Generative Flow Networks for Model Adaptation in Digital Twins of Natural Systems

Pascal Archambault[1*], Houari Sahraoui[1] and Eugene Syriani[1]

[1*] *DIRO*, Université de Montréal, Quebec, Canada.

*Corresponding author(s). E-mail(s): pascal.archambault@umontreal.ca;
Contributing authors: sahraouh@iro.umontreal.ca;
syriani@iro.umontreal.ca;

Abstract

Digital twins of natural systems must remain aligned with physical systems that evolve over time, are only partially observed, and are typically modeled by mechanistic simulators whose parameters cannot be measured directly. In such settings, model adaptation is naturally posed as a simulation-based inference problem. However, sparse and indirect observations often fail to identify a unique and optimal calibration, leaving several simulator parameterizations compatible with the available evidence. This article presents a GFlowNet-based approach to model adaptation for digital twins of natural systems. We formulate adaptation as a generative modeling problem over complete simulator configurations, so that plausible parameterizations can be sampled with probability proportional to a reward derived from agreement between simulated and observed behavior. Using a controlled environment agriculture case study based on a mechanistic tomato model, we show that the learned policy recovers dominant regions of the adaptation landscape, retrieves strong calibration hypotheses, and preserves multiple plausible configurations under uncertainty.

Keywords: digital twins, model-driven engineering, simulation-based inference, mechanistic simulation, model adaptation, generative modeling, generative flow networks, natural systems, cyber-biophysical systems, controlled environment agriculture

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

1 Introduction

Digital twins have become an important technology for the monitoring, prediction, and control of complex systems, particularly in industrial and cyber-physical settings [1, 2]. More recently, the concept has also been extended to natural systems in domains such as smart farming [3] and health sciences [4]. This extension raises distinct modeling challenges. In natural systems, observations are often sparse, noisy, delayed, destructive, or costly to acquire, while uncertainty remains central throughout model updating and downstream use [5].

In such settings, digital twin services rely on mechanistic simulators that encode domain knowledge about the underlying physical, chemical, or biological processes. Maintaining their relevance as the physical system evolves may require changes to model parameters or to the simulator structure itself. This work focuses on parametric adaptation, in which simulator configurations are calibrated against observations from a target context. Such calibration supports the broader validation of the digital twin, which also concerns the simulator's fidelity over time and suitability for downstream services. Because parameters are inferred indirectly from observed behavior, calibration poses an inverse problem. In natural systems, sparse and indirect observations often leave several parameterizations compatible with the same behavior, especially when many parameters cannot be measured directly [6].

This motivates the use of simulation-based inference, which addresses parameter inference in simulator-based models when the likelihood is unavailable or intractable. Candidate parameterizations can be proposed, instantiated in the simulator, and evaluated through their discrepancy with observed trajectories without requiring an explicit analytical likelihood. Within this setting, Generative Flow Networks [7] (GFlowNets) are appealing because they learn reusable stochastic policies that generate complete configurations with probability proportional to a reward. This makes them well suited to adaptation settings in which several high-quality parameterizations may remain plausible and should be preserved explicitly rather than collapsed into a single fitted solution.

This article studies GFlowNet-based model adaptation in a digital twin setting for greenhouse crop modeling, where a mechanistic tomato growth simulator [8] is adapted to new observational contexts from sparse trajectories. The greenhouse setting provides a useful testbed because it allows the approach to be analyzed first in a fully enumerable regime, where all candidate parameterizations in the considered search space can be evaluated exactly, and then in a larger regime where exhaustive characterization is no longer feasible.

The main contributions of this work are as follows:

- We position mechanistic model adaptation as a dedicated service within the digital twin lifecycle, where sparse observations and non-identifiability can leave multiple simulator configurations consistent with the available evidence.
- We propose a GFlowNet-based generative adaptation mechanism that constructs simulator configurations through structured, group-level parameter perturbations and learns to sample them in proportion to a reward derived from simulation–observation agreement. This allows the adaptation service to preserve multiple

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

high-quality calibration hypotheses rather than returning a single fitted configuration.
- We report an empirical study on a smart farming model, examining candidate retrieval, alignment with an enumerable target landscape, and the behavior of the approach as the adaptation space becomes larger and less tractable.

The remainder of this article is organized as follows. Section 2 reviews digital twins of natural systems and GFlowNets. Section 3 formulates model adaptation as a simulation-based service within the digital twin lifecycle and introduces the requirements that guide our approach. Section 4 presents the proposed generative formulation and its GFlowNet-based implementation. Section 5 showcases our results and discusses their implications for the model adaptation service, Section 6 presents the related works, and Section 7 concludes the paper.

2 Background

This section reviews the two main concepts framing our approach. We first introduce digital twins of natural systems and the particular challenges they raise for model adaptation under uncertainty. We then present GFlowNets as generative models for structured spaces.

2.1 Digital twins of natural systems

Digital twins are virtual representations of physical systems that remain linked to their real-world counterparts through data exchange and model updating, enabling monitoring, prediction, and decision support [9]. Recent work in model-driven engineering further distinguishes digital twins by the nature of the system being twinned, separating engineered from non-engineered contexts and explicitly including biological systems within the latter [10]. This distinction matters because moving from artifacts whose structure is largely known by design to natural systems introduces forms of variability, partial observability, and uncertainty that are not controlled in the same way. Natural systems may evolve through phases such as plan, growth, life, and end-of-life [10]. Across these phases, the digital twin must coordinate model artifacts, contextual data, and downstream services while the underlying biological or environmental system continues to change. Unlike many engineered systems whose relevant structure is largely fixed by design, natural systems may vary substantially over time as a result of development, environmental exposure, or management conditions. The models embedded in the twin then cannot be treated as static artifacts calibrated once and reused unchanged. They must remain suitable for the current lifecycle phase and operating context of the physical system.

Digital twins for natural systems have recently gained attention in domains such as ecology [11], and medicine [12]. In smart farming, they are commonly associated with the integration of sensing, simulation, and data-driven services to support monitoring, what-if analysis, and operational decision-making across greenhouses, fields, and livestock systems [13]. At the same time, the literature suggests that these systems remain

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

comparatively immature, and that robust predictive and prescriptive capabilities are still limited in many practical deployments [14].

2.2 GFlowNets

GFlowNets [15] are generative models designed for problems in which complete solutions are constructed through a sequence of dependent decisions and where several distinct high-quality solutions may all be relevant. Given a positive reward $R(x)$ defined over complete solutions x, a trained GFlowNet samples terminal objects with probability proportional to a reward function, so that higher-reward solutions are sampled more frequently while lower-reward alternatives retain proportionally smaller but non-zero mass. They are therefore well suited to multimodal inference settings in which the objective is not only to identify one good candidate, but also to represent and sample multiple plausible ones.

Formally, a GFlowNet operates on a directed acyclic graph whose nodes represent states, corresponding to partial objects under construction, and whose edges represent actions that extend one state into another. An initial state encodes an empty or default configuration, while terminal states correspond to complete candidate solutions. A GFlowNet learns a forward policy over this construction graph such that terminal objects are sampled in proportion to the reward. That is, for policy $\pi(x)$:

$$\pi(x) \propto R(x) \tag{1}$$

The term *flow* refers to the propagation of probability mass through the graph from the initial state toward terminal states, with training encouraging local transition probabilities to induce this reward-proportional distribution over complete objects [7].

This reward-proportional sampling distinguishes GFlowNets from methods designed to return a single optimum, such as gradient-based optimization or reward-maximizing reinforcement learning. Rather than concentrating search around one maximizer, GFlowNets learn a reusable policy whose sampling frequencies are calibrated to reward magnitudes across the entire solution space. This makes them useful both in multimodal search problems and in settings where samples must be generated repeatedly without solving a new search problem from scratch each time. Such uses have been reported, for example, in multi-objective optimization [16], posterior approximation over graph structures in Bayesian structure learning [17], and amortized inference for large language models [18]. These properties are directly relevant to simulator adaptation. When observations are sparse and several parameterizations can reproduce the observed trajectories with similar fidelity, the objective is not only to identify a strong candidate, but also to represent and sample multiple plausible ones. In this sense, GFlowNets provide a natural bridge toward the generative adaptation perspective developed in Section 4.

3 Simulation-based model adaptation in digital twins of natural systems

This section positions model adaptation as a service within the lifecycle of a digital twin of a natural system. We first situate this service with respect to the model

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

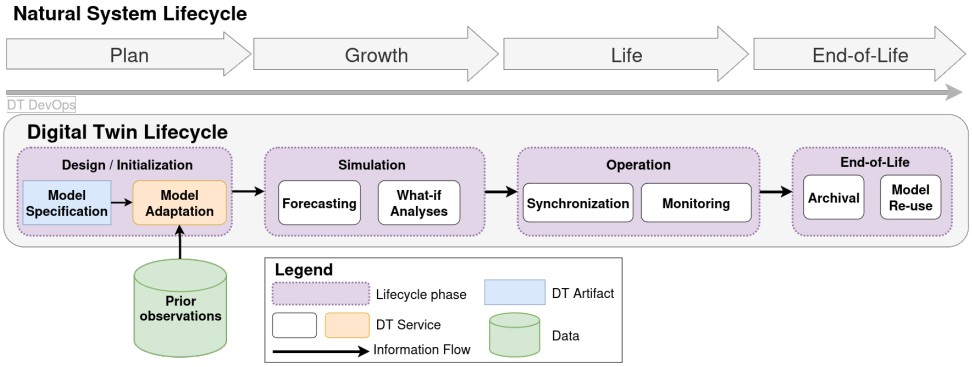


Fig. 1 Model adaptation as a service within the lifecycle of a digital twin for a natural system. During design and initialization, the service combines a model specification with contextual observations to produce an adapted simulator that can be consumed by downstream simulation and operational services.

artifacts and downstream services that depend on it. We then describe the internal simulation-based workflow through which candidate parameterizations are generated, evaluated, and refined from observations collected in the target context. On this basis, we finally identify the requirements that motivate the generative approach developed in Section 4.

3.1 Model adaptation in the digital twin lifecycle

Because digital twins of natural systems are tied to physical systems that evolve through phases such as plan, growth, life, and end-of-life [10], they must coordinate model artifacts and services across that lifecycle. These artifacts include model specifications, mechanistic simulators, configuration data, and contextual observations collected from the physical system. On top of them, the twin supports services such as monitoring, forecasting, what-if analysis, synchronization, and operational decision support. In the setting considered here, the mechanistic simulator is a central artifact because several downstream services depend on it as their behavioral model. As the natural system evolves, the twin must maintain coherence between the current observations, the simulator configuration, and the services that consume it. This is where model adaptation becomes necessary: not as an isolated calibration exercise, but as the mechanism through which the simulator remains suitable for the current context and lifecycle phase of the physical system.

Accordingly, model adaptation is best treated as a service within the digital twin lifecycle rather than as a calibration step performed only once during initialization. Its role is to take a base simulator together with contextual observations and produce an adapted model configuration that can be consumed by downstream digital twin services. In practice, this service maintains the operational relevance of the mechanistic simulator as new evidence becomes available, operating conditions change, or the natural system moves from one lifecycle phase to another.

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

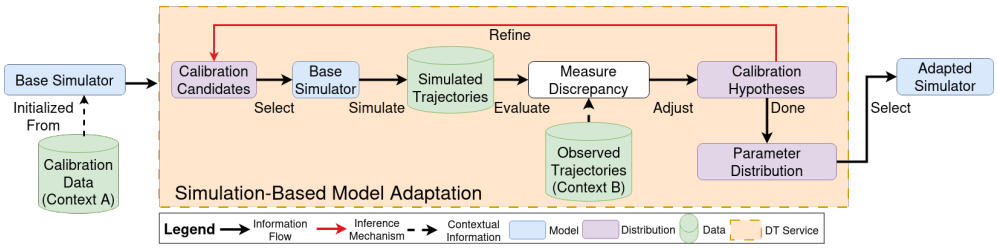


Fig. 2 Simulation-based workflow of the model adaptation service. Starting from a base simulator and observations collected in the target context, the service iteratively generates candidate parameterizations, executes the simulator, compares simulated and observed trajectories, and refines a distribution over plausible calibration hypotheses. The resulting adapted simulator, or set of plausible adapted configurations, can then be consumed by downstream digital twin services.

