. Our core idea is to replace predefined, hand-crafted graph operators with direct XML evolution operators that can modify topology, joints, actuator assignments, and physical attributes in the simulator-native representation (see Figure 1). To guide this process, we develop structure-aware directional feedback that identifies morphology-level failure modes and conditions the XML operator on evaluated designs. To improve the efficiency of evolution, we further use XML archive seeding to populate the initial island archive with promising genotypes, together with a hierarchical proxy evaluator to balance the solution quality and evaluation efficiency.

The main contributions are as follows: (1) We formulate morphology search as direct XML-level evolution and instantiate it with validity-aware LLM operators that edit complete XML genotypes. (2) We integrate structure-aware directional feedback, an island-based MAP-Elites archive with XML archive seeding, and a hierarchical proxy evaluator to support effective exploration of the flexible morphology space. (3) We conduct extensive experiments on challenging locomotion benchmarks, which validate that our approach generates high-performing morphologies that outperform strong baselines by 14.4% in downstream motoring performance and discovers a diverse range of novel designs.

2 Related Works

2.1 Robot Morphological Design

Robot morphological design is a complex bilevel optimization problem [15, 12, 11, 14, 18, 5, 23], with the outer level optimizing the structure and the inner level refining the control policy for motion. Traditional approaches such as genetic algorithms [7, 2, 17] and evolutionary strategies [33, 6, 37], rely on heuristic or search-based methods, which are constrained by predefined search spaces and require explicit design rules. Recent RL-based methods [36, 8, 28, 13] have advanced control policy optimization for various skeletons but handle structures indirectly through parameterized representations, often imposing constraints for feasibility. In contrast, our approach directly evolves

the robot's skeleton by modifying its underlying XML file, enabling exploration through a broader simulator-level design interface and facilitating the discovery of efficient configurations outside many graph-action spaces.

2.2 LLM-based Evolutionary Frameworks

Recent studies [30, 16, 35, 22] have explored using LLMs as intelligent search operators within evolutionary optimization frameworks, capitalizing on their strong capabilities in code generation and in-context reasoning. For instance, RoboMorph [24] represents robot designs as grammar strings and employs best-shot prompting to iteratively propose new production rules. RoboMoRe [9] addresses the co-design of robot morphology and reward functions using masked templates for parameter optimization. LASEr [29] introduces reflection mechanisms to promote structural diversity. While these works demonstrate the potential of LLMs in morphology optimization, they typically rely on manually predefined solution spaces. Consequently, the generated morphologies often suffer from limited diversity and frequently fail to outperform traditional RL-based baselines. To address these limitations, MorphoGen reframes morphology design as unconstrained XML code generation. By introducing a novel critic LLM that provides directional guidance and a hierarchical proxy fitness function, MorphoGen significantly improves search efficiency within the complex, open-ended design space.

3 Preliminaries

We represent a robot morphology as a rooted kinematic tree $\mathcal{T} = (V, E, r)$, where $V = \{b_0, b_1, \ldots, b_n\}$ denotes the set of rigid bodies, $r \in V$ is the root body, and E is the set of directed edges connecting parent and child bodies. Each body b_i is associated with physical attributes Φ_i, such as its geometry and mass properties. Each edge $e_{ij} = (b_i, b_j) \in E$ represents a kinematic connection between two bodies and is associated with joint and actuation properties $\Theta_{ij} = (J_{ij}, A_{ij})$, where J_{ij} describes the joint configuration and A_{ij} describes the corresponding actuation properties.

The morphology is encoded as a complete XML description $\mathcal{X}$, which specifies the body hierarchy, joints, geometries, actuators, and associated physical parameters. Let F denote the serialization process that maps a morphology description to its XML representation:

$$\mathcal{X} = F(\mathcal{T}, \Phi, \Theta). \tag{1}$$

Unlike methods that optimize only a predefined set of graph operations or structural parameters, MorphoGen directly evolves the XML genotype $\mathcal{X}$.

Given an initial population of XML genotypes $\mathcal{P}_0 = \{X_0^{(1)}, X_0^{(2)}, \ldots, X_0^{(N)}\}$, the evolutionary search generates successive candidate populations $\mathcal{P}_g$ for generations $g = 1, \ldots, G$. Each candidate morphology $\mathcal{X}$ is paired with a controller π_X whose policy is optimized for that morphology. Its locomotion performance is measured in a physics-based simulator by a task-specific objective $\mathcal{L}(\mathcal{X}, \pi_X)$. The morphology optimization problem is therefore formulated as

$$\mathcal{X}^* = \arg\max_{\mathcal{X} \in \Omega_X} \mathcal{L}(\mathcal{X}, \pi_X), \tag{2}$$

where Ω_X denotes the space of candidate XML morphologies considered by the search and π_X denotes the controller associated with morphology $\mathcal{X}$ under the specified evaluation protocol.

The robot morphology optimization problem is thus formulated as:

Input: An initial XML structure $\mathcal{X}_0$, and an initial motion controller π_0.

Output: An optimized XML structure $\mathcal{X}^*$, along with its associated controller $\pi_{\mathcal{X}^*}$.

Objective: Find the morphology $\mathcal{X}^* = \arg\max_{\mathcal{X} \in \Omega_{\mathcal{X}}} \mathcal{L}(\mathcal{X}, \pi_{\mathcal{X}})$ that maximizes the fitness.

4 Method

We propose MorphoGen, an LLM-based evolutionary framework that directly iterates on the robot's raw XML genotype to generate high-performing skeletal structures. As illustrated in Figure 2, MorphoGen employs a guided evolutionary search strategy. Beginning with initial robot morphologies,

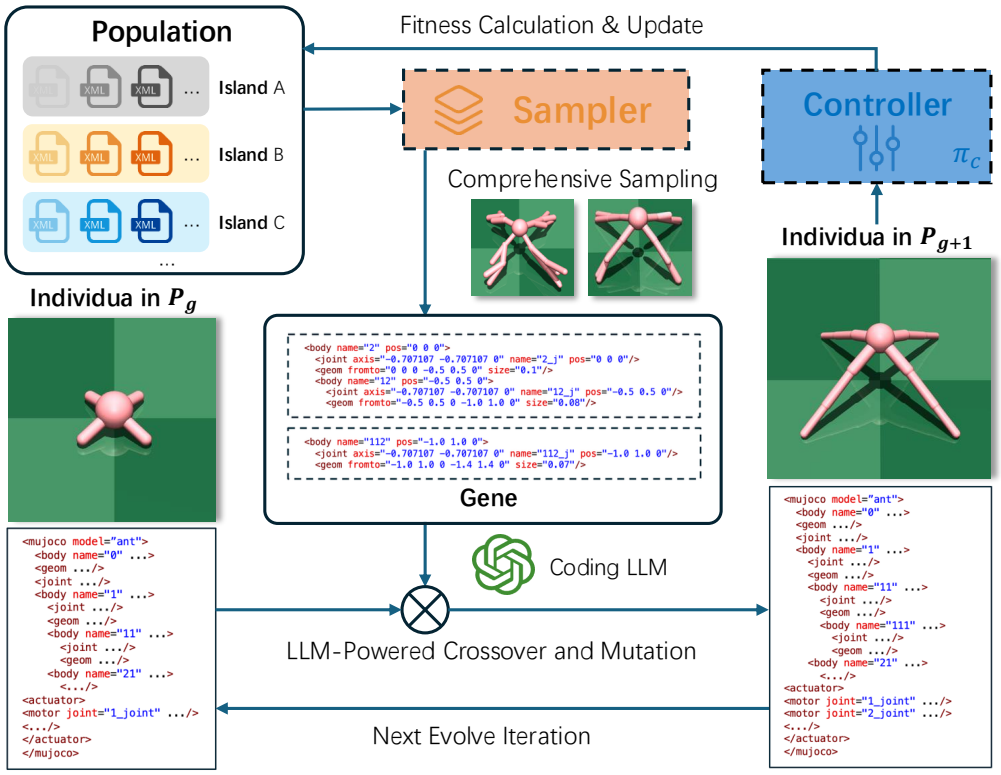

Figure 2: Overview of the MorphoGen evolutionary pipeline. The optimizer stores evaluated XML genotypes in an island archive, samples parents with fitness and MAP-Elites diversity criteria, applies direct XML evolution operators, and evaluates valid children through a cascade controller pipeline.

MorphoGen validates XML syntax, extracts structural features, trains or reuses a controller for fitness evaluation, and stores the resulting genotype in an island archive. A structure-aware feedback module then analyzes evaluated morphologies through their XML tree, actuator coverage, locomotion score, and optional trajectory descriptors. This feedback serves as a directional search signal for the XML evolution operator, which performs semantically-aware mutation and crossover operations directly on parent XML genotypes to produce a new generation of offspring morphologies. Subsequently, each newly generated XML genotype is evaluated with a hierarchical proxy fitness by efficient control policy fine-tuning. This closed-loop process transforms the design search from blind graph edits into targeted XML-level optimization.

4.1 Direct XML Evolution Framework

Drawing inspiration from AlphaEvolve [22], we formulate robot morphology optimization as the evolution of complete XML genotypes. The main components of the evolutionary process are defined as follows:

- **Individual.** Each individual $\mathcal{X}$ is a complete MuJoCo XML description that specifies a robot's body structure, joints, actuators, and physical attributes. The XML description serves as the genotypedirectly optimized by MorphoGen.
- **Gene.** Genes $\{G_1, G_2, \ldots, G_k\}$ denote conceptually meaningful blocks within an XML description, such as body, joint, geometry, or actuator definitions.
- **Island.** An island $I = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$ maintains a collection of candidate morphologies and supports local evolutionary search.
- **Population.** The population $\mathcal{P} = \{I_1, I_2, \ldots, I_m\}$ consists of multiple islands and maintains the candidate morphologies explored by the evolutionary process.

Designing robot morphology differs drastically from code generation of AlphaEvolve [22] in every stage of the pipeline, and we propose parent selection, crossover and mutation, and fitness calculation

steps that prioritize morphology understanding, XML genotype modification, and locomotion optimization. In specific, one generation P_g of robot morphology XML definitions are evolved to the next P_{g+1} as follows:

Comprehensive Sampling. To construct the parent pool for the next generation from the current population $\mathcal{P}_g$, we employ a hybrid selection strategy that balances the preservation of high-performing morphologies with exploration of structurally distinct candidates. The strategy combines three criteria: (1) Elitism [24, 21]: The top individuals with the highest fitness scores of each island are selected to preserve and propagate high-performing genotypes. (2) Diversity [20]: To encourage exploration of structurally distinct morphologies, we select individuals based on their structural dissimilarity, measured using the Tree Edit Distance(TED) between their morphology trees. (3) Randomness [9]: A subset of individuals is sampled uniformly at random to retain candidates that may not be selected by the fitness-based or diversity-based criteria. Together, these criteria balance the retention of high-performing candidates with structural exploration and random sampling.

In the released implementation, each task uses $G = 100$ generations and $m = 5$ islands, with a population size of 30 and an elite archive size of 8. The MAP-Elites feature map uses `complexity_score` and `actuator_efficiency`, each discretized into 8 bins. The configured selection ratios are 0.2 for elites, 0.5 for exploration, and 0.3 for exploitation. Appendix B reports the full implementation configuration.

Direct XML Crossover and Mutation. Unlike traditional evolutionary algorithms that rely on simplistic, random operators, MorphoGen leverages a coding LLM to perform semantically coherent XML modifications. We define explicit operator contracts rather than treating generation as open-ended text completion. For mutation, the XML operator receives a target parent M_p, its fitness F_p, structural feature vector ϕ_p, feedback C_p, and evolution history H_g:

$$M_{\text{child}} = O_{\text{mut}}(\mathcal{X}_p, F_p, \phi_p, C_p, H_g).$$

For crossover, the operator additionally receives a set of reference parents $\mathcal{R} = \{(\mathcal{X}_r, F_r, \phi_r, C_r)\}$,

$$M_{\text{child}} = O_{\text{cross}}(\mathcal{X}_p, F_p, \phi_p, C_p, \mathcal{R}, H_g).$$

The implementation uses XML full-rewrite mode rather than line diffs because valid morphology edits often require coordinated changes across `body`, `joint`, `geom`, and `actuator` blocks. The operator merges high-quality XML blocks from the target and sampled parents, producing a structurally sound offspring that inherits meaningful traits. In the mutation operation, novel variations are introduced into a single parent M_p, modifying its XML genotype to reach better motion performance.

Fitness Calculation. After generating a new XML genotype $\mathcal{X}$, we fix its structure and employ an RL algorithm [36] to develop an adaptive controller π_C for locomotion. The RL training process optimizes π_C over a fixed number of PPO steps in a simulated environment, after which the control policy is deployed to evaluate the robot's performance. The fitness $F(\mathcal{X})$ is defined by the Euclidean distance traveled from the starting point within a fixed simulation horizon. During evolutionary search, this fitness is estimated using the hierarchical proxy evaluator described in Section 4.2, allowing candidate morphologies to be ranked without fully optimizing a controller for every candidate. Details of robot controller optimization are provided in Appendix D.

4.2 Efficient Evolutionary Optimization

Evolutionary algorithms are notoriously sample-inefficient, which is even worse in robot design due to the costly inner-loop optimization required to evaluate each skeletal structure. MorphoGen addresses this bottleneck with three key innovations: XML archive seeding to initialize the island archive with valid and behaviorally diverse genotypes, structure-aware directional feedback to enable targeted optimization, and hierarchical proxy fitness to accelerate evaluation. Together, these components transform the evolutionary search into a more efficient and directed process.

XML Archive Seeding. Though direct XML evolution allows flexible modifications to robot morphologies, arbitrary edits do not necessarily produce functionally effective candidates. A syntactically valid XML description may still contain inconsistent structural references, insufficiently actuated joints, or a morphology with poor locomotion capability. MorphoGen therefore initializes the island archive with a set of valid and behaviorally diverse XML genotypes to provide useful starting points for evolutionary search. During subsequent evolution, the XML operator can modify body topology,

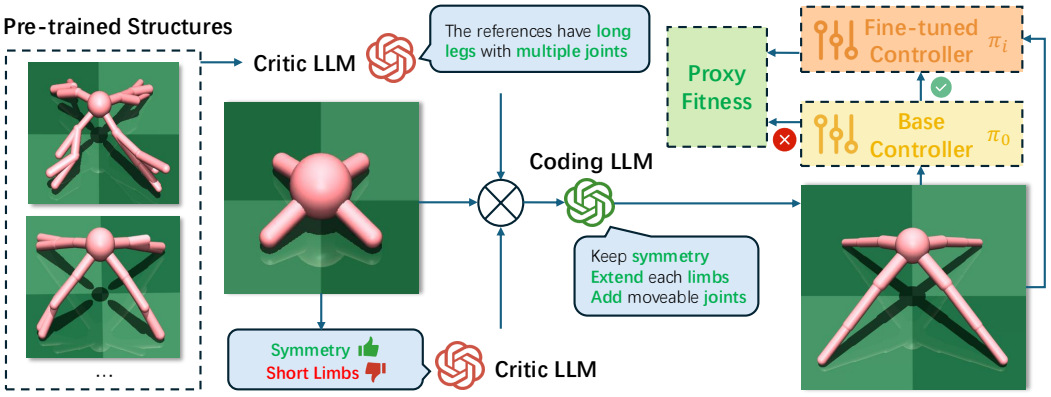

Figure 3: Illustration of the efficient evolutionary optimization process in MorphoGen.

joint configurations, limb geometry, actuator assignments, and physical attributes, including changes beyond those present in the initial morphologies.

Structure-Aware Directional Feedback. To guide XML evolution beyond unguided structural modifications, MorphoGen employs a critic LLM to analyze each evaluated morphology and provide actionable design suggestions. The feedback module considers the XML structure, actuator coverage, locomotion fitness, and available trajectory statistics to identify potential structural strengths and weaknesses associated with the observed performance. The resulting feedback is incorporated into the context of the coding LLM during mutation and crossover. It guides the operator toward targeted modifications, such as adjusting limb geometry, revising joint configurations, or changing body segments, while allowing coordinated changes across the XML description. These suggestions act as soft search guidance rather than a predefined set of permitted structural operations.