Fig. 1 situates this service within the design and initialization phase of the digital twin lifecycle, where a model specification and observations collected in the target context are combined to produce an adapted model. This adapted model is then passed to simulation services, including forecasting and what-if analysis, and ultimately supports downstream operational activities. The figure emphasizes that adaptation is not an isolated preprocessing step, but a model-management service that sustains the link between simulator specification, simulation, and use.

This role is especially important in natural systems, where the information required to update the simulator is often incomplete [19]. Many simulator parameters cannot be measured directly, while observations may be sparse, noisy, delayed, destructive, or costly to obtain. Moreover, similar observed trajectories may remain consistent with several simulator parameterizations. For this reason, model adaptation is not necessarily invoked continuously in real time. It may instead be triggered when new observations become available, when the physical system enters a new lifecycle phase, or when discrepancies between simulated and observed behavior indicate that the current simulator is no longer adequate for the target context. Model adaptation is thus a recurring service that supports the continued use of the simulator across the lifecycle of the natural system.

3.2 Simulation-based model adaptation

Having situated model adaptation within the digital twin lifecycle, we now turn to the internal workflow through which this service operates. In the present setting, the service performs parametric adaptation of an existing mechanistic simulator through simulation-based calibration. It seeks parameterizations whose simulated behavior agrees with observations from the target context, while keeping the simulator's structure unchanged. Broader forms of adaptation may instead require modifying model components or governing equations as the physical system evolves. Here, the target context denotes the conditions under which the physical system is observed and the simulator is intended to be used, for example a given growth stage, environmental regime, or management setting.

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

Fig. 2 summarizes this workflow. The process begins from a base simulator that has been calibrated on an original context. Candidate parameterizations are then instantiated by the service and are then simulated to produce trajectories under the target context. These simulated trajectories are compared with the observed trajectories through a discrepancy measure that reflects adaptation quality. The resulting scores are used to update a distribution over calibration hypotheses and to guide the generation of subsequent candidates. Through repeated simulation and evaluation, the service refines the set of plausible parameterizations and returns an adapted simulator configuration, or more generally a set of plausible adapted configurations, for downstream use.

This workflow is simulation-based because candidate hypotheses are assessed through forward execution of the mechanistic simulator rather than through an explicit analytical likelihood model [20]. This is well suited to the types of simulators considered here, where the relevant quantities can be generated by execution but the inverse mapping from observations to parameters is unavailable in closed form. Here, the inference target is the simulator parameterization rather than the time-evolving hidden state of a fixed model. The objective is to identify mechanistic parameter settings that make the simulated trajectories consistent with the observed system behavior with sufficient fidelity for downstream digital twin services.

This formulation highlights two consequences that motivate the remainder of the paper. First, because observations constrain parameters only indirectly through simulated behavior, adaptation must proceed through repeated hypothesis generation and evaluation. Second, because the available evidence may remain insufficient to isolate a unique calibration, the service must reason over a set of plausible configurations rather than over a single fitted parameter vector.

The workflow also provides a basis for identifying potential model inadequacy. If systematic discrepancies persist across observational contexts despite parameter adaptation, the simulator may lack relevant variables or physical processes. Such discrepancies could trigger a reassessment of the model structure, supported by additional observations or domain expertise, rather than further calibration of the same parameter space. These observations motivate the requirements introduced next.

3.3 Requirements of the model adaptation service

The workflow presented in Section 3.2 highlights two challenges for the adaptation service. Because simulator parameters are inferred indirectly from observed trajectories, candidate configurations must be evaluated through the behavior of the mechanistic model. Moreover, sparse observations may support several configurations with comparable agreement, even when their parameter values differ substantially. These considerations motivate two complementary requirements: grounding adaptation in forward simulation and preserving plausible parameterizations when the available evidence remains inconclusive.

Requirement 1 (R1): Simulation-based model adaptation. During the design or update phase of a digital twin of a natural system, model adaptation must remain grounded in the mechanistic simulator that supports downstream services. In the present setting, this involves calibrating simulator parameters against observations

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

from the target context to identify configurations consistent with observed behavior. Such calibration contributes to the broader validation of the digital twin, which also considers the model's fidelity over time and its suitability for forecasting, what-if analysis, and decision support. Identifying these configurations is particularly challenging in natural-system models, where the effects of many parameters are correlated and expressed through only partially observed dynamics.

This is particularly important in natural-system models, where the effects of many parameters are correlated and expressed through only partially observed dynamics. The contribution of any single parameter may then be difficult to isolate, since measured outputs reflect the combined action of several interacting processes over time. Under these conditions, candidate parameterizations are evaluated through forward simulation, using their agreement with observed trajectories to guide calibration. This grounds the adaptation process in the mechanistic simulator while retaining alternative configurations that remain consistent with the available evidence.

Requirement 2 (R2): Preservation of plausible parameterizations. Model adaptation in natural systems must avoid collapsing unresolved ambiguity into a single calibrated solution. A single retained calibration may appear well supported while masking other parameterizations that are equally consistent with the observed system behavior, and better reflect the dynamics of the natural twin. Thus, the objective of the adaptation service should not be limited to selecting one best-fitting configuration, but should aim to preserve multiple plausible simulator parameterizations that remain compatible with the target context. Making an early commitment to one model configuration would hide epistemic uncertainty that has not been resolved by the evidence. In a digital twin, that uncertainty matters for downstream services, since forecasting, what-if analysis, and decision support may depend on which plausible parameterization is retained. Preserving several compatible parameterizations allows the twin to quantify uncertainty over model adaptation and make it available to downstream services, whose sensitivity to that uncertainty may differ across monitoring, predictive, and prescriptive tasks.

Taken together, these requirements define adaptation not as the recovery of a single calibrated parameter vector, but as the identification and maintenance of plausible simulator configurations supported by observed system behavior. This view motivates the generative approach developed in the next section.

4 GFlowNet-based model adaptation

The previous section established two requirements for model adaptation in digital twins of natural systems: adaptation must remain grounded in the mechanistic simulator, and it must preserve multiple plausible parameterizations when the available evidence does not justify a unique calibration. We address these requirements by casting model adaptation as a generative modeling problem over simulator configurations. Section 4.1 develops this perspective, Section 4.2 specializes it to GFlowNets, and Section 4.3 presents the corresponding training and inference procedure.

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

4.1 Generative modeling for model adaptation

Standard simulation-based inference methods are often framed as posterior-estimation problems: given observations, the objective is to recover a distribution over parameters that is consistent with the simulator and the data. In the present setting, the adaptation service must produce complete simulator configurations that can be simulated, compared, and subsequently used by downstream digital twin services. This becomes particularly important when the relation between observed behavior and simulator parameters is underdetermined, so that several disconnected or structured regions of parameter space may remain plausible.

A generative modeling perspective addresses this need by learning to sample configurations from a reward-defined distribution rather than by explicitly estimating a Bayesian posterior. Here, generative modeling refers to learning a model that can construct complete simulator configurations and sample them from a distribution over plausible parameterizations. The mechanistic simulator remains the object being adapted, and candidate configurations are still evaluated through their simulated agreement with observations. The generative model does not replace the mechanistic structure with a purely statistical surrogate. Its role is to represent and sample plausible simulator configurations in a form that preserves uncertainty over adaptation while remaining directly usable for downstream digital twin services.

Among generative models, GFlowNets are particularly well suited to the present problem because they learn to construct structured objects while sampling them approximately in proportion to a reward function. In the present setting, that reward is derived from agreement between simulated and observed system behavior, which makes GFlowNets a natural candidate for representing and exploring the space of plausible simulator parameterizations.

4.2 GFlowNet model for parameter inference

Our GFlowNet defines a stochastic policy over the discrete space of candidate simulator configurations, so that complete parameterizations are generated with probability proportional to a positive reward. In the present setting, terminal states correspond to complete simulator parameterizations, and the learned policy acts as a generative adaptation mechanism over mechanistic model configurations. The resulting object is not a single calibrated parameter vector, but a sampler over plausible simulator parameterizations whose probabilities are shaped by agreement with observed system behavior.

Let $\mathcal{X}$ denote the discrete space of simulator configurations induced by the perturbation scheme used for model adaptation. Each terminal state $x \in \mathcal{X}$ corresponds to a complete parameterization $\boldsymbol{\theta}(x)$ obtained from a reference configuration $\boldsymbol{\theta}_0$ through a sequence of bounded perturbations. A non-terminal state s_t represents a partial construction after t decisions, and a trajectory

$$\tau = (s_0, a_1, s_1, \ldots, a_T, s_T = x) \tag{2}$$

corresponds to the sequential construction of one full simulator configuration. In the setting considered here, actions are defined over mutually exclusive parameter groups,

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

and each action applies a discrete perturbation to one group. Parameter inference is thus formulated as a structured parameter-set construction problem.

The learnable space induced by this construction is illustrated in Fig. 3. Rather than selecting a complete simulator parameterization in a single step, the GFlowNet constructs it sequentially through a series of decisions over parameter groups. Partial states correspond to intermediate configurations in which only a subset of decisions has been made, whereas terminal states correspond to complete simulator parameterizations. This sequential construction defines the space explored by the adaptation policy. During training, probability mass is progressively concentrated along branches that lead to higher-reward parameterizations, while alternative branches remain accessible when they also correspond to plausible mechanistic configurations.

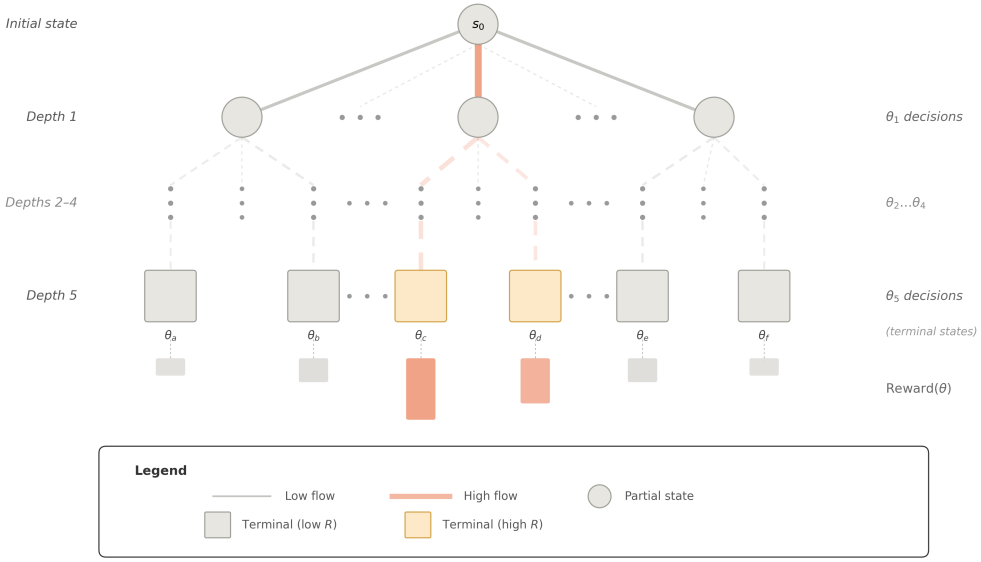


Fig. 3 Structured learnable space induced by the GFlowNet adaptation process. Starting from an initial state, the policy constructs a complete simulator parameterization through sequential decisions over parameter groups. Intermediate nodes represent partial configurations, and terminal nodes represent complete parameterizations.