Proxy Fitness. Learning an optimal control policy for each XML genotype M_i from scratch is computationally expensive, often requiring millions of simulation interactions. However, recognizing that an offspring is often a small modification of its parent, we posit that their optimal control policies are also close in the parameter space, which enables efficient policy reuse. Therefore, instead of full retraining, we approximate the best control policy of the current morphology by briefly fine-tuning a reusable base policy. Moreover, with potentially thousands of generated skeletal structures throughout the whole process, even fine-tuning can be time-consuming. To overcome the above challenges, MorphoGen employs an evaluator-side reusable controller π_0 as a coarse motion probe, followed by a two-stage hierarchical proxy fitness evaluation. Specifically, in the first stage, each skeletal structure M_i is rolled out with π_0 and the corresponding locomotion score is used as the coarse fitness F_i^1. This evaluation is extremely fast and does not require any additional training costs. We take F_i^1 as an initial screening criterion to ignore those structures that have obvious functional issues. For skeletal structures with F_i^1 greater than a threshold τ, the second stage is triggered. In the second stage, π_0 is fine-tuned against structure M_i for a small controller budget to get an adaptive controller π_i, which is applied to calculate the locomotion score as fitness F_i^2. The final proxy fitness F_i is determined by the hierarchical evaluation procedure:

$$F_i = \begin{cases} F_i^{(1)}, & \text{if } F_i^{(1)} \leq \tau, \\ F_i^{(2)}, & \text{if } F_i^{(1)} > \tau, \end{cases} \tag{3}$$

This hierarchical approach drastically reduces the average evaluation time per individual, allowing MorphoGen to assess a much larger volume of candidates.

In summary, through an integrated framework of direct XML evolution, structure-aware directional feedback, and accelerated fitness evaluation, we achieve efficient optimization in the raw XML definition space. We now validate the effectiveness of this approach experimentally.

5 Experiments

5.1 Experiment Settings

Baselines We compare MorphoGen with traditional evolutionary methods, including Neural Graph Evolution (NGE) [32], Evolutionary Structure Search (ESS) [6], and Random Graph Search (RGS);

Table 1: Comparison of locomotion scores across methods, environments, and controller budgets. "/" indicates the method is not applicable to the corresponding environment.

Method	Fine-tuned Controller (FC)				Fully Trained Controller (FTC)			
	Hopper	Ant	Swimmer	Gap	Hopper	Ant	Swimmer	Gap
ESS	$163.4_{\pm 14.1}$	$108.9_{\pm 10.1}$	$93.1_{\pm 11.5}$	$99.6_{\pm 13.0}$	$658.7_{\pm 52.7}$	$1387.5_{\pm 113.3}$	$349.0_{\pm 28.9}$	$218.6_{\pm 19.7}$
RGS	$483.5_{\pm 54.4}$	$508.3_{\pm 31.0}$	$107.8_{\pm 13.8}$	$240.7_{\pm 26.3}$	$849.5_{\pm 69.0}$	$1394.2_{\pm 79.0}$	$301.4_{\pm 31.5}$	$670.8_{\pm 39.8}$
NGE	$1725.9_{\pm 115.3}$	$896.2_{\pm 41.8}$	$119.0_{\pm 7.0}$	$601.4_{\pm 40.6}$	$2647.1_{\pm 224.0}$	$2015.3_{\pm 205.4}$	$513.9_{\pm 42.2}$	$822.3_{\pm 58.0}$
QDSA	$1816.4_{\pm 196.4}$	$974.6_{\pm 89.2}$	$591.3_{\pm 25.3}$	$559.8_{\pm 54.5}$	$6918.2_{\pm 779.2}$	$3621.7_{\pm 297.6}$	$753.5_{\pm 40.0}$	$1817.2_{\pm 248.1}$
T2A	$2143.2_{\pm 192.2}$	$1140.2_{\pm 136.2}$	$703.4_{\pm 24.6}$	$651.0_{\pm 76.3}$	**8130.4**$_{\pm 934.5}$	$4268.7_{\pm 351.4}$	$877.4_{\pm 65.9}$	$2147.5_{\pm 322.6}$
SARD	$1841.4_{\pm 210.4}$	$1174.8_{\pm 79.6}$	$970.7_{\pm 17.3}$	*896.3*$_{\pm 67.2}$	$7109.0_{\pm 839.3}$	*4347.1*$_{\pm 391.7}$	$892.8_{\pm 66.6}$	$2532.7_{\pm 366.2}$
LOKI	$1520.8_{\pm 164.6}$	$835.6_{\pm 76.4}$	$890.3_{\pm 37.4}$	$802.4_{\pm 77.1}$	$5977.2_{\pm 652.5}$	$3033.6_{\pm 255.2}$	$899.5_{\pm 60.7}$	$2696.4_{\pm 356.0}$
RoboMorph	/	$853.3_{\pm 60.8}$	/	/	/	$2842.3_{\pm 353.4}$	/	/
RoboMoRe	$1612.0_{\pm 193.2}$	$1022.4_{\pm 87.8}$	$808.1_{\pm 17.2}$	*718.8*$_{\pm 68.8}$	$6697.1_{\pm 492.4}$	$3514.1_{\pm 278.9}$	$693.9_{\pm 34.5}$	$2833.8_{\pm 286.5}$
LOKI-LLM	$1758.6_{\pm 190.5}$	$985.1_{\pm 90.5}$	$1002.6_{\pm 42.4}$	$930.7_{\pm 89.2}$	$6915.6_{\pm 754.6}$	$3576.4_{\pm 301.1}$	$1013.2_{\pm 68.3}$	$3127.5_{\pm 413.4}$
MorphoGen[1]	**2474.1**$_{\pm 163.6}$	**1560.4**$_{\pm 110.3}$	**1056.3**$_{\pm 15.5}$	**969.1**$_{\pm 62.4}$	$\underline{7409.0}_{\pm 654.2}$	**4740.6**$_{\pm 282.6}$	**1058.0**$_{\pm 37.4}$	**5652.7**$_{\pm 280.5}$
MorphoGen[2]	$\underline{2265.2}_{\pm 135.8}$	$\underline{1425.5}_{\pm 71.3}$	$\underline{1024.4}_{\pm 15.1}$	$\underline{946.7}_{\pm 66.0}$	*7127.9*$_{\pm 741.5}$	$\underline{4357.9}_{\pm 267.2}$	$\underline{1029.4}_{\pm 45.2}$	$\underline{5258.3}_{\pm 193.8}$
MorphoGen[3]	*2215.0*$_{\pm 123.5}$	*1181.3*$_{\pm 74.3}$	*1008.6*$_{\pm 19.5}$	$860.8_{\pm 57.7}$	$6384.4_{\pm 307.2}$	$3994.2_{\pm 233.4}$	*960.1*$_{\pm 17.2}$	*3877.7*$_{\pm 141.0}$

reinforcement-learning-based methods, including Transform2Act (T2A) [36] and Symmetry-Aware Robot Design (SARD) [8]; and LLM-based morphology optimization methods, including RoboMorph [24] and RoboMoRe [9]. To further examine the contribution of direct XML evolution, we include a QDSA-style MAP-Elites baseline [21], LOKI [13], and LOKI-LLM, a variant that replaces LOKI's predefined morphology operators with LLM-based operators. These comparisons cover conventional evolutionary search, learning-based morphology optimization, quality-diversity search, and LLM-guided morphology evolution. Additional implementation details for the baselines and MorphoGen are provided in Appendix A and Appendix B, respectively.

Evaluation We evaluate all methods in the MuJoCo simulator across four locomotion environments: Hopper, Ant, Swimmer, and Gap Crosser (Gap) [31]. Performance is measured by the forward distance traveled within a fixed simulation horizon. The methods differ in their search procedures: NGE, ESS, RGS, and MorphoGen optimize fixed morphology structures using PPO-based [25] controller training, whereas T2A and SARD jointly optimize morphology and control. LLM-based methods follow their respective morphology-generation and search protocols. To ensure a consistent final evaluation, each method's selected morphology is fixed after its prescribed search budget and evaluated under two standardized controller-training settings. The fine-tuned controller (FC) is trained for 0.3M PPO environment steps, representing performance under a limited controller-training budget. The fully trained controller (FTC) is trained until convergence or a maximum budget of 5.0M PPO environment steps. Within each setting, all selected morphologies use the same controller initialization, training strategy, optimization hyperparameters, reward function, simulator-step budget, and evaluation protocol. Each morphology is evaluated using six controller random seeds, and we report the mean locomotion score and its variation across seeds.