Let $\mathcal{C} = \{1, \ldots, C\}$ denote the set of observational contexts used for adaptation, and let $\mathbf{y}_c^{\text{obs}}$ denote the observations associated with context $c \in \mathcal{C}$. For a candidate configuration x, the mechanistic simulator $\mathcal{M}$ produces

$$\mathcal{M}(\boldsymbol{\theta}(x), c) \mapsto \mathbf{y}_c^{\text{sim}}(x) \tag{3}$$

More precisely, let θ_t denote the current parameterization after t group-level decisions, with initial state θ_0 equal to the reference configuration. For a decision taken at

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

cycle c on group g_t with action a_t, each affected parameter p is updated according to

$$\theta_{t+1,p} = \mathrm{clip}\big(\theta_{t,p} + \eta_c \ \cdot sf \ \cdot \mathrm{sign}_{a_t,p}(x) \cdot (u_p - l_p),\, l_p,\, u_p\big)\,. \tag{4}$$

where $sf \in (0, 1]$ is the step fraction, $\mathrm{sign}_{a_t,p}(x) \in \{-1, 0, +1\}$ encodes the direction of the perturbation prescribed by the selected group–action pair for parameter p, and (l_p, u_p) are the admissible bounds of parameter p. The operator $\mathrm{clip}(\cdot, l_p, u_p)$ ensures that updated values remain within the valid parameter range. Parameters for which $\mathrm{sign}_p(x) = 0$ remain unchanged. In the multi-cycle setting, the effective update magnitude is annealed across cycles through $\eta_c = 2^{-(c-1)}$, so that later passes refine earlier decisions more conservatively.

For example, consider the *shift warm* action for the temperature-inhibition group in Table 1. This action applies coordinated perturbations to the simulator's temperature-response parameters, rather than modifying the greenhouse temperature itself. For each parameter p, the selected action determines whether its value increases, decreases, or remains unchanged. With $sf = 0.15$, the first-cycle step magnitude is 15% of the parameter's admissible range, $0.15(u_p - l_p)$, independently of its current value. In the second cycle, the step magnitude is halved to 7.5% of the admissible range, allowing finer adjustments to the configuration.

Adaptation quality is evaluated jointly across contexts rather than from a single trajectory. Thus, we define the contextual discrepancy

$$\ell_c(x) = D\big(\mathbf{y}_c^{\mathrm{sim}}(x), \mathbf{y}_c^{\mathrm{obs}}\big) \tag{5}$$

where $D(\cdot, \cdot)$ denotes the trajectory-level discrepancy used for calibration. In practice, D is computed from normalized residuals so that variables or time points with larger magnitudes do not dominate the adaptation criterion purely because of scale. The resulting contextual loss vector is

$$\boldsymbol{\ell}(x) = \big(\ell_1(x), \ldots, \ell_C(x)\big) \tag{6}$$

which summarizes how well one simulator configuration agrees with the observations in each context.

A difficulty in this setting is that raw contextual losses can be highly heterogeneous. Some contexts are systematically easier to fit than others, and the scale of the discrepancy may vary substantially across contexts and across the explored state space. If used directly, these differences can make the reward either too flat or overly concentrated, which weakens the ability of the GFlowNet to preserve meaningful diversity. To regularize this effect, each contextual loss is normalized through empirical quantiles:

$$\widetilde{\ell}_c(x) = \frac{\ell_c(x) - q_{\mathrm{lo}}^{(c)}}{q_{\mathrm{hi}}^{(c)} - q_{\mathrm{lo}}^{(c)} + \varepsilon} \tag{7}$$

where $q_{\mathrm{lo}}^{(c)}$ and $q_{\mathrm{hi}}^{(c)}$ denote lower and upper quantiles estimated from the distribution of losses in context c, and $\varepsilon > 0$ avoids numerical instability. The normalization in Eq. (7)

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

makes contextual losses more comparable and makes reward shaping less sensitive to context-specific scale effects.

Robustness across contexts is then incorporated directly into the adaptation criterion. Rather than aggregating the normalized losses in Eq. (7) by a simple mean alone, we sort them in decreasing order,

$$\widetilde{\ell}_{(1)}(x) \geq \widetilde{\ell}_{(2)}(x) \geq \cdots \geq \widetilde{\ell}_{(C)}(x) \tag{8}$$

and define a tail-sensitive score over the K worst contexts:

$$\mathcal{L}_{\text{tail}}(x) = \frac{1}{K} \sum_{j=1}^{K} \widetilde{\ell}_{(j)}(x) \tag{9}$$

The overall adaptation loss is then written as

$$\mathcal{L}(x) = (1 - \lambda) \frac{1}{C} \sum_{c=1}^{C} \widetilde{\ell}_c(x) + \lambda \, \mathcal{L}_{\text{tail}}(x) \tag{10}$$

where $\lambda \in [0, 1]$ controls the importance of robustness to hard contexts.

Eq. (10) makes the adaptation objective tail-risk-sensitive. That is, parameterizations are favored not only for good average fit, but also for avoiding poor performance on the worst-performing contexts [21]. This is useful here because digital twin models may need parameterizations that remain credible across several operating conditions rather than configurations that perform well only on average. This makes model adaptation more robust to parameterizations that perform poorly on a subset of contexts.

Following the standard GFlowNet formulation, the terminal reward defines a positive unnormalized target distribution over complete parameterizations. We instantiate this target in Boltzmann form from the contextual loss in Eq. (10):

$$R(x) = \exp\big(-\beta \, \mathcal{L}(x)\big) \tag{11}$$

so that parameterizations with lower risk receive exponentially higher reward, and $\beta > 0$ controls how strongly probability mass is concentrated around them.

Under this construction, the GFlowNet learns to sample complete parameterizations with frequency proportional to the reward in Eq. (11). This proportionality biases sampling toward the best-performing regions of the reward landscape, while still assigning probability mass to alternative plausible configurations according to their relative quality.

This formulation differs from point calibration in a crucial way: the objective is not to identify one parameterization that minimizes a global discrepancy, but to learn a generative policy over complete mechanistic configurations whose sampling probabilities reflect both fit and robustness across contexts. In this way, the GFlowNet satisfies **R1** by keeping inference grounded in the mechanistic simulator, while operationalizing **R2** through explicit generation of multiple plausible simulator configurations.

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

4.3 Training and inference procedure

Training proceeds by repeatedly sampling trajectories in the GFlowNet state space, evaluating the terminal parameterizations reached by those trajectories, and updating the forward policy so that terminal sampling frequencies align with the reward-defined target distribution. For a terminal state x, this requires simulating the mechanistic model across the observational contexts in Eq. (3), computing the contextual discrepancies in Eq. (5), normalizing them according to Eq. (7), and aggregating them into the tail-risk-sensitive adaptation loss defined in Eqs. (9) and (10). The resulting score is then transformed into the terminal reward through the Boltzmann construction in Eq. (11). Simulator execution is the most expensive component of the training pipeline, since it provides the trajectory-level evidence from which terminal reward is constructed.

Depending on the size of the terminal state space, reward construction can be handled in two ways. When the set of complete parameterizations is small enough to enumerate, simulator outputs, contextual losses, and terminal rewards can be computed offline and cached for all terminal states. In larger settings, these quantities are obtained on demand when terminal states are encountered during training. In both cases, the optimization problem remains the same: the GFlowNet is trained so that its terminal distribution matches the reward-induced target distribution over complete simulator configurations.

Let $P_F(\tau)$ and $P_B(\tau)$ denote the forward and backward trajectory probabilities associated with a complete construction trajectory τ in Eq. (2), ending in terminal state x. The model is trained with the trajectory balance objective [22], which relates trajectory probabilities and terminal rewards through

$$\log Z + \log P_F(\tau) - \log P_B(\tau) \approx \log R(x), \tag{12}$$

where Z is the normalizing constant of the target distribution. In practice, training minimizes the squared trajectory balance residual associated with Eq. (12) over sampled trajectories. In the discrete perturbation setting considered here, the construction graph is a tree, since each complete parameterization is reached by a unique sequence of decisions. Each non-initial state has a single parent, which makes the backward policy deterministic. As a result, $P_B(\tau)$ is fixed and resolves to one, thus no learned backward policy is required. The training signal is determined solely by the forward construction policy, the terminal reward in Eq. (11), and the learned normalizing constant.

Operationally, the proposed mechanism separates adaptation episodes from subsequent sampling. During an adaptation episode, contextual observations are used to evaluate candidate simulator configurations and train the generative policy. Once trained, this policy can be queried to generate multiple plausible parameterizations on demand without repeating the full adaptation procedure. As discussed in Section 3.1, such episodes may be initiated when new observations become available, operating conditions change, or discrepancies indicate that the current simulator is no longer adequate. The present study evaluates the adaptation and sampling mechanism itself, rather than implementing the runtime detection of adaptation triggers or

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

the continuous synchronization of an operational digital twin. These functions remain responsibilities of the surrounding digital-twin architecture into which the proposed service can be integrated.

5 Evaluation

To assess the proposed generative model adaptation mechanism, the experimental study is organized around the following research questions:

- **RQ1 - Mode Discovery**: to what extent can the proposed GFlowNet recover distinct high-quality regions of the adaptation landscape, thereby enabling the generation of diverse and plausible simulator parameterizations under sparse observations?
- **RQ2 - Retrieval Performance**: how effectively does the proposed approach retrieve strong adaptation hypotheses, and how does it compare with baseline search methods under matched simulator-evaluation budgets?
- **RQ3 - Scalability**: how does the proposed approach behave as the adaptation problem moves from an enumerable regime to larger and more computationally demanding settings, and what practical limitations emerge for real-system model adaptation?

The rest of this section is organized as follows: Section 5.1 presents the case studied in our experiments, Section 5.2 details our experimental setup, Section 5.3 showcases our results, Section 5.4 discusses the relevance of our approach to the requirements presented in Section 3.3, and Section 5.5 describes threats to validity.

5.1 Case Study

Controlled environment agriculture refers to crop production systems in which key environmental variables such as temperature, light, humidity, CO_2, irrigation, and nutrient supply are actively regulated in order to shape plant development and resource use. Greenhouse horticulture is one of the most established forms of Controlled environment agriculture, and it provides a particularly relevant setting for model-based reasoning because crop performance emerges from the interaction between biological processes and tightly managed climate-control strategies. Controlled environment agriculture has also been recognized as a relevant setting for digital twin development, where models, observations, and decision-support services must be combined in a coherent operational framework [23]. These characteristics make greenhouse production an informative application domain for the present work: it is mechanistically rich, operationally data-intensive, and representative of the broader class of natural systems for which digital twins must integrate heterogeneous models and observations [3]. Recent reviews further suggest that smart farming remains a useful testbed for studying such model-based digital twin mechanisms in practice [24].

The empirical study is based on data from the second edition of the Autonomous Greenhouse Challenge, conducted in six high-tech glasshouse compartments at the Wageningen Research Centre in Bleiswijk, The Netherlands [25]. The production cycle spans approximately six months and concerns cherry tomato cultivation of cultivar

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

Axiany. Five compartments were operated remotely by multidisciplinary international teams, while a sixth compartment was managed manually by commercial Dutch growers and used as a reference. The objective of the challenge was to maximize net profit by balancing production and fruit quality against resource consumption, including water, nutrients, heating, electricity, and CO_2. For the purposes of the present study, we use the subset of the challenge data that is relevant to simulator adaptation, namely production data and environmental measurements collected under the different greenhouse control strategies.

This dataset is well suited to the present study because it exposes the adaptation mechanism to several observational contexts generated under different control policies. From the perspective of simulator calibration, each compartment provides a distinct realization of the same crop-production process under a different management regime. The inference problem is therefore not to match a single trajectory in isolation, but to identify simulator parameterizations that remain compatible with multiple observed trajectories collected under heterogeneous operating conditions. This setting is closely aligned with the motivation of the paper: sparse and noisy observations may support several plausible mechanistic explanations, so a useful adaptation mechanism should support not only good fit, but also the recovery of multiple strong hypotheses.