5.2 Overall Performance

To completely demonstrate the quality and diversity of the generated skeleton structures of our methods, we report the top three results, and with name MorphoGen^i denotes the top i-th result. We summarize the motion performance of the generated robot skeleton structure in Table 5.1, from which we can have the following observations,

- **Traditional evolutionary methods struggle to discover high-performing morphologies.** Without a clear exploration direction or strong priors, methods such as NGE, ESS, and RGS often produce degenerate structures with poor locomotion ability. These approaches consistently yield the lowest performance, representing a 67.4% performance reduction compared to RL-based methods. This gap suggests that predefined structural modifications and unguided sampling can limit the effectiveness of morphology search in the extremely large and highly non-convex design spaces.
- **Restricted morphology representations limit search performance.** Methods like T2A, RoboMorph and RoboMoRe rely on predefined structural grammars or fixed skeletal templates, while QDSA and LOKI employ predefined morphology operators or a learned latent morphology space. Although these methods enable effective search within their respective representations, their design interfaces may limit the range of expressible structural modifications. Even after replacing LOKI's

Table 2: Average computational cost of differnt methods across four tasks. Percentages in the MorphoGen row indicate relative changes against the arithmetic mean of RoboMorph and RoboMoRe.

Method	LLM inference		Compute resources		
	Calls	Tokens	Wall-clock (h)	GPU-h	Simulator steps
NGE	–	–	2.2	0	2.4M
T2A	–	–	52.8	42.7	281.2M
SARD	–	–	61.8	51.3	328.4M
RoboMorph	400	12.48M	13.0	11.2	52.1M
RoboMoRe	400	10.82M	21.6	19.9	96.4M
MorphoGen	**800** (+100.0%)	**25.28M** (+117.0%)	**10.7** (-38.2%)	**7.7** (-50.5%)	**34.8M** (-53.1%)

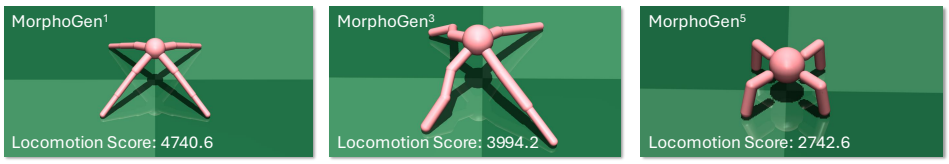

Figure 4: Various high-performing morphologies produced by MorphoGen.

predefined operators with LLM-based modifications, MorphoGen still achieves an average FC improvement of 27.1% across the four tasks. This result suggests that directly evolving XML descriptions can better exploit the flexibility of the design space.

- **MorphoGen achieves strong performance across tasks and controller-training budgets.** Our method consistently generates quality skeleton structures that achieve superior locomotion performance across diverse environments and training budgets. Specifically, it improves FC performance by an average of 14.4%, with task-wise gains ranging from 4.1% on Gap Crosser to 32.8% on Ant. These findings indicate that our approach not only enables diverse and high-quality skeleton generation, but also demonstrates strong potential for real-world applications where training budgets are limited.

We further quantify the computational resources consumed by each method, including LLM calls, token usage, wall-clock time, GPU-hours, and simulator interactions. Table 2 summarizes the resource consumption reported for searches across the four locomotion tasks. Notably, though MorphoGen uses twice as many LLM calls as RoboMorph and RoboMoRe on average, it reduces the wall-clock time by 38.2%, the GPU usage by 50.5% and simulator interactions by 53.1% relatively. MorphoGen employs structure-aware directional feedback to guide mutations and crossovers toward targeted morphology modifications, while XML archive seeding provides useful starting candidates. In addition, the hierarchical proxy evaluator screens candidates using a reusable base controller and allocates further controller fine-tuning to candidates that pass the task-specific threshold, reducing the need for expensive evaluation of less promising designs. Taken together, these results suggest that MorphoGen can trade increased LLM inference for reduced simulator and controller-training costs. Its advantage lies not in minimizing every individual resource metric, but in combining more targeted candidate generation with selective evaluation to achieve a favorable resource–performance trade-off.

5.3 Morphological and Behavioral Diversity

Beyond identifying high-performing morphologies, an effective morphology optimization method should explore a diverse set of viable design solutions. We therefore analyze the diversity of solutions from structural diversity, which characterizes differences in morphology configurations, and behavioral diversity, which captures differences in the resulting locomotion patterns.

Structural diversity. We first examine the morphology structures discovered by different methods, as shown in Figure 5. In contrast to the intricate and often incomprehensible structures generated by RL-based methods, the skeletons produced by our approach are clear, intuitive and visually interpretable. Specifically, our method naturally evolves various bio-inspired structures that resemble real-world organisms: the generated ant skeleton resembles a spider, the hopper skeleton looks like a horse, and the swimmer skeleton mimics an octopus. Figure 4 further illustrates the top-performing ant skeletons generated by MorphoGen which achieve high scores while possessing distinct structures. This highlights our method's capacity to explore a wide variety of effective design solutions rather

Table 3: Structural and behavioral diversity of morphologies. Higher values indicate greater diversity.

Method	Pairwise nTED↑	Novelty↑	Uniqueness↑	D_{behavior} ↑
T2A	0.22	0.37	3.0	0.55
SARD	0.36	0.33	3.8	0.60
RoboMorph	0.31	0.18	2.6	0.63
RoboMoRe	0.18	0.14	1.2	0.47
MorphoGen	**0.57**	**0.52**	**4.8**	**0.81**

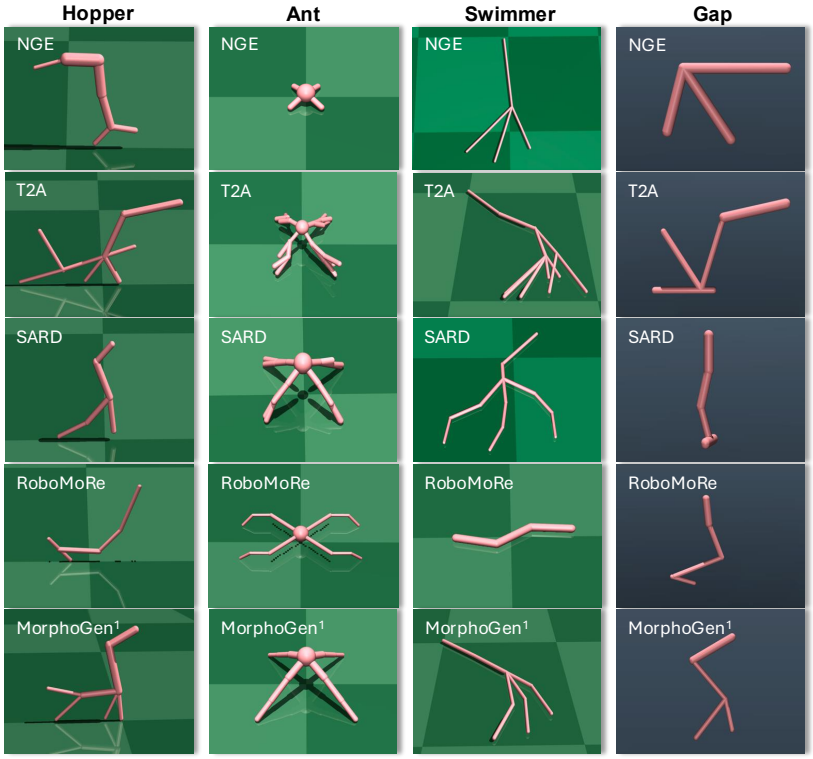

Figure 5: Representative robot morphologies generated by different methods.

than converging on a single, narrow set of configurations. To quantify these differences, we evaluate the selected morphologies with pairwise normalized tree-edit distance (nTED), novelty relative to the initial seed, and topological uniqueness (see Appendix E). As shown in Table 3, MorphoGen achieves the highest pairwise nTED of 0.57, a 58.3% increase over SARD (0.36), indicating greater structural variation among the selected designs. It also attains a novelty score of 0.52, outperforming T2A (0.37) by 40.5%, and achieves the highest topological uniqueness among all compared methods. These complementary metrics indicate that MorphoGen's selected designs differ substantially from one another and from their initial seeds. In contrast, methods based on predefined structural rules or fixed skeletal templates impose restrictions on the modifications that can be explored. Direct XML evolution allows MorphoGen to coordinate changes across body topology, geometry, joints, and actuators, supporting broader exploration of the morphology design space.

Behavioral Diversity. Structural diversity is valuable when it translates into distinct functional behaviors, rather than merely different geometric configurations. We therefore investigate whether the diverse morphologies discovered by MorphoGen also exhibit varied locomotion patterns. For each selected morphology, we evaluate its fine-tuned controller (FC) and characterize the resulting motion using behavioral descriptors that capture forward velocity, lateral drift, survival time, and body height. We then quantify behavioral diversity by measuring the average pairwise distance between these descriptors (see Appendix E for details). As shown in Table 3, MorphoGen achieves a behavioral diversity score of 0.81, exceeding RoboMorph (0.63) by 28.6% and RoboMoRe (0.47) by 72.3%. This indicates that MorphoGen's structural diversity is accompanied by greater variation in the locomotion behaviors of the selected designs. Unlike RoboMoRe, which refines parameters within fixed skeletal templates, MorphoGen can directly modify body topology, joint configurations,

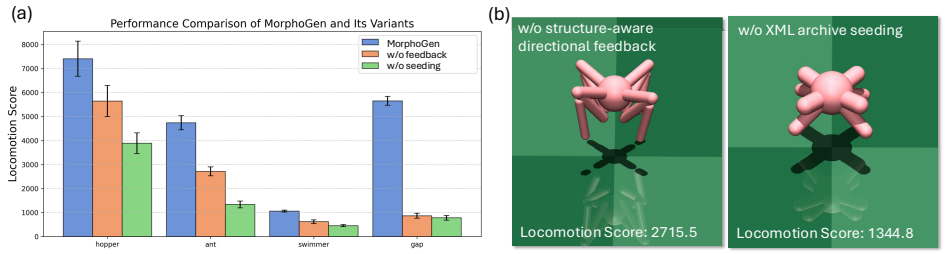

Figure 6: Ablation study on key components of MorphoGen.

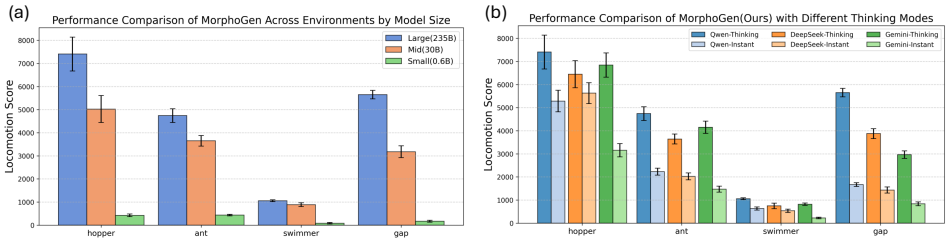

Figure 7: (a) The performance of Qwen3 with varying parameter quantities as the backbone model. (b) Performance comparison of models with and without a reasoning component.

and actuator assignments in the XML representation. Such coordinated structural changes can alter how robots move and interact with the environment, providing a broader range of possible locomotion behaviors.

5.4 Ablation Study

We conduct ablation studies to examine the effectiveness of structure-aware directional feedback, XML archive seeding, and the LLM backbone used for XML evolution. The component ablation results are shown in Figure 6, and the backbone selection results are shown in Figure 7.

Removing either structure-aware directional feedback or XML archive seeding substantially degrades the quality of evolved morphologies, reducing the average motion score by 50.5% and 65.7%, respectively. Without structure-aware feedback, the XML evolution operator lacks targeted morphology-level guidance derived from XML structure, actuator coverage, locomotion fitness, and failure modes, making exploration much less efficient. This effect is especially pronounced in the Gap Crosser task, where the score drops from 5652.7 to 866.4, corresponding to an 84.7% decrease. Meanwhile, removing XML archive seeding also causes a severe performance decline. Without a seeded island archive, the search starts from a sparse region of the free-form XML space and spends more evaluations on invalid or weakly actuated ineffective morphologies. For example, on the Ant task, the resulting morphology achieves only 28.2% of the motion efficiency of the full framework.

We further study the influence of the LLM backbone. As shown in Figure 7, reducing model capability sharply decreases the locomotion score from 7409.0 to 424.3, a 94.3% drop. This indicates that free-form XML evolution requires a strong backbone to maintain XML validity, coordinate bodies, joints, geoms, and actuators, and translate structure-aware feedback into effective morphology edits.

6 Discussion and Limitations

This work introduces MorphoGen, an evolutionary framework that redefines robot morphology design as a code generation problem using LLMs to evolve raw XML genotypes. By integrating structure-aware feedback, archive seeding, and hierarchical proxy evaluation, it discovers diverse, high-performing morphologies across locomotion benchmarks, especially under limited controller-training budgets. Our experiments are restricted to simulation and do not establish physical feasibility or real-world safety. Although the XML interface avoids a fixed morphology grammar, search remains conditioned on prompts, initialization, simulator validity, and controller compatibility. The observed gains reflect the combined framework rather than the XML representation alone.

Future work will investigate physical validation, robustness across independent searches, and morphology–controller co-evolution to improve reliability and reduce evaluation costs.

Acknowledgements and Disclosures

This work is supported in part by the National Key Research and Development Program of China (Grant No. 2024YFC3307603) and the National Natural Science Foundation of China (Grant No. U23B2030). This work is also supported in part by the Zhongguancun Academy (Grant No.s 02012403), Tsinghua University–Toyota Research Center, and the Beijing National Research Center for Information Science and Technology (BNRist). The authors declare no competing interests.

References

[1] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Brodbeck, L., Hauser, S., and Iida, F. Morphological evolution of physical robots through model-free phenotype development. *PloS one*, 10(6):e0128444, 2015.

[3] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. D. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[4] Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOlution*, 7(1): 11–23, 2014.

[5] Cheney, N., Bongard, J., SunSpiral, V., and Lipson, H. On the difficulty of co-optimizing morphology and control in evolved virtual creatures. In *Artificial life conference proceedings*, pp. 226–233. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2016.

[6] Cheney, N., Bongard, J., SunSpiral, V., and Lipson, H. Scalable co-optimization of morphology and control in embodied machines. *Journal of The Royal Society Interface*, 15(143):20170937, 2018.

[7] Datta, R., Pradhan, S., and Bhattacharya, B. Analysis and design optimization of a robotic gripper using multiobjective genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(1):16–26, 2015.

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

[9] Fang, J., Sun, Y., Ma, C., Lu, Q., and Yao, L. Robomore: Llm-based robot co-design via joint optimization of morphology and reward. *arXiv preprint arXiv:2506.00276*, 2025.

[10] Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[11] He, Z. and Ciocarlie, M. Morph: Design co-optimization with reinforcement learning via a differentiable hardware model proxy. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7764–7771. IEEE, 2024.

[12] Hu, J., Whitman, J., and Choset, H. Glso: Grammar-guided latent space optimization for sample-efficient robot design automation. In *Conference on Robot Learning*, pp. 1321–1331. PMLR, 2023.

[13] Jeon, H., Eftekhar, A., Walsman, A., Zeng, K.-H., Farhadi, A., and Krishna, R. Convergent functions, divergent forms. *Advances in Neural Information Processing Systems*, 38:46049–46076, 2026.

[14] Jiang, Y., Zhang, T., Ho, D., Bai, Y., Liu, C. K., Levine, S., and Tan, J. Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2884–2890. IEEE, 2021.

[15] Kim, Y., Pan, Z., and Hauser, K. Mo-bbo: Multi-objective bilevel bayesian optimization for robot and behavior co-design. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9877–9883. IEEE, 2021.

[16] Liu, F., Lin, X., Yao, S., Wang, Z., Tong, X., Yuan, M., and Zhang, Q. Large language model for multiobjective evolutionary optimization. In *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 178–191. Springer, 2025.

[17] Liu, X., Pathak, D., and Kitani, K. M. Herd: Continuous human-to-robot evolution for learning from human demonstration. *arXiv preprint arXiv:2212.04359*, 2022.

[18] Lu, H., Wu, Z., Xing, J., Li, J., Li, R., Li, Z., and Shi, Y. Bodygen: Advancing towards efficient embodiment co-design. *arXiv preprint arXiv:2503.00533*, 2025.

[19] Matthews, D., Spielberg, A., Rus, D., Kriegman, S., and Bongard, J. Efficient automatic design of robots. *Proceedings of the National Academy of Sciences*, 120(41):e2305180120, 2023.

[20] Mouret, J.-B. and Clune, J. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[21] Nordmoen, J., Veenstra, F., Ellefsen, K. O., and Glette, K. Map-elites enables powerful stepping stones and diversity for modular robotics. *Frontiers in Robotics and AI*, 8:639173, 2021.

[22] Novikov, A., Vũ, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J., Mehrabian, A., et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

[23] Pathak, D., Lu, C., Darrell, T., Isola, P., and Efros, A. A. Learning to control self-assembling morphologies: a study of generalization via modularity. *Advances in Neural Information Processing Systems*, 32, 2019.

[24] Qiu, K., Pałucki, W., Ciebiera, K., Fijałkowski, P., Cygan, M., and Kuciński, Ł. Robomorph: Evolving robot morphology using large language models. *arXiv preprint arXiv:2407.08626*, 2024.