The mechanistic simulator used in our study is derived from the tomato yield model of Vanthoor et al., originally developed for model-based greenhouse design [8]. The model describes tomato production as a function of indoor climate, with particular emphasis on the effects of temperature, light, and CO_2 concentration on yield. Its internal structure is organized around a carbohydrate buffer that mediates assimilate allocation to fruits, leaves, and stems, while temperature-dependent growth inhibition functions account for both instantaneous and mean temperature effects on crop production. This makes it a suitable mechanistic basis for the present experiments: the model remains physiologically interpretable, yet can be simulated repeatedly under many candidate parameterizations.

5.2 Experimental Setup

This subsection describes the calibration problem, the compared methods, and the evaluation protocol used throughout the empirical study. The experiments are designed to assess the three research questions introduced above, namely mode discovery, retrieval performance under matched simulator-evaluation budgets, and scalability as the adaptation space becomes too large for exhaustive analysis.

5.2.1 Crop Simulator and Observation Protocol

In the present study, adaptation is carried out over six observational contexts, each corresponding to one greenhouse compartment from the dataset introduced in the case study. For each candidate parameterization, the simulator is executed independently on all six observed contexts. The resulting simulated trajectories are compared with the corresponding observations. Rather than matching the full simulated time series, the adaptation signal is constructed from a finite set of observation points extracted from each trajectory. These points are concentrated near the end of the simulation horizon,

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

where accumulated differences between parameterizations become most informative for the crop-production variables of interest. In practice, the observations correspond to crop yield dry mass measured according to the greenhouse collection schedule, namely once every two weeks.

The six context-specific discrepancies are then combined into the contextual loss defined in Section 4.2. A parameterization is considered good only to the extent that it provides coherent agreement across the full observation set rather than fitting one context in isolation. This makes the adaptation problem more constrained than single-trajectory fitting while preserving the possibility that several distinct parameterizations remain plausible under sparse observations. The rationale is that adapted simulator parameters should capture crop-growth variation across contexts rather than overfit a single greenhouse trajectory.

5.2.2 Parameter Grouping and Search-Space Construction

Calibration is performed by perturbing a baseline simulator parameterization rather than by estimating all simulator parameters freely and independently. The chosen baseline parameterization corresponds to the parameters derived by Vanthoor et al. [8] for their model. The adjustable parameters are organized into five mutually exclusive groups obtained from a physically motivated functional decomposition. More specifically, this decomposition follows a physical interpretation of the growth process encoded by the simulator: incoming radiation first interacts with the canopy, which affects canopy temperature, absorption, and subsequent physiological processes that ultimately determine growth and yield. The groups are defined so that each one corresponds to a coherent component of this progression rather than to an arbitrary subset of coordinates. As a result, perturbations correspond to coherent model modifications rather than arbitrary coordinate changes. Table 1 summarizes this grouping.

This grouping serves two purposes in the experiments. First, it preserves interpretability by ensuring that each decision can be related to a meaningful component of the mechanistic model. Second, it induces a structured discrete search process in which complete parameterizations are built progressively through a sequence of group-level perturbations. Each state in the search process represents a partially specified simulator parameterization. Starting from a reference configuration, an action applies a discrete perturbation to one parameter group and updates the current model instance accordingly. Because the groups are mutually exclusive, each decision modifies exactly one component of the decomposition and does not overlap with the effects of previous decisions on other groups. A terminal state corresponds to a complete calibration candidate obtained after all required group-level perturbations have been assigned.

Several state-space configurations are considered in order to study both calibration quality and scalability. In the smallest configuration, each parameter group is perturbed once, yielding an enumerable state space of 2625 terminal states. We refer to this baseline as the 1-cycle setting. This makes it possible to precompute the complete calibration landscape and to compare the learned GFlowNet distribution against the exact reward-defined target distribution. Larger configurations are obtained by allowing repeated perturbation cycles over the same five groups. This preserves the

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

Order	Group	# Params.	Possible perturbations	Vanthoor model parameters
1	Leaf and canopy geometry	3	none, increase, decrease	LAI_max, SLA, n_plants
2	Photosynthetic potential	9	none, increase, decrease, higher_sensitivity, lower_sensitivity	J_max_leaf, Jpot_activation, Jpot_deactivation, Jpot_entropy, alpha, deg_curv_elec_transport, Tcan_CO2_comp_point, CO2_air_stomata, net_ass_rate
3	Temperature inhibition	8	none, shift_warm, shift_cold, widen_optimum, narrow_optimum	k_sw_min_Tcan, s_min_Tcan, k_sw_max_Tcan, s_max_Tcan, k_sw_min_Tcan24, s_min_Tcan24, k_sw_max_Tcan24, s_max_Tcan24
4	Temperature and development	7	none, increase, decrease, higher_sensitivity, lower_sensitivity	bias_g_Tcan24, slope_g_Tcan24, TS_start, TS_end, c_dev1, c_dev2, r_fruit_Set
5	Biomass growth and maintenance	12	none, more_fruit_growth, more_veg_growth, lower_resp_cost, higher_resp_cost, higher_sensitivity, lower_sensitivity	G_max, c_fruit_growth, c_leaf_growth, c_stem_growth, c_fruit_maintenance, c_leaf_maintenance, c_stem_maintenance, Q_10_maintenance, rg_fruit, rg_leaf, rg_stem, c_rgr

Table 1 Grouped model parameters used to construct the discrete adaptation space. Each group is associated with a finite set of canonical perturbations that define coordinated updates over the parameters in that group. The grouping follows the ordered decomposition implemented in the perturbation scheme, from canopy-level traits to biomass growth and maintenance.

grouped construction principle while increasing the combinatorial size of the terminal-state space, eventually reaching a regime in which exhaustive enumeration is no longer feasible. In the experiments, this larger regime is represented by a 2-cycle setting with 4.8×10^6 possible states, and is used to assess the scalability and limitations of our proposed approach under more demanding combinatorial state-spaces.

5.2.3 Compared Methods and Implementation Details

The primary method under study is a GFlowNet trained over the grouped discrete perturbation space described above. The forward policy is implemented as a multilayer perceptron with three hidden layers of width 256 and is trained with the trajectory balance objective. Optimization uses a learning rate of 5×10^{-4}. Because the grouped construction graph is a tree, each terminal state is reached by a unique action sequence, so the backward trajectory probability is fixed to 1 and no learned backward policy is required.

The reward used for training follows the formulation introduced in Section 4.2. For each terminal state, the simulator is evaluated independently on the six observational contexts, the resulting contextual discrepancies are quantile-normalized, and the normalized losses are aggregated into the tail-risk-sensitive adaptation loss $\mathcal{L}(x)$. In the present experiments, the tail component $\mathcal{L}_{tail}(x)$ is defined from the two worst-performing contexts, and the robustness weight is fixed to $\lambda = 0.25$. Terminal reward is then obtained through the Boltzmann construction $R(x) = \exp(-\beta\mathcal{L}(x))$, so that the concentration of probability mass around low-loss regions is controlled by the parameter β.

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

Two experimental factors are varied systematically. The first is the grouped perturbation magnitude, controlled through the step fraction, for which we consider $sf = 0.15$ and $sf = 0.3$, corresponding respectively to finer and coarser grouped moves in parameter space. The second is the reward concentration parameter β. In the 1-cycle setting, we evaluate $\beta \in \{2, 4, 8\}$. In the 2-cycle setting, we retain the best 1-cycle value, $\beta = 4$, since a full ablation becomes prohibitively expensive once simulator evaluations are no longer cached. For repeated perturbation cycles, the effective perturbation magnitude is annealed across cycles by halving the update size at each pass over the parameter groups. This preserves the grouped construction principle while allowing later cycles to refine earlier decisions more conservatively.

Training budgets are adjusted to reflect the computational difference between the enumerable and non-enumerable settings. In the 1-cycle setting, the full reward landscape can be precomputed and cached, so training scales well and could in principle be extended further; in practice, models are trained for 1000 steps with a batch size of 16. In the 2-cycle setting, simulator evaluations become the main bottleneck, so training is limited to 2000 steps as a compromise between convergence and wall-clock cost. This larger setting is therefore used not only to test scalability, but also to expose the practical limits of the current adaptation mechanism under realistic simulation-evaluation constraints.

Two search-based methods are used to assess retrieval performance and scalability within the same grouped discrete adaptation space. The first is random search over the grouped perturbation structure. The second is Optuna's Tree-structured Parzen Estimator [26, 27] (TPE), chosen as an adaptive baseline for structured discrete optimization. Both methods can be evaluated directly using the same simulator-based adaptation loss as the GFlowNet. For retrieval comparisons, methods are evaluated under budgets that reflect the computational regime being studied. In the enumerable 1-cycle setting, comparisons are made under matched sampling budgets, since accounting for GFlowNet training cost would allow adaptive baselines to cover a large fraction of the finite search space directly. In the 2-cycle setting, where simulator evaluation becomes the dominant bottleneck, methods are compared under matched simulator-evaluation budgets to assess scalability.

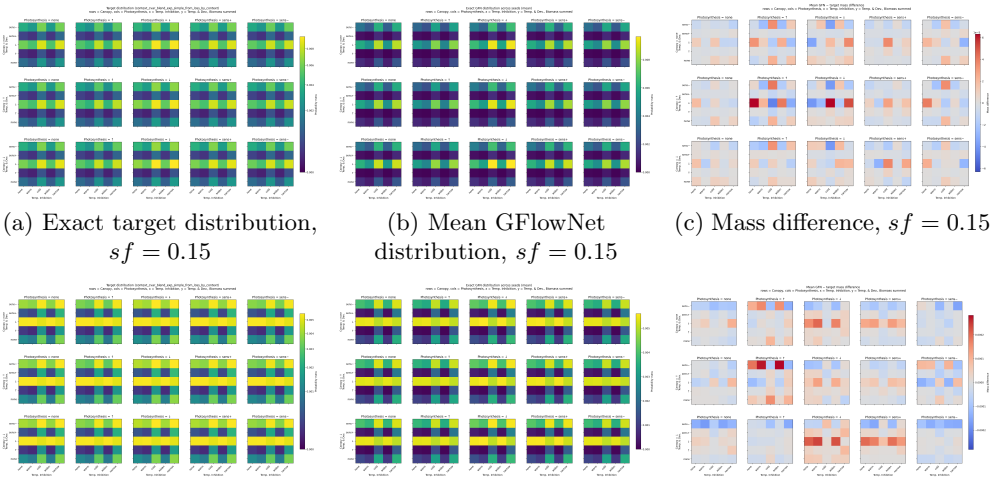
5.3 Results

This subsection reports the empirical results with respect to the three research questions introduced in Section 5. We begin with mode discovery in the enumerable 1-cycle regime, where the exact reward landscape makes it possible to compare the learned GFlowNet distribution against the target landscape and its dominant basins. We then examine retrieval performance under fixed budgets through ranked distributional comparisons, top-k recovery, and best-so-far behavior. Finally, we turn to the larger 2-cycle regime, where exhaustive analysis is no longer feasible and evaluation focuses instead on retrieval quality, diversity among strong retrieved candidates, and qualitative agreement with the observed trajectories. Together, these results assess whether the proposed generative adaptation mechanism can recover high-quality regions of the adaptation space, retrieve strong calibration hypotheses, and remain useful as simulator-evaluation costs increase.

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

5.3.1 Mode discovery

RQ1 examines whether the proposed GFlowNet recovers distinct high-quality regions of the adaptation landscape, thereby enabling the generation of diverse and plausible simulator parameterizations under sparse observations. This question is studied in the enumerable 1-cycle regime, where the exact reward landscape is available and can be used as ground truth. We focus here on the representative setting $\beta = 4$, and compare two step fractions, $sf = 0.15$ and $sf = 0.3$, in order to show how the structure of the adaptation landscape changes with the granularity of the grouped perturbations.