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

[26] Sharma, A. Openevolve: an open-source evolutionary coding agent, 2025. URL `https://github.com/codelion/openevolve`.

[27] Sims, K. Evolving virtual creatures. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pp. 699–706. 2023.

[28] Singh, B., Kumar, R., and Singh, V. P. Reinforcement learning in robotic applications: a comprehensive survey. *Artificial Intelligence Review*, 55(2):945–990, 2022.

[29] Song, J., Yang, Y., Xiao, H., Peng, W., Yao, W., and Wang, F. Laser: Towards diversified and generalizable robot design with large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[30] Tian, H., Han, X., Wu, G., Guo, A., Zhou, Y., Zhang, J., Li, S., Wei, J., and Zhang, T. An llm-empowered adaptive evolutionary algorithm for multi-component deep learning systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 20894–20902, 2025.

[31] Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

[32] Wang, T., Zhou, Y., Fidler, S., and Ba, J. Neural graph evolution: Towards efficient automatic robot design. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL `https://openreview.net/forum?id=BkgWHnR5tm`.

[33] Xu, J., Spielberg, A., Zhao, A., Rus, D., and Matusik, W. Multi-objective graph heuristic search for terrestrial robot design. In *2021 IEEE international conference on robotics and automation (ICRA)*, pp. 9863–9869. IEEE, 2021.

[34] Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[35] Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., and Chen, X. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2023.

[36] Yuan, Y., Song, Y., Luo, Z., Sun, W., and Kitani, K. M. Transform2act: Learning a transform-and-control policy for efficient agent design. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL `https://openreview.net/forum?id=UcDUxjPYWSr`.

[37] Zhao, A., Xu, J., Konaković-Luković, M., Hughes, J., Spielberg, A., Rus, D., and Matusik, W. Robogrammar: graph grammar for terrain-optimized robot design. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020.

A Baseline Details

We provide additional implementation details for all baselines below.

- **Neural Graph Evolution (NGE)** [32] formulates robot design as a graph search problem and employs graph neural networks as controllers to share policies across different morphologies. This significantly reduces evaluation cost during search and enables efficient discovery of kinematically meaningful designs.
- **Evolutionary Structure Search (ESS)** [6] jointly evolves morphology and control in soft robots. It introduces *morphological innovation protection*, which temporarily relaxes selection pressure on recently mutated morphologies, allowing controllers to readapt and thus preventing premature convergence.
- **Random Graph Search** [32] performs naive random sampling of graph-structured morphologies without guided exploration. Although simple, it serves as a strong baseline to highlight the advantage of structured evolutionary search.
- **Transform2Act** [36] treats robot design as a conditional decision-making process and learns policies that jointly determine both the morphology and control. By leveraging reinforcement learning with modular architectures, it achieves improved sample efficiency over pure evolutionary approaches.
- **Symmetry-Aware Robot Design (SARD)** [8] incorporates group-theoretic symmetry constraints into the robot design process. By searching for and enforcing optimal symmetry subgroups, it reduces the design space, improves control stability, and produces more efficient morphologies across diverse tasks.
- **RoboMorph** [24] employs a structured grammar library to describe the sequence of morphology designs. An evolutionary loop retains the highest-fitness individuals as parents for the next generation.
- **RoboMoRe** [9] fixes the graph structure of morphology and searches for the best parameters of limbs. It follows a coarse-to-fine paradigm and jointly optimizes morphology and the reward function.

B Implementation Details

Table 4: Main implementation details from the released configuration files.

Item	Value
Outer morphology generations	100
Checkpoint interval	5 generations
Maximum XML length	50,000 characters
Population size	30
Elite archive size	8
Number of islands	5
Feature map	`complexity_score`, `actuator_efficiency`
Feature bins	8
Elite / exploration / exploitation ratios	0.2 / 0.5 / 0.3
Top / diverse programs in context	3 / 5
Evolution model	`qwen-plus`
Feedback model	`qwen-max`
LLM max output tokens	8192
Cascade thresholds	Hopper 0.20, Ant 0.025, Swimmer 0.20, Gap 0.25
Parallel evaluations	1
FC controller budget	0.3M PPO steps
FTC controller budget	until convergence or 5.0M PPO steps
LLM backbone	Qwen3 [34]

MAP-Elites Archive The MAP-Elites archive uses two structural descriptors to defineits feature map. Let N_{body} denote the number of articulated body nodes, J the set of named hinge joints, and A

Table 5: FC locomotion scores during MorphoGen search. At iteration 0, Standard and Warm-start denote the two initial morphologies. At later iterations, values denote the best-so-far FC score among morphologies ranked by the search-time objective over the archive.

Task	Iteration 0		Evolution iteration				
	Standard	Warm-start	20	40	60	80	100
Hopper	356.5	422.7	916.5	1435.6	1879.5	2254.8	2474.1
Ant	162.2	278.8	398.2	674.0	983.7	1329.4	1560.4
Swimmer	324.0	457.4	631.1	764.3	918.4	1046.4	1056.3
Gap Crosser	85.7	101.4	189.6	312.6	465.7	555.1	969.1

the set of joints referenced by motor actuators. We define the complexity score and actuator efficiency as

$$c(M) = \min\left(\frac{N_{\text{body}}}{10}, 1\right), \qquad e(M) = \frac{|J \cap A|}{|J|}. \tag{4}$$

Both descriptors lie in $[0, 1]$ and are discretized into eight equal-width bins, yielding an 8×8 MAP-Elites feature map. The archive retains the elite candidate in each occupied cell.

C Warm-start Protocol.

We consider two types of task-specific initial morphologies. First, standard seeds are obtained from the canonical morphologies provided by the Gymnasium MuJoCo environments. Second, baseline warm-start seeds are obtained by running T2A with approximately 10% of its full search budget and selecting the morphology with the highest native search-time fitness for each task. For methods that support warm-start initialization and can represent the corresponding morphology, we use the same task-specific seed pool to reduce differences attributable to initialization quality. When a seed cannot be directly represented by a baseline's predefined grammar or morphology parameterization, we document the incompatibility rather than assuming identical search spaces.

To examine the influence of initialization quality, we compare MorphoGen's standard task-specific seeds with warm-start seeds obtained from T2A. The standard seeds are the canonical morphologies provided by the Gymnasium MuJoCo environments, whereas each warm-start seed is selected as the highest-fitness morphology found by T2A using approximately 10% of its full search budget. At iteration 0, we evaluate both initial seeds using the fine-tuned controller (FC) protocol. At subsequent iterations, we report the FC score of the best-so-far morphology ranked by MorphoGen's search-time objective over the complete archive.

Table 5 summarizes the results. MorphoGen improves substantially over both initialization sources across all four tasks. Relative to the warm-start seed, the final FC score increases by approximately 5.9× on Hopper, 5.6× on Ant, 2.3× on Swimmer, and 9.6× on Gap Crosser. These improvements indicate that the search continues to discover higher-performing morphologies after initialization, rather than merely retaining the quality of the initial designs. The gains are particularly pronounced on Hopper, Ant, and Gap Crosser, where the final scores are substantially higher than those of the warm-start seeds.

D Controller Optimization

PPO Optimizer. The controller optimization for each generated robot structure is performed using RL to develop an adaptive control policy π_C for locomotion tasks. The problem is formulated as an infinite-horizon discounted Markov Decision Process (MDP), defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition probability function, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\rho_0 : \mathcal{S} \to [0, 1]$ is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. For a fixed robot morphology, the objective is to learn a control policy $\pi_C : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative discounted reward:

$$J(\pi_C) = \mathbb{E}_{s_0 \sim \rho_0, a_t \sim \pi_C(\cdot|s_t), s_{t+1} \sim P(\cdot|s_t, a_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right],$$

where $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$ are the state and action at time t.

We employ Proximal Policy Optimization (PPO) [25] to optimize the policy, following the approach in Yuan et al. [36] for locomotion control.

Controller Training Curves. To further showcase the efficacy of our approach, we visualize the motion performance of skeleton structures generated by different methods throughout controller training. As shown in Figure 8, our method significantly outperforms baseline approaches with the FTC, achieving an average performance improvement of 40.5% across all environments. Moreover, our approach generates skeleton structures that facilitate rapid learning of effective control policies. Specifically, while baseline methods often require hundreds of PPO updates to achieve acceptable locomotion performance, our method consistently produces skeletons that achieve effective motion strategies in fewer than 100 PPO updates.

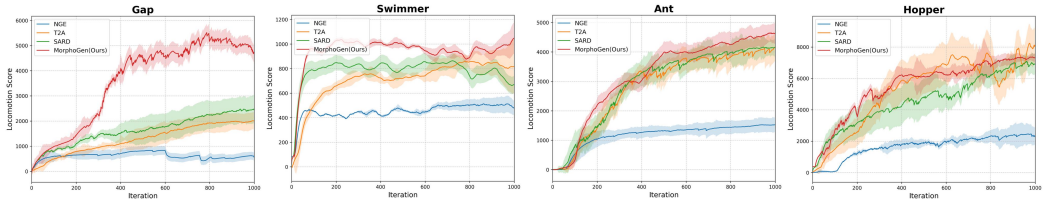

Figure 8: Training curves of locomotion metric, comparing MorphoGen with baseline methods.

E Diversity Analysis

This appendix provides the metric definitions and evaluation protocols underlying the structural and behavioral diversity analyses in Section 5.3. We first describe the attributed-tree representation and structural-diversity metrics, then detail the behavioral descriptors and their pairwise distance. Finally, we retain the original Ant-specific analysis of limb-length parameter diversity as complementary evidence.

E.1 Structural Diversity Metrics

Attributed morphology trees. Each robot morphology is parsed from its MuJoCo XML description into a rooted attributed tree T_i. Nodes represent rigid bodies, and edges represent kinematic connections. The node and edge attributes encode the structural and physical properties used in the comparison, including geometry types and dimensions, joint types, joint axes and ranges, and actuator assignments.

Pairwise normalized tree-edit distance. We measure structural dissimilarity using the normalized tree-edit distance:

$$d_{\text{struct}}(i, j) = \frac{\text{TED}(T_i, T_j)}{\max(|T_i|, |T_j|)}, \tag{5}$$

where $|T_i|$ denotes the number of body nodes in morphology tree T_i. The tree-edit distance measures the minimum cost of transforming one attributed tree into another through node or edge insertions, deletions, and attribute substitutions. The normalization accounts for differences in tree size. A larger value indicates greater structural dissimilarity.

For a selected set $\mathcal{S}$ containing K morphologies, pairwise nTED is calculated as the mean distance over all unordered pairs:

$$D_{\text{nTED}}(\mathcal{S}) = \frac{2}{K(K-1)} \sum_{\substack{i,j \in \mathcal{S} \\ i<j}} d_{\text{struct}}(i, j). \tag{6}$$

Novelty relative to initialization. To assess whether the search discovers designs beyond the initial morphology, we measure the normalized structural distance between each selected morphology and its associated initial seed. Let $s(i)$ denote the seed associated with candidate i. The novelty score for a selected set is

$$N(\mathcal{S}) = \frac{1}{K} \sum_{i \in \mathcal{S}} d_{\text{struct}}(i, s(i)). \tag{7}$$

A higher score indicates that the selected designs differ more substantially from their initial seeds.

Topological uniqueness. We additionally assess the diversity of body-tree topology among the top-five selected morphologies. Two morphologies are considered topologically equivalent if their body trees are isomorphic under the criterion used by the implementation. The uniqueness metric counts the number of distinct topological equivalence classes in the selected set.

E.2 Behavioral Diversity Metrics

Structural differences do not necessarily imply different locomotion behaviors. We therefore analyze the trajectories generated by the trained fine-tuned controllers (FC) associated with the selected morphologies.

For each morphology, behavioral descriptors are averaged over 10 evaluation episodes. The descriptor vector is

$$\mathbf{b}_i = [v_{\text{forward}}, d_{\text{lateral}}, t_{\text{survival}}, h_{\text{body}}], \tag{8}$$

where the components represent mean forward velocity, lateral drift, survival or fall time, and mean root-body height.

We normalize the descriptors according to the evaluation protocol and denote the normalized vector by $\widehat{\mathbf{b}}_i$. Behavioral diversity is measured by the mean pairwise Euclidean distance among the selected morphologies:

$$D_{\text{behavior}}(\mathcal{S}) = \frac{2}{K(K-1)} \sum_{\substack{i,j \in \mathcal{S} \\ i<j}} \left\| \widehat{\mathbf{b}}_i - \widehat{\mathbf{b}}_j \right\|_2. \tag{9}$$

The reported behavioral-diversity score is normalized to $[0, 1]$, with larger values indicating greater variation among the measured locomotion behaviors.

F Proxy Fitness

The primary bottleneck in evolutionary morphology optimization is the extremely high computational cost of training an optimal controller for every candidate design. Our hierarchical proxy fitness strategy is designed to address this challenge by significantly reducing the evaluation time per morphology, thus enabling greater exploration. The time difference between evaluation methods is substantial: a typical proxy fitness calculation spends approximately 2 minutes, whereas training an optimal controller from scratch costs about 3 hours (a $100\times$ increase). Using the full controller for fitness calculation is computationally impractical for extensive evolutionary search.

To study the impact of the proxy fitness on overall optimization performance, we compare three variants on a fixed computational budget, reducing the morphology generations for the full-controller variant to make the study feasible.

Table 6: Performance Comparison of Proxy Fitness Variants.

Variant	Generations	Total Time	Hopper	Ant	Swimmer
w/o Proxy Fitness (Optimal Controller)	40	$\sim$ 108h	5796.0	3862.8	789.7
w/ Proxy Fitness (Limited Iterations)	40	$\sim$ 1.2h	5018.3	3194.6	754.8
w/ Proxy Fitness (Full Search Budget)	100	$\sim$ 4h	**7409.0**	**4740.6**	**1058.0**

While using an optimal controller yields slightly higher quality designs when the number of morphology generations is artificially fixed at 40, its massive computational overhead makes extensive

Table 7: Sensitivity of proxy-fitness performance to the task-specific threshold. Each cell reports FC score / wall-clock time. τ_0 denotes the default threshold for the corresponding task.

Task	$\boldsymbol{\tau = 0}$	$\boldsymbol{\tau = 0.5\tau_0}$	$\boldsymbol{\tau = \tau_0}$	$\boldsymbol{\tau = 2\tau_0}$
Hopper	2443.6 / 6.9 h	2318.9 / 4.3 h	2474.1 / 2.8 h	2038.5 / 1.7 h
Ant	1582.7 / 7.4 h	1476.0 / 4.6 h	1560.4 / 3.1 h	1149.3 / 1.9 h
Swimmer	1142.3 / 6.5 h	1106.3 / 4.0 h	1056.3 / 2.4 h	894.6 / 1.5 h
Gap Crosser	1024.4 / 6.7 h	977.3 / 4.1 h	969.1 / 2.4 h	779.8 / 1.6 h

exploration impossible. In contrast, the proxy fitness allows the algorithm to evaluate orders of magnitude more morphologies within the same time frame. By utilizing the proxy fitness and facilitating a larger number of morphology generations, MorphoGen achieves 28.2% better overall performance, while consuming 96% fewer compute resources per generation. This confirms that the efficiency provided by the hierarchical proxy fitness is essential for effective evolutionary search in the XML design space.

Threshold Sensitivity. The task-specific threshold τ determines whether a candidate proceeds from the inexpensive first-stage evaluation to the more costly second-stage controller fine-tuning. To examine the sensitivity of this decision, we vary the threshold while keeping the remaining evaluation settings unchanged. We consider four settings: $\tau = 0$, $0.5\tau_0$, τ_0, and $2\tau_0$, where τ_0 is the default threshold for each task. We report both the final FC score and the corresponding wall-clock time.

Table 7 shows that the threshold controls a trade-off between evaluation cost and morphology performance. Setting $\tau = 0$ yields higher FC scores than the default setting on Ant, Swimmer, and Gap Crosser, but increases the evaluation time substantially. Conversely, increasing the threshold to $2\tau_0$ reduces runtime across all four tasks, but lowers the final FC score in every task. The default threshold therefore provides a practical balance between candidate evaluation cost and search performance, although the best trade-off varies across tasks.

G XML Operator and Feedback Interfaces

The following prompts specify the interfaces used by the coding and feedback LLMs in MorphoGen. The coding LLM generates complete MuJoCo XML genotypes through mutation or crossover, while the feedback module provides morphology-level guidance based on structural information and evaluation outcomes. The design principles in these prompts act as soft proposal priors rather than a predefined structural grammar or a fixed set of permitted morphology-edit actions.

Coding LLM

System Message
You are an expert in robotics, MuJoCo simulation, and robot morphology optimization. Your task is to improve the locomotion performance of a robot by modifying its complete MuJoCo XML description.