(a) Exact target distribution, $sf = 0.15$

(b) Mean GFlowNet distribution, $sf = 0.15$

(c) Mass difference, $sf = 0.15$

(d) Exact target distribution, $sf = 0.3$

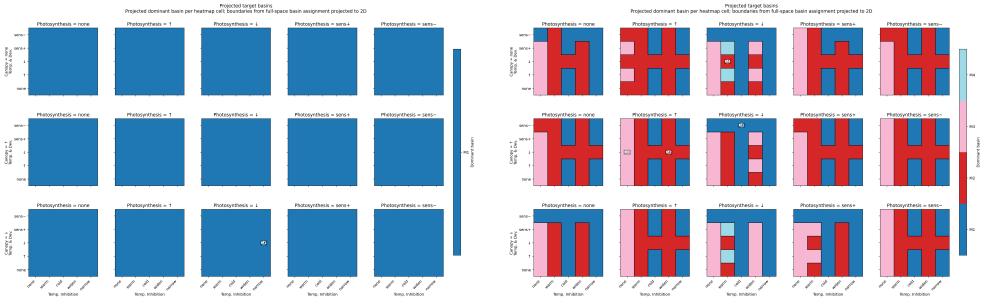
(e) Mean GFlowNet distribution, $sf = 0.3$

(f) Mass difference, $sf = 0.3$

Fig. 4 Mode-discovery analysis in the enumerable 1-cycle setting for $\beta = 4$ under two step fractions. Panels (a)–(c) correspond to the finer perturbation regime $sf = 0.15$, while panels (d)–(f) correspond to the coarser regime $sf = 0.3$. In each row, the left panel shows the exact target distribution, the middle panel shows the mean GFlowNet terminal distribution across seeds, and the right panel shows the difference between the two.

Fig. 4 compares the exact target distribution, the mean GFlowNet terminal distribution across seeds, and the corresponding mass difference for the two step fractions. In both settings, the learned policy recovers the overall structure of the exact target distribution and places mass on the same dominant high-reward regions. We observe that the GFlowNet tends to concentrate slightly more mass on the highest-reward areas of the landscape, while assigning less mass to lower-reward states. This effect is limited in the $sf = 0.15$ setting, where the target landscape is largely dominated by a single basin, and remains localized in the $sf = 0.3$ setting, where the landscape becomes genuinely multimodal. Overall, the learned distribution closely tracks the exact target while exhibiting a mild preference for its most rewarding regions.

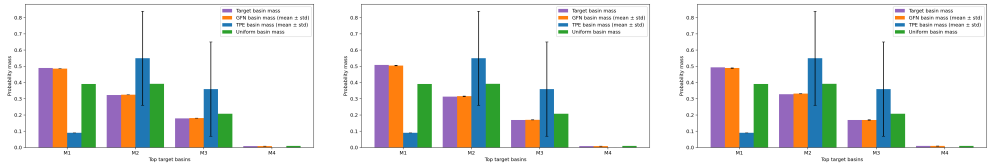
Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)



(a) Basin map for $sf = 0.15,\ \beta = 4$ (b) Basin map for $sf = 0.3,\ \beta = 4$

Fig. 5 Comparison of basin structure in the enumerable 1-cycle setting for $\beta = 4$ under two step fractions. Basins are obtained from the exact enumerable landscape by assigning each terminal state to the local mode reached by ascent on the full 1-cycle neighborhood graph. The projected maps show, for each displayed cell, the dominant basin contributing the largest share of mass in that region.

Modes are showcased more explicitly in Fig. 5. Here, basins are not defined directly in the projected heatmaps. Instead, they are obtained from the exact enumerable 1-cycle landscape by representing terminal states as a neighborhood graph and assigning each state to the local mode reached by repeated ascent toward higher target probability. Basin mass is then obtained by summing the probability mass of all member states assigned to that mode. The basin comparison shows that the finer step fraction yields a landscape dominated by a single basin, whereas the coarser step fraction reveals four dominant basins. This difference helps explain why the $sf = 0.3$ setting is the most informative one for evaluating mode discovery.



(a) Basin masses for $sf = 0.3,\ \beta = 2$ (b) Basin masses for $sf = 0.3,\ \beta = 4$ (c) Basin masses for $sf = 0.3,\ \beta = 8$

Fig. 6 Comparison of probability mass assigned to the dominant target basins in the enumerable 1-cycle setting for $sf = 0.3$ under three reward sharpness settings. From left to right, the panels show $\beta = 2$, $\beta = 4$, and $\beta = 8$.

The basin-level comparison in Fig. 6 shows that the proposed approach consistently recovers the dominant target modes across all three reward sharpness settings. In other words, mode discovery is not restricted to a particular choice of β. The overall concentration of probability mass across the principal basins remains broadly similar from $\beta = 2$ to $\beta = 8$, which suggests that changing the reward sharpness has only a limited effect on the basin-level allocation itself. What remains stable across these settings is that the GFlowNet places mass on the same dominant basins as the exact target distribution, rather than collapsing onto a single optimum or dispersing

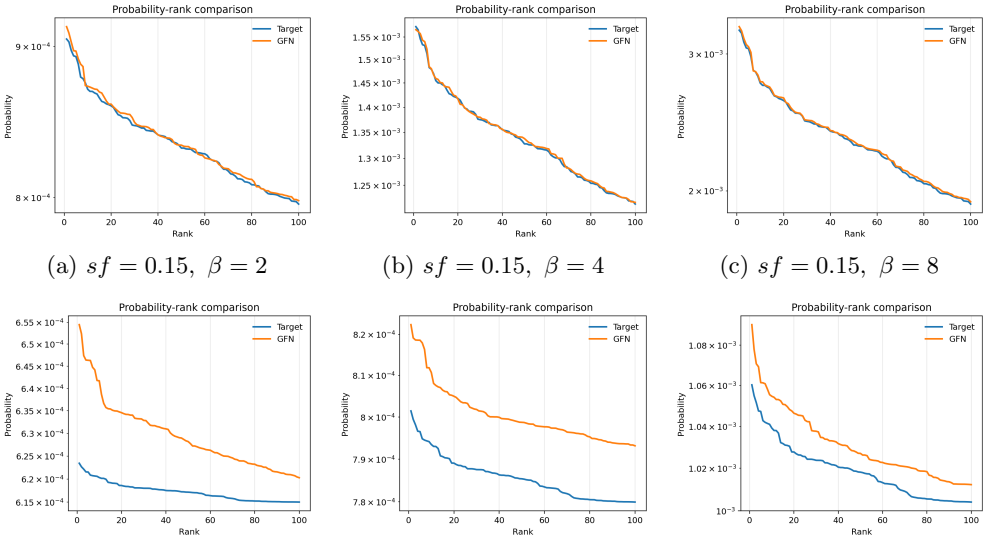
Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

mass arbitrarily. This provides further evidence that the learned policy captures the principal mode structure of the adaptation landscape.

Taken together, these results show that the proposed approach preserves multiple plausible high-quality parameterizations rather than collapsing onto a single solution. This is especially clear when the grouped perturbation space induces a genuinely multimodal target distribution, as in the $sf = 0.3$ setting, where the GFlowNet recovers the dominant basins of the exact landscape. In that sense, these results provide evidence for **R2**: the learned policy does not merely identify one strong calibration, but maintains several plausible regions of the parameter space from which diverse simulator configurations can be sampled.

5.3.2 Retrieval performance

RQ2 investigates whether the learned GFlowNet policy is useful not only as a generative model of the adaptation landscape, but also as a practical retrieval mechanism for high-quality simulator parameterizations. This question is again studied in the enumerable 1-cycle regime, where the exact target distribution is available and can be used as a reference. We evaluate retrieval from two complementary perspectives. First, we compare the ranked probability profiles induced by the learned and exact distributions in order to assess whether the GFlowNet preserves the global allocation of mass across terminal states. Second, we measure retrieval quality directly through top-k recovery, best-so-far performance, and qualitative overlays against the observed dry-mass measurements.



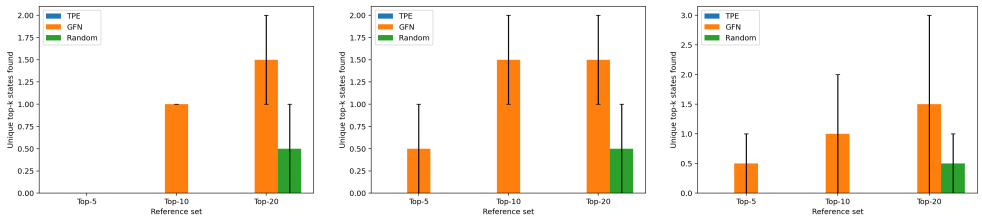
(a) $sf = 0.15,\ \beta = 2$ (b) $sf = 0.15,\ \beta = 4$ (c) $sf = 0.15,\ \beta = 8$

(d) $sf = 0.3,\ \beta = 2$ (e) $sf = 0.3,\ \beta = 4$ (f) $sf = 0.3,\ \beta = 8$

Fig. 7 Probability-rank comparison between the exact target distribution and the learned GFlowNet distribution in the enumerable 1-cycle setting under two step fractions and three reward sharpness settings. In each panel, states are ordered by decreasing probability and the ranked profiles of the two distributions are compared directly.

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

Fig. 7 shows that the learned distribution remains closely aligned with the ranked probability structure of the exact target distribution across all six settings. This indicates that the GFlowNet preserves not only the dominant modes identified in RQ1, but also the broader allocation of mass across high-quality terminal states. For $sf = 0.15$, this agreement is strong for all three values of β, and the learned distribution closely matches the exact target distribution. For $sf = 0.3$, the match becomes less exact, which suggests that the learned normalization constant Z shifts the overall allocation of probability mass. However, even in that case the proportionality to the underlying reward landscape is preserved. The ranked comparisons show that the learned policy remains globally consistent with the exact target distribution, with the main deviations appearing at the level of normalization rather than in the ordering of high-quality states. As a result, the highest-scoring candidates receive more probability mass under the learned distribution than they would under the exact target landscape, and are therefore sampled more often.



(a) $sf = 0.3,\ \beta = 2$ (b) $sf = 0.3,\ \beta = 4$ (c) $sf = 0.3,\ \beta = 8$

Fig. 8 Recovery of unique target top-k states in the enumerable 1-cycle setting for the multimodal $sf = 0.3$ regime under three reward sharpness settings.

This distributional agreement is reflected in retrieval behavior. Fig. 8 measures how many distinct target top-k states are recovered by each strategy in the representative multimodal $sf = 0.3$ setting. Across all three β values, the GFlowNet consistently recovers more unique target top-k states than the competing methods. By contrast, the tree-structured Parzen estimator (TPE) recovers little or none of the exact target top-k set in this setting, while random search recovers only a limited number of top states at larger k. This is important because it shows that the learned distribution is not merely similar to the target in aggregate. It is also useful in practice for retrieving states that belong to the exact high-quality subset of the landscape.