Optimization Objective
Generate a candidate morphology that is expected to improve locomotion performance while maintaining a valid, internally consistent, and simulatable robot description. Use the supplied fitness information, structural features, feedback suggestions, and evolution history to inform your decisions.

Design Principles
Consider the following principles when they are relevant to the current morphology and task:

1. **Structural consistency:** Maintain a connected and coherent body hierarchy, valid parent-child relationships, and consistent references among bodies, joints, geometries, and actuators.
2. **Locomotion capability:** Consider how body proportions, limb configurations, joint arrangements, and contact geometry may affect propulsion, balance, stability, and interaction with the terrain.
3. **Actuation and control compatibility:** Ensure that actuator definitions reference valid joints and remain compatible with the controller's action interface. Avoid introducing unintended changes to the number or ordering of controllable degrees of freedom.
4. **Physical plausibility:** Choose physically meaningful geometry, dimensions, masses, joint axes, and joint ranges. Avoid configurations likely to cause invalid dynamics or ineffective movement.
5. **Complexity-performance trade-off:** Consider whether added structural complexity is likely to provide a useful functional benefit relative to its potential impact on control difficulty.

These principles guide candidate generation but do not define a fixed morphology grammar, prescribed topology, or exhaustive list of allowable structural modifications.

XML Editing Instructions

- Perform the requested mutation or crossover using the supplied parent morphology information.
- Identify a primary design objective based on the available fitness, structural features, directional feedback, and evolution history. Focus the modification on this objective when possible.
- Allow coordinated changes to related XML components when they are necessary to maintain structural consistency or to realize the intended morphology modification.
- For crossover, use the reference parents as sources of potentially useful structural characteristics. Do not simply concatenate XML fragments or copy components without checking their dependencies and references.
- Preserve useful aspects of the target parent when they are compatible with the proposed improvement. Avoid unrelated changes that have no clear motivation from the available information.

- Treat feedback suggestions as guidance rather than mandatory operations. Other modifications may be considered when supported by the morphology, fitness, or evolution history.
- Maintain valid XML syntax and the structural consistency required by MuJoCo. Do not invent body, joint, or actuator references that do not exist in the resulting XML.
- Do not assume that a structural modification will necessarily improve performance. Generate a plausible candidate for subsequent simulation and fitness evaluation.

Output Requirements
Return only the complete MuJoCo XML description of the new candidate. Do not return explanations, Markdown fences, comments outside the XML, or a partial XML patch. The candidate will subsequently be checked and evaluated by the optimization pipeline.

User Message
Task and Operation
Operation type: {operation_type}
Task configuration: {task_configuration}

Target Parent
Current morphology XML: {current_xml}
Current locomotion fitness: {current_fitness}
Current structural features: {current_structure_features}
Current morphology tree: {current_structure_tree}

Reference Parents for Crossover
Reference parent information, if applicable: {reference_parents}

Directional Feedback
Feedback from the structure-aware critic: {critic_feedback}

Evolution History
Relevant previous candidates, modifications, and evaluation outcomes: {evolution_history}

Generate a new candidate morphology that follows the operation type and uses the available information to guide the modification. Return the complete XML only.

Structure-Aware Feedback Module

System Message
You are a robotics expert acting as a structural critic for an evolutionary robot morphology optimization system using MuJoCo. Your task is to analyze an evaluated robot morphology and provide one concise, actionable suggestion that may guide the next morphology modification.

Analysis Objectives
Use the available XML description, morphology tree, locomotion fitness, and trajectory information to identify a promising direction for further morphology optimization.
Consider the following questions when relevant:

- Which structural characteristics appear useful for the observed locomotion performance?
- Which observed failure patterns or performance limitations suggest a potentially useful morphology change?
- Which primary structural aspect could be modified to address the identified limitation?
- Would the proposed modification require coordinated changes to related bodies, joints, geometries, or actuators?

Feedback Requirements

1. Provide one primary modification direction that is specific enough to guide the coding LLM.

2. Ground the suggestion in the supplied morphology information, fitness, and trajectory descriptors whenever possible. Distinguish observed evidence from hypotheses about the causes of poor performance.
3. Identify the relevant structural component or relationship and briefly explain why changing it may improve locomotion.
4. When useful, mention related XML components that may need coordinated adjustments to maintain structural consistency or controller compatibility.
5. Prefer actionable structural guidance over generic statements such as "improve stability" or "optimize locomotion" without identifying a plausible modification.
6. Do not prescribe an exact XML patch, generate a new morphology, or restrict the coding LLM to a fixed set of structural operations.
7. Do not invent trajectory observations, failure causes, or performance measurements that are absent from the input. If the evidence is insufficient to identify a specific weakness, state the uncertainty and provide a cautious suggestion based on the available structural information.

The feedback is intended to guide evolutionary search, not to guarantee an improvement or define a mandatory modification. The coding LLM may consider other changes when supported by the available information.

User Message
Current morphology XML: {current_xml}
Current locomotion fitness: {current_fitness}
Morphology tree summary: {current_structure_tree}
Structural features: {current_structure_features}
Trajectory descriptors: {trajectory_summary}
Available trajectory descriptors may include forward velocity, lateral drift, survival or fall time, body height, and observed contact failures.
Primary focus area for this feedback: {focus_area}

Analyze the supplied information and return one concise directional suggestion. Briefly identify a relevant structural strength or limitation, propose a plausible modification, and explain its expected effect on locomotion.

Output Format
Observed evidence: Summarize the relevant structural or trajectory observation.
Primary suggestion: Describe one actionable morphology modification direction.
Rationale: Explain why the modification may help, including any relevant dependencies between XML components.
Uncertainty: Briefly note any important uncertainty when the available evidence does not establish the cause of the observed behavior.

H Use of LLMs

LLMs are an algorithmic component of MorphoGen for XML genotype editing and structure-aware feedback. Outside that role, LLMs were used only for language polishing. The research ideas, system design, experiments, and analysis are controlled by the authors.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 describe direct XML-level morphology evolution with an island archive, LLM-based XML operators, structure-aware feedback, and a two-

stage proxy evaluator. Section 5 reports the benchmark and diversity results, and Section 6 discusses the scope and limitations.

Guidelines:

- The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Section 6, Discussion and Limitations, discusses simulation-only evaluation, dependence on prompts and initialization, simulator validity, controller compatibility, and the difficulty of attributing full-system gains solely to the XML representation.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present new theoretical theorems, lemmas, or formal proof claims. Section 3 and Appendix D formulate the morphology optimization problem and controller objective rather than establish theoretical guarantees.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 describes the optimization pipeline, Section 5.1 describes the benchmark protocol, and Appendices A–D provide baseline, implementation, initialization, and controller information. Appendices F and G describe proxy evaluation and the XML operator and feedback interfaces, respectively.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The abstract provides the public MorphoGen repository URL. The repository provides the XML search code, controller-evaluation components, seed XML files, configurations, and task scripts.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Section 5.1 specifies the baselines, environments, evaluation metric, and FC/FTC controller budgets. Appendix B reports the main configuration, Appendix D describes controller optimization, and Appendix F describes proxy-fitness evaluation.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Table 1 reports performance variation across six controller seeds for each evaluated morphology and environment, and Figure 8 shows controller-training curves with variability. These characterize controller-level variation rather than variation across independent morphology searches.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Table 2 reports LLM calls, token usage, wall-clock time, GPU-hours, and simulator interactions. Appendix F provides additional proxy-evaluation timing comparisons.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The experiments are conducted in MuJoCo simulation and do not involve human subjects or private personal data. Section 6 acknowledges that physical feasibility and real-world safety remain unverified.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The Introduction and Discussion describe potential benefits of automated morphology design and more efficient controller learning. Section 6 acknowledges the limitations of simulation-only evaluation and the lack of verified physical feasibility and real-world safety.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released assets concern simulated morphology optimization and do not include high-risk pretrained generative models, scraped datasets, or human-subject data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper credits existing methods and software through citations, including MuJoCo, PPO, Transform2Act, and the robot-design baselines.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The public repository linked in the abstract provides the MorphoGen implementation and usage instructions, including seed XML files, configurations, evaluator components, and task scripts. Appendices B–D describe the implementation, initialization, and controller optimization.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments, participant studies, or data collection from human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB or equivalent human-subject approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The methodological use of LLMs for XML editing and structure-aware feedback is described in Section 4 and Appendix G. Appendix H states that other LLM usage was limited to language polishing.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.