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

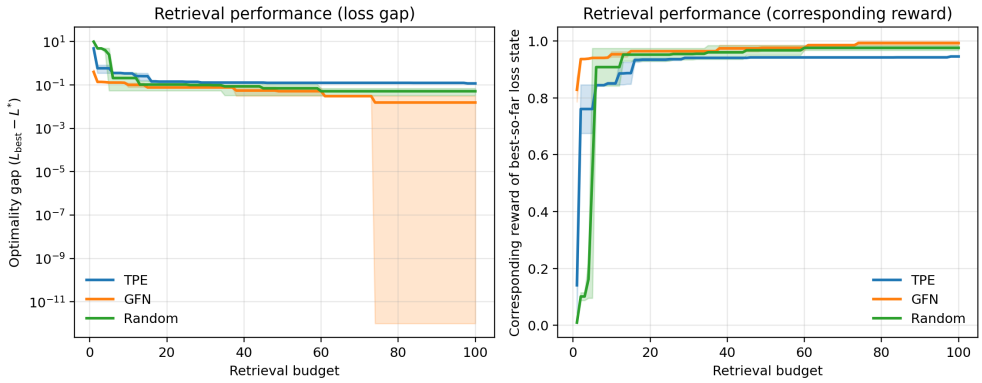
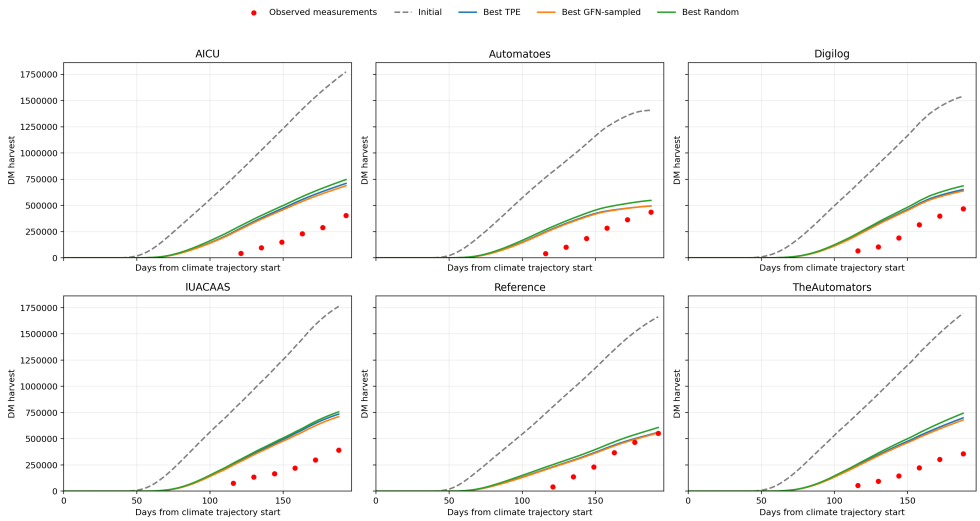
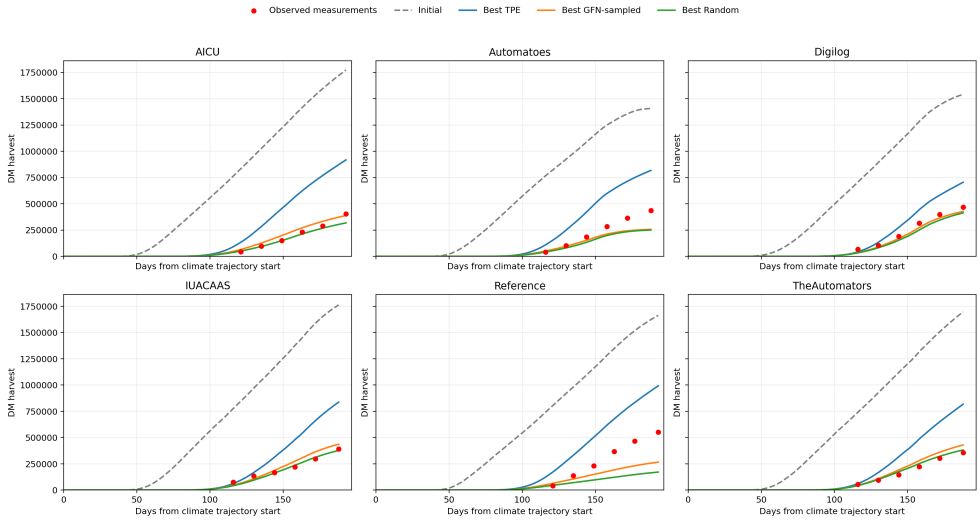


Fig. 9 Retrieval performance in the enumerable 1-cycle setting for $sf = 0.3$ and $\beta = 4$. The left panel reports the best-so-far optimality gap, while the right panel reports the corresponding reward of the best-so-far state. Curves are aggregated across seeds.

The retrieval curves in Fig. 9 provide a budget-aware view of the same phenomenon in the representative multimodal setting $sf = 0.3, \beta = 4$. In this setting, the GFlowNet reaches strong configurations rapidly and remains competitive throughout the evaluation budget. At the same time, random search also performs surprisingly well. This should be interpreted jointly with Fig. 7. The target distribution in this setting remains relatively flat, so many terminal states have similar losses. Combined with the size of the enumerable state space, this makes strong configurations comparatively easy to encounter by uninformed sampling alone, which in turn favors random search in the retrieval curves. The idea is not that random search reveals useful structure in the landscape, but that the learned policy remains competitive even in a regime where the landscape offers a comparatively weak separation between strong and mediocre candidates.



(a) Best-state overlay for $sf = 0.15,\ \beta = 4$



(b) Best-state overlay for $sf = 0.3,\ \beta = 4$

Fig. 10 Best-state overlays in the enumerable 1-cycle setting for the representative $\beta = 4$ case under two step fractions. For each team, observed dry-mass measurements are compared against the trajectories produced by the best retrieved state of each method and by the initial configuration.

Finally, Fig. 10 reconnects the retrieval results to the controlled environment agriculture setting. Rather than evaluating states only through ranked probabilities or loss values, these overlays show how the best configurations retrieved by each method reproduce the observed dry-mass measurements across teams. In both settings, the initial configuration remains clearly misaligned with the observations, while the retrieved states move the simulator trajectories toward the measured behavior. At $sf = 0.15$, the differences between retrieved trajectories remain relatively modest, whereas the

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

$sf = 0.3$ setting makes the contrast clearer: the retrieved configurations remain much closer to the observations than the initial state, while the adaptive baseline tends to overshoot cumulative dry mass more strongly. In that sense, the figure provides an interpretable bridge between state-space retrieval quality and the physical-system trajectories that motivate adaptation.

Taken together, these results show that the learned policy is not only capable of reproducing the dominant structure of the target distribution, but is also useful as a retrieval mechanism for strong simulator parameterizations. The close agreement in ranked probability profiles, the repeated recovery of target top-k states, and the improved best-so-far performance all indicate that the GFlowNet supports **R1** in practice: it remains sufficiently grounded in the adaptation landscape to retrieve parameterizations that are well aligned with the observations. At the same time, because this retrieval performance is achieved through a learned distribution rather than through a single deterministic calibration, the approach satisfies **R2**.

5.3.3 Scalability

RQ3 explores how the proposed approach behaves when the adaptation problem is scaled beyond the enumerable 1-cycle regime. To study this question, we consider the 2-cycle setting, where the number of reachable terminal configurations increases substantially and the exact target distribution is no longer available. In this regime, the evaluation focuses on retrieval quality, diversity among the strongest retrieved candidates, and qualitative agreement with the observed trajectories rather than on exact distributional fidelity. We restrict the analysis to $\beta = 4$. This choice was motivated by the need to keep the reward landscape smooth enough to remain conducive to learning while still sharp enough to retrieve high-reward samples.

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)



(a) Equal-budget comparison for $sf = 0.15,\ \beta = 4$



(b) Equal-budget comparison for $sf = 0.3,\ \beta = 4$

Fig. 11 Equal-budget comparison in the 2-cycle setting for the representative $\beta = 4$ case under two step fractions. Best loss and median top-20 loss evaluate the quality of the retrieved states, while the mean Hamming distance among the top retrieved states measures how broadly each method explores distinct high-quality regions of the search space.

Fig. 11 evaluates retrieval quality and diversity after the same simulation budget. The two loss-based quantities capture complementary aspects of retrieval quality. Best loss measures the quality of the single strongest state found by each method, whereas median top-20 loss indicates whether good performance is limited to one isolated configuration or extends to a broader set of strong candidates. The mean Hamming distance among the top retrieved states measures diversity: larger values indicate that the strongest retrieved candidates are distributed across more distinct regions of the search space rather than concentrated around near-duplicate solutions, since their keys differ by a greater amount.

Under this equal-budget comparison, the GFlowNet remains competitive with TPE on the quality measures while maintaining a substantially broader spread of high-quality candidates. At $sf = 0.15$, the two methods remain close in terms of best loss and median top-20 loss, but the Hamming distance clearly separates them, showing that the learned policy explores a more diverse set of promising configurations. At $sf = 0.3$, the GFlowNet becomes more favorable on the quality measures as well, while still preserving this broader diversity. This indicates that, even in the larger 2-cycle regime, the learned policy does not collapse onto a narrow subset of configurations, but continues to retrieve strong candidates from multiple regions of the parameter space.

At the same time, random search performs strongly in both settings, and especially in the coarser $sf = 0.3$ regime. This should not be read as evidence that uninformed sampling is intrinsically competitive with a learned retrieval policy, but rather as a consequence of the enlarged 2-cycle state space and of the limited simulator budget available to train the GFlowNet. In this regime, the learned policy can still retrieve states with lower loss than random search, but it no longer separates itself as clearly

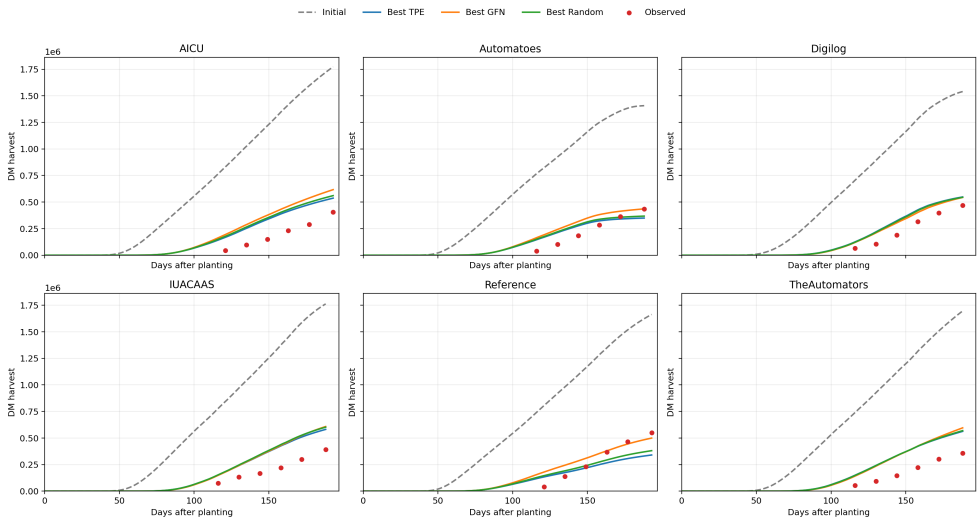
as in the enumerable 1-cycle setting because it has not amortized enough of the larger search space. This is reflected in the diversity results as well. Random search attains the largest Hamming distances, which is expected from an uninformed sampling strategy, while the GFlowNet achieves comparable diversity with consistently better loss values. Read in that light, the main result is not that random search reveals useful structure in the landscape, but that the learned policy remains competitive even though the enlarged search space makes it much harder to improve decisively over a flat reward landscape.

Step fraction	Method	Best loss	Median top-20 loss	Mean Hamming top-20	Training time
$sf = 0.15$	TPE	0.542 ± 0.003	$\mathbf{0.560 \pm 0.009}$	5.19 ± 0.31	$\mathbf{1{:}37{:}46 \pm 17{:}22}$
$sf = 0.15$	GFN	$\mathbf{0.539 \pm 0.002}$	0.561 ± 0.001	6.81 ± 0.12	$12{:}51{:}33 \pm 2{:}14{:}28$
$sf = 0.15$	Random	0.547 ± 0.006	0.709 ± 0.045	$\mathbf{7.41 \pm 0.06}$	—
$sf = 0.3$	TPE	0.595 ± 0.031	0.670 ± 0.027	6.39 ± 0.59	$\mathbf{1{:}46{:}29 \pm 18{:}13}$
$sf = 0.3$	GFN	$\mathbf{0.535 \pm 0.001}$	$\mathbf{0.611 \pm 0.012}$	7.61 ± 0.04	$8{:}42{:}14 \pm 3{:}43{:}30$
$sf = 0.3$	Random	0.570 ± 0.024	0.671 ± 0.022	$\mathbf{7.66 \pm 0.07}$	—

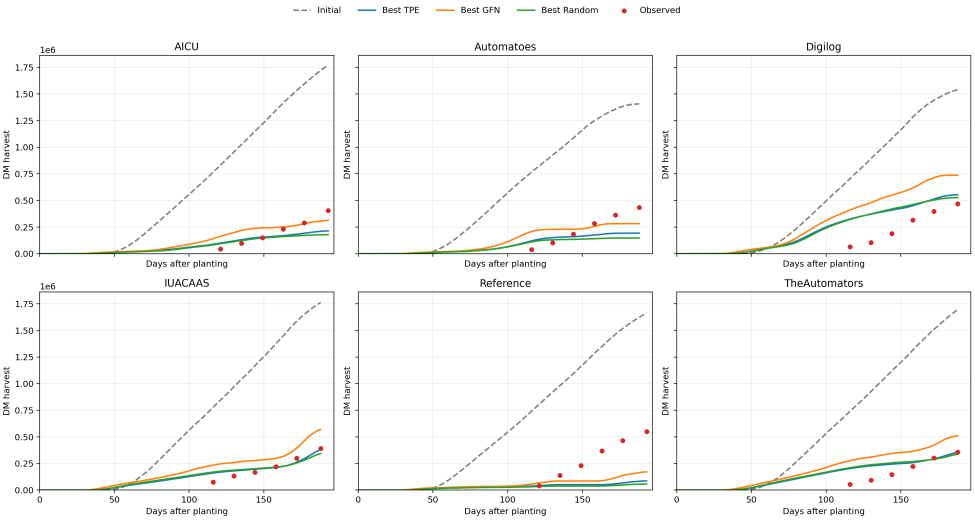
Table 2 RQ3 summary in the 2-cycle setting under an equal retrieval budget for $\beta = 4$. Lower is better for best loss and median top-20 loss; higher is better for mean Hamming top-20.

Table 2 highlights the main trade-off that emerges in the 2-cycle setting. At $sf = 0.15$, the GFlowNet only marginally improves upon TPE on best loss, while TPE attains a nearly identical median top-20 loss and fits about six times faster. At $sf = 0.3$, the distinction becomes clearer: the GFlowNet achieves the best values for both best loss and median top-20 loss, which indicates that it retrieves not only a strong single candidate, but also a stronger set of high-quality candidates under the same budget. The diversity column complements this comparison. Random search attains the largest Hamming distances in both settings, which is expected from an uninformed strategy that spreads samples broadly across the state space. More importantly, the GFlowNet maintains higher Hamming distances than TPE while remaining competitive, or favorable, on the quality measures. This suggests that the learned policy continues to cover several distinct promising regions of the search space rather than concentrating exclusively on one narrow optimum. At the same time, this broader coverage comes at a substantial computational cost: fitting the 2-cycle GFlowNet requires roughly 9 to 13 hours, whereas TPE typically completes in under 2 hours. During training, our GFlowNet caches candidates and their score, causing subsequent seeds to train faster, which explains the observed variance in training times.

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)



(a) Best-state overlay for $sf = 0.15,\ \beta = 4$



(b) Best-state overlay for $sf = 0.3,\ \beta = 4$

Fig. 12 Best-state overlays in the 2-cycle setting for $\beta = 4$ under two step fractions. For each team, observed dry-mass measurements are compared against the trajectories produced by the best retrieved state of each method and by the initial configuration.

Finally, Fig. 12 reconnects the scaling results to the greenhouse domain. Compared with the enumerable 1-cycle regime, the retrieved trajectories now display a much wider range of shapes across teams, which reflects the additional degrees of freedom introduced by the 2-cycle setting. In principle, this larger space should make better fits possible, since the simulator can express a broader set of cumulative dry-mass trajectories. In practice, however, this also makes the adaptation problem substantially harder to solve. In both settings, the retrieved configurations improve clearly over the

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

initial state, which remains visibly misaligned with the observations, but the differences between methods become more modest and more team-dependent than in the 1-cycle case. At $sf = 0.15$, many of the retrieved trajectories remain fairly close to one another, which is consistent with the small separation observed in the equal-budget comparison. At $sf = 0.3$, the contrast becomes clearer, but no single method dominates uniformly across all teams. This supports the broader interpretation of **RQ3**: scaling increases the expressive flexibility of the search space, but with simulation as the main bottleneck, identifying those better-fitting regions reliably becomes a much more difficult problem.

Taken together, these results show that the proposed approach continues to support both **R1** and **R2** when the adaptation problem is scaled to the 2-cycle regime, but that its practical effectiveness is currently limited by amortization cost. Under a matched retrieval budget, the GFlowNet remains competitive and continues to preserve a broad set of plausible candidate states, which is consistent with **R2**. At the same time, it remains capable of retrieving parameterizations that are well aligned with the observations, which supports **R1**, albeit in a lesser capacity than the 1-cycle case. However, the time required to fit the learned policy remains substantial, and this now constitutes the main performance bottleneck of the approach. In that sense, the 2-cycle results do not contradict the generative objective of the method. They show instead that further improvements in evaluation efficiency, batching, or surrogate support will be needed before amortized generative retrieval can be fully exploited at this scale.

5.4 Discussion

This study examined whether GFlowNet-based adaptation can satisfy **R1** and **R2** for digital twins of natural systems, where sparse and noisy observations often make simulator calibration underdetermined. The results suggest that the answer is positive in the enumerable 1-cycle regime, and only partially so at larger scale. In the 1-cycle setting, the learned policy recovered the dominant structure of the adaptation landscape, remained closely aligned with the exact target distribution, and retrieved parameterizations whose simulated trajectories matched the observations. In the 2-cycle setting, the same qualitative behavior persisted, but amortization costs grew substantially with the size of the search space.

The clearest evidence for **R2** comes from **RQ1**. In the enumerable regime, the learned policy did not collapse onto a single optimum and instead preserved several plausible high-quality regions of the landscape. This property is essential for digital twins of natural systems, where multiple simulator parameterizations may remain consistent with the available evidence. Preserving several plausible candidates is therefore not incidental to adaptation, but part of what makes it useful for downstream tasks such as monitoring, prediction, and prescription.

Conversely, RQ2 provides the clearest support for **R1**. In the 1-cycle setting, the ranked probability comparisons, top-k recovery results, retrieval curves, and best-state overlays all indicate that the learned policy remains sufficiently grounded in the adaptation landscape to recover parameterizations whose simulated trajectories are well aligned with the observations. This complements the interpretation of RQ1

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

directly. If RQ1 shows that the learned policy preserves the relevant structure of the landscape, RQ2 shows that this structure is not merely descriptive, but can be exploited to retrieve strong, observation-aligned configurations in practice.

RQ3 then shows how both conclusions change when we increase the possible state space. In the 2-cycle regime, the learned policy continues to retrieve strong candidates and to maintain a broader spread of promising configurations than TPE, which suggests that **R2** is retained at least in part in the enlarged search space. Support for **R1**, however, becomes more conditional. The learned policy remains competitive under a matched retrieval budget, but its separation from TPE and from random search is substantially reduced. We interpret this not as a breakdown of the formulation itself, but as evidence that the available simulator budget is insufficient to amortize the enlarged landscape to the point where the retrieval advantage remains decisive.

From the perspective of a greenhouse digital twin, these results illustrate how generative adaptation supports the maintenance of crop-growth simulators under different operating conditions. The trajectory comparisons in Figs. 10 and 12 show that the retrieved configurations reproduce observed crop dry-mass accumulation more closely than the initial simulator. Meanwhile, the mode-discovery and diversity results indicate that several distinct configurations may provide satisfactory agreement with the same observations. For growers and agronomists, retaining these alternatives could support forecasting and what-if analyses that account for uncertainty in simulator calibration, rather than relying on a single configuration. The benefits of this diversity for downstream decision-making remain to be evaluated.

These implications extend beyond tomato growth modeling, as the proposed generative formulation is not specific to a particular mechanistic simulator. However, its implementation depends on the simulator structure, parameter grouping, and available observations. Although the six greenhouse compartments represent different operating conditions, they remain instances of the same crop-production system. The results therefore establish feasibility in this setting, while generality across other mechanistic models remains to be assessed.

The main practical limitation observed in the experiments is computational. In the 2-cycle regime, the number of reachable terminal configurations increases substantially, making simulator evaluation the dominant bottleneck. As reported in Table 2, fitting the GFlowNet required many hours of training, yet this investment was insufficient to fully amortize the larger landscape. This computational cost also constrains how frequently adaptation could be performed as new observations become available. A second limitation concerns the induced reward landscape itself. When the landscape is relatively flat, many states achieve similar losses, which weakens the separation between informed and uninformed exploration and can make random search appear more competitive than under a sharper regime. This flatness also affects the learned sampler. Even under an exact match to the target distribution, high-reward states may remain rarely sampled due to the size of the state space. Reward-proportional sampling favors higher-quality configurations, but its effectiveness depends on the reward function, which must be carefully designed and validated for the target domain. As observed in RQ2, different values of β affected the concentration of the learned distribution. Adjusting this concentration can allocate more probability mass to top-ranked

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

states, supporting the retrieval of high-reward calibration hypotheses for downstream digital twin services.

Even with these limitations, the results remain encouraging for digital twins of natural systems. They show that a generative adaptation service can preserve multiple plausible parameterizations while still retrieving strong ones, which is precisely the combination required when observations are destructive, costly, or sparsely distributed in time. The main challenge exposed by the present study is therefore efficiency: progress at larger scale will depend on better batching, stronger caching strategies, or surrogate support that enables more effective amortization of larger search spaces.

5.5 Threats to validity

Internal validity. The reported results depend on several experimental choices. The adaptation landscape is induced by a specific grouped perturbation design, reward formulation, and simulator-based loss definition, all of which affect the relative difficulty of the retrieval problem for GFlowNet, TPE, and random search. In addition, the 1-cycle and 2-cycle regimes support different forms of evidence: the former benefits from an exact enumerable target distribution, whereas the latter can only be assessed through retrieval-based comparisons. The number of seeds and computational budgets also remains limited, and caching affects wall-clock time by reducing simulator effort unevenly across runs. Further, our experiments were performed on clusters with shared access across our institutions, thus computing power was not consistent throughout model training. Finally, flatter reward landscapes reduce the separation between strong and mediocre candidates, which can make random search appear more competitive than it would under a sharper target distribution.

External validity. The evaluation was conducted on a single mechanistic greenhouse simulator in controlled environment agriculture, with one grouped perturbation design, two step fractions, a narrow range of reward sharpness settings, and baselines centered on TPE and random search. The conclusions therefore support the feasibility of GFlowNet-based adaptation for digital twins of natural systems in this setting, but they do not establish that the same results will transfer unchanged to other simulators, crops, sensing regimes, or adaptation services. Broader generality will need to be assessed across other natural-system domains, other simulator structures, and larger search spaces where the computational bottlenecks identified here may manifest differently. The mechanism also assumes that the simulator structure and adjustable parameter space can represent the relevant physical processes. Persistent discrepancies across observational contexts may indicate omitted factors or structural inadequacy that parameter updates cannot resolve. Assessing such discrepancies would require examining absolute residuals against expected measurement variability and, where available, independent observations. These analyses could motivate additional measurements or structural model revision, but identifying missing factors is not addressed by the present implementation.

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

6 Related Works

This section situates the proposed approach with respect to three complementary lines of work. We first review model-driven engineering research on digital twins, focusing on lifecycle management, adaptation, and the role of model artifacts and services. We then discuss digital twins of natural and biological systems, where uncertainty, indirect observations, and evolving operating contexts make simulator adaptation particularly challenging. Finally, we examine simulation-based inference and recent generative approaches for likelihood-free parameter inference in mechanistic models, in order to clarify how the present work differs from existing posterior-estimation and calibration-oriented formulations.

6.1 Model-driven engineering for digital twins

Systematic engineering of digital twins has emerged as a distinct MDE research agenda, with early work outlining its foundational challenges around model heterogeneity, synchronization across abstractions, and lifecycle-wide coordination [28]. Recent synthesis work extends this agenda by aligning MDE capabilities with digital twin engineering needs across the lifecycle [10], by providing a systematic mapping of MDE techniques applied to digital twins [29], and by mapping the broader software engineering landscape [30]. On lifecycle management and self-adaptation, declarative formulations treat the twin and its physical counterpart as a coupled system evolving across lifecycle stages [31], while notational approaches reason about continuous digital twin evolution [32]. Our work contributes to this line of research by treating model adaptation as an explicit digital twin service and as the inference mechanism that refreshes the mechanistic simulator when new evidence justifies it, while locating this service within the design phase of the digital twin lifecycle.

6.2 Digital twins of natural and biological systems

Digital twins of natural systems remain less established but are directly relevant to our setting. Prior work in greenhouse-based cyber-biophysical systems identifies simulator construction, uncertainty handling, and adaptation as central challenges [3], and has produced self-adaptation architectures that distinguish behavioral from structural adaptation [33]. Our contribution addresses parametric adaptation within this behavioral layer, replacing pointwise parameter tuning with reward-proportional sampling over complete simulator configurations. Structural adaptation, such as revising model components or governing equations, remains outside the proposed mechanism.

In smart farming more broadly, control-oriented digital twin frameworks have been proposed [34], though recent reviews report that robust predictive capabilities remain limited in practice [14]. In biomedicine, digital twins have been grounded in probabilistic graphical models [9], with uncertainty quantification discussed as a central concern across the lifecycle [35]. Despite their domain differences, these systems share a common modeling difficulty: observations are typically partial, heterogeneous, and obtained at the level of system behavior rather than directly at the level of simulator parameters. Maintaining several plausible simulator configurations is thus

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

valuable because it preserves adaptation uncertainty in a form that can still support downstream prediction, what-if analysis, and decision-making.

6.3 Simulation-based inference and generative modeling

Calibration under intractable likelihoods is typically addressed through approximate Bayesian computation [36] or neural posterior estimation [37], as surveyed in [20]. Recent work applies these techniques directly to digital twins, using neural posterior estimation for amortized parameter inference in physiological digital twins [38] and formulating probabilistic digital twins through measure-theoretic constructions [39]. These approaches target posterior estimation over parameters and can likewise produce complete parameter samples. Our approach instead learns a generative policy that constructs simulator configurations through structured group-level decisions, with sampling shaped by a reward derived from simulation–observation agreement. Generative Flow Networks provide the formal basis for reward-proportional sampling of compositional objects, with mode-covering behavior that distinguishes them from reward-maximizing reinforcement learning and mode-seeking variational inference, and have been surveyed as tools for scientific discovery [40]. Established applications include Bayesian structure learning [17] and multi-objective optimization [16], with training stabilized through trajectory balance [22]. To our knowledge, prior work has not applied GFlowNets to mechanistic simulator adaptation within a digital twin setting.

The discrete construction space considered here does not preclude Bayesian inference. Approximate Bayesian computation [41] can approximate parameter posteriors by accepting or weighting simulated configurations according to their agreement with observations, while neural posterior estimation [42] learns conditional posterior distributions from simulated parameter–observation pairs. Comparing these approaches with the proposed reward-proportional sampler would require specifying a prior, defining a comparable Bayesian inference target, and accounting for simulator-evaluation and training costs. Given the sparse and indirect observations characteristic of our setting, establishing and validating such a probabilistic formulation remains an avenue for future work.

7 Conclusion

This work investigates model adaptation as a service within the lifecycle of digital twins of natural systems, with generative modeling as an inference mechanism. The objective was not simply to recalibrate a mechanistic simulator, but to maintain its operational usefulness as new evidence becomes available. To that end, the adaptation service was framed around two requirements: **R1**, the ability to retrieve parameterizations that remain well aligned with the available observations, and **R2**, the ability to preserve several plausible high-quality parameterizations rather than collapse prematurely onto a single solution. These requirements are especially important in natural systems, where observations are sparse, noisy, and often insufficient to identify one unambiguous explanation of the physical process.

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

Our results show that this objective is attainable when the parameterization is characterized by a small state space. The learned model policy recovered the dominant structure of the adaptation landscape, remained closely aligned with the exact target distribution, and retrieved parameterizations whose simulated trajectories matched the observations. In that setting, the method provided strong support for both **R1** and **R2**: it remained grounded enough to recover observation-aligned configurations while also preserving several plausible alternatives that can support downstream digital twin services.

Results on a larger parameter state-space showed that achieving better performance is limited by computational costs. Under a matched retrieval budget, the GFlowNet remained competitive and continued to preserve a broader spread of plausible candidates than baselines. However, the enlarged search space made amortization substantially more difficult, and simulator evaluation emerged as the dominant bottleneck. The main limitation identified by this study is thus not conceptual, but computational.

Taken together, our results suggest that GFlowNets are a promising foundation for model adaptation in digital twins of natural systems. They support both retrieval of observation-aligned parameterizations and preservation of plausible alternatives, which is precisely the combination required in underdetermined natural-system settings. Future work should therefore focus on improving scalability through more efficient simulator use.

References

[1] Zhou, J., Zhang, S. & Gu, M. Revisiting digital twins: Origins, fundamentals, and practices. *Frontiers of Engineering Management* **9**, 668–676 (2022).

[2] Javaid, M., Haleem, A. & Suman, R. Digital twin applications toward industry 4.0: A review. *Cognitive Robotics* **3**, 71–92 (2023).

[3] David, I. *et al.* Digital twins for cyber-biophysical systems: Challenges and lessons learned. 2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS) (2023). Pp. 1–12. IEEE. doi: 10.1109/MODELS58315.2023.00014.

[4] Katsoulakis, E. *et al.* Digital twins for health: a scoping review. *NPJ digital medicine* **7**, 77 (2024).

[5] Lin, L., Bao, H. & Dinh, N. Uncertainty quantification and software risk analysis for digital twins in the nearly autonomous management and control systems: A review. *Annals of Nuclear Energy* **160**, 108362 (2021).

[6] Gallo, L., Frasca, M., Latora, V. & Russo, G. Lack of practical identifiability may hamper reliable predictions in covid-19 epidemic models. *Science advances* **8**, eabg5234 (2022).

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

[7] Bengio, Y. *et al.* Gflownet foundations. *Journal of Machine Learning Research* **24**, 1–55 (2023).

[8] Vanthoor, B., De Visser, P., Stanghellini, C. & Van Henten, E. J. A methodology for model-based greenhouse design: Part 2, description and validation of a tomato yield model. *Biosystems engineering* **110**, 378–395 (2011).

[9] Kapteyn, M. G., Pretorius, J. V. R. & Willcox, K. E. A probabilistic graphical model foundation for enabling predictive digital twins at scale. *Nature Computational Science* **1**, 337–347 (2021).

[10] Michael, J. *et al.* Model-driven engineering for digital twins: Opportunities and challenges. *Systems Engineering* **28**, 659–670 (2025).

[11] de Koning, K. *et al.* Digital twins: dynamic model-data fusion for ecology. *Trends in ecology & evolution* **38**, 916–926 (2023).

[12] An, G. & Cockrell, C. Drug development digital twins for drug discovery, testing and repurposing: a schema for requirements and development. *Frontiers in systems biology* **2**, 928387 (2022).

[13] Pylianidis, C., Osinga, S. & Athanasiadis, I. N. Introducing digital twins to agriculture. *Computers and Electronics in Agriculture* **184**, 105942 (2021).

[14] Purcell, W. & Neubauer, T. Digital twins in agriculture: A state-of-the-art review. *Smart Agricultural Technology* **3**, 100094 (2023).

[15] Bengio, E., Jain, M., Korablyov, M., Precup, D. & Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. *Advances in neural information processing systems* **34**, 27381–27394 (2021).

[16] Jain, M. *et al.* Multi-objective GFlowNets. Proceedings of the 40th International Conference on Machine Learning (ICML) (2023). Proceedings of Machine Learning Research, Vol. 202, pp. 14631–14653. PMLR.

[17] Deleu, T. *et al.* Bayesian structure learning with generative flow networks. Uncertainty in Artificial Intelligence (2022). Pp. 518–528. PMLR.

[18] Hu, E. J. *et al.* Amortizing intractable inference in large language models. *arXiv preprint arXiv:2310.04363* (2023).

[19] Blair, G. S. Digital twins of the natural environment. *Patterns* **2** (2021).

[20] Cranmer, K., Brehmer, J. & Louppe, G. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences* **117**, 30055–30062 (2020).

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

[21] Rockafellar, R. T., Uryasev, S. *et al.* Optimization of conditional value-at-risk. *Journal of risk* **2**, 21–42 (2000).

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

[23] Eramo, R. *et al.* Conceptualizing digital twins. *IEEE Software* **39**, 39–46 (2022).

[24] Subeesh, A. & Chauhan, N. Agricultural digital twin for smart farming: A review. *Green Technologies and Sustainability* 100299 (2025).

[25] Hemming, S., de Zwart, F., Elings, A., Petropoulou, A. & Righini, I. Autonomous greenhouse challenge, second edition (2019) (2020). URL https://doi.org/10.4121/uuid:88d22c60-21b3-4ea8-90db-20249a5be2a7.

[26] Watanabe, S. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint arXiv:2304.11127* (2023).

[27] Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. Optuna: A next-generation hyperparameter optimization framework. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019). Pp. 2623–2631.

[28] Bordeleau, F., Combemale, B., Eramo, R., van den Brand, M. & Wimmer, M. Towards model-driven digital twin engineering: Current opportunities and future challenges. Systems Modelling and Management (ICSMM 2020) (2020). Communications in Computer and Information Science, Vol. 1262, pp. 43–54. Springer. doi: 10.1007/978-3-030-58167-1_4.

[29] Lehner, D. *et al.* Model-driven engineering for digital twins: a systematic mapping study. *Software and Systems Modeling* **24**, 1339–1377 (2025).

[30] Dalibor, M. *et al.* A cross-domain systematic mapping study on software engineering for digital twins. *Journal of Systems and Software* **193**, 111361 (2022).

[31] Kamburjan, E., Bencomo, N., Tapia Tarifa, S. L. & Johnsen, E. B. Declarative lifecycle management in digital twins. Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (2024). Pp. 353–363.

[32] Mertens, J., Klikovits, S., Bordeleau, F., Denil, J. & Haugen, Ø. Continuous evolution of digital twins using the dartwin notation. *Software and Systems Modeling* (2024).

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)

[33] Kamburjan, E. *et al.* GreenhouseDT: An exemplar for digital twins. Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (2024). Pp. 175–181.

[34] Verdouw, C., Tekinerdogan, B., Beulens, A. & Wolfert, S. Digital twins in smart farming. *Agricultural Systems* **189**, 103046 (2021).

[35] Thelen, A. *et al.* A comprehensive review of digital twin—part 2: roles of uncertainty quantification and optimization, a battery digital twin, and perspectives. *Structural and multidisciplinary optimization* **66**, 1 (2023).

[36] Sisson, S. A., Fan, Y. & Beaumont, M. A. (eds) *Handbook of Approximate Bayesian Computation* (Chapman and Hall/CRC, 2018).

[37] Greenberg, D. S., Nonnenmacher, M. & Macke, J. H. Automatic posterior transformation for likelihood-free inference. International Conference on Machine Learning (ICML) (2019). PMLR, pp. 2404–2414.

[38] Hoang, T.-D. *et al.* A real-time digital twin for type 1 diabetes using simulation-based inference. International Workshop on Digital Twin for Healthcare (2025). Pp. 35–46. Springer.

[39] Agrell, C., Rognlien Dahl, K. & Hafver, A. Optimal sequential decision making with probabilistic digital twins: Theoretical foundations. *SN Applied Sciences* **5**, 114 (2023).

[40] Jain, M. *et al.* Gflownets for ai-driven scientific discovery. *Digital Discovery* **2**, 557–577 (2023).

[41] Sunnåker, M. *et al.* Approximate Bayesian Computation. *PLOS Computational Biology* **9**, e1002803 (2013).

[42] Gutmann, M. U. & Corander, J. Bayesian optimization for likelihood-free inference of simulator-based statistical models. *Journal of Machine Learning Research* **17**, 1–47 (2016).

Preprint. Accepted @ International Journal on Software and Systems Modeling (SoSyM